Hannah of xHain

(Four-)legged robots for everybody

Matthias Kubisch Felix Just

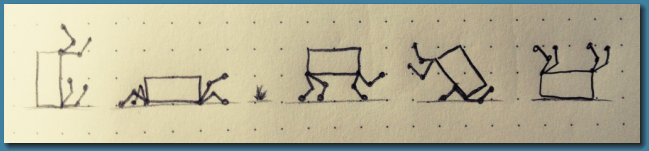
December 28, 2018

THE TEAM

Mario Lasseck, Paul Scheunemann, Thomas Lobig,
Felix Just, Hannah, Matthias Kubisch.

Supreme Machines

Outline



1. Hardware, Principles and Design
2. Electronics and Firmware
3. Simulation, Neural Networks and Motion Control

Our mission.

We want to empower researchers, developers, teachers and makers world-wide to create four-legged robots and their applications on the lowest entry level possible.

We ourselves faced this problem.

Fourlegged robots are **amazing** and have **promissing future applications**, but these days still...

building robots is hard:

- expensive,
- components too heavy,
- complicated to find,
- or closed-source.

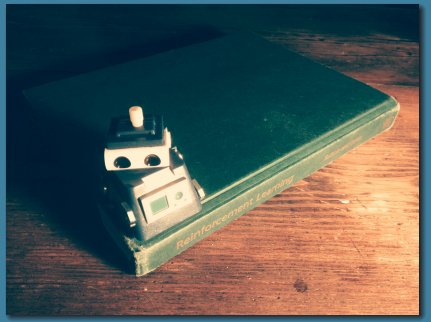
buying robots is hard:

- commercial legged robots are merely toys and often inappropriate
- promissing robots are either:
  - not affordable
  - not (yet) available for purchase at all

Solutions so far

If you are passionate about building four-legged robots today...

- use toy robots if you are doing research or education
- buy high priced robots if you can afford it and work for large research organizations or large enterprises,
- or spend 2 years building your own robot (that's what we did).



...there is relatively little in between :/

After 2 yrs of development...

Hannah

an open-source robot with:

- modular, distributed motor control system
- a lot of sensory feedback
- low material cost, below 2K
- highly available parts
- easy assembly

Why open-source and free hardware-designs?

We all know that:

- free/libre hardware designs needed for education and research
- lower entry-level for startups, more diverse economy
- easy access for creatives and hobbyists
- rapid prototyping without reinventing the wheel

Foto: CC-BY-SA 3.0 Sam Williams

..and most important but not always obvious:

To push the distributed search process for the most sustainable solution.

Hardware Design

Design Principles

No component may restrict the freedom of the overall system's design:

- highly available
- affordable
- replaceable
- repairable

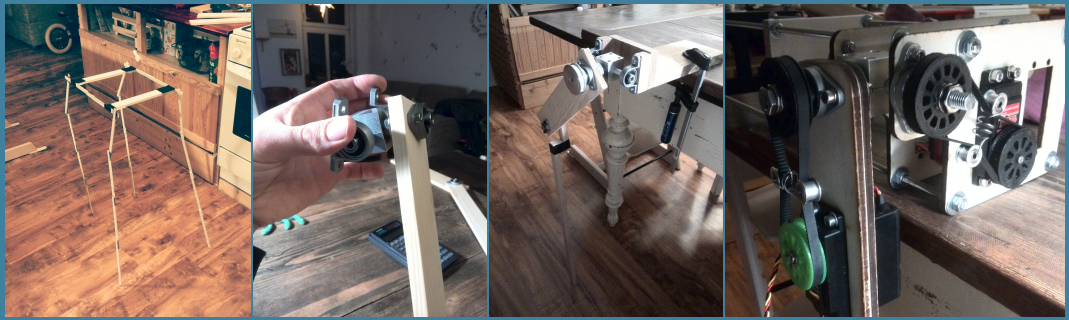
Hence we focused on...

- standard parts (e.g. bearings, nuts), available from different manufacturers
- custom parts which can be 3D printed on home-quality printers
- or laser cut/milled in the makespace
- easy assembly with standard tools

Hopefully in a few years, assembling our legged robot is as easy as repairing a bike.

First test designs

Starting physical and test-driven, not CAD.

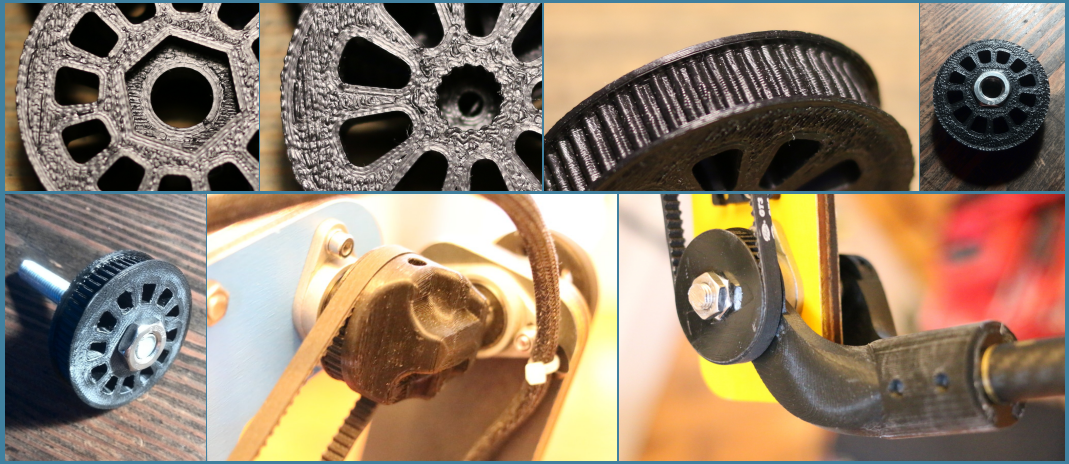


Laser cut parts

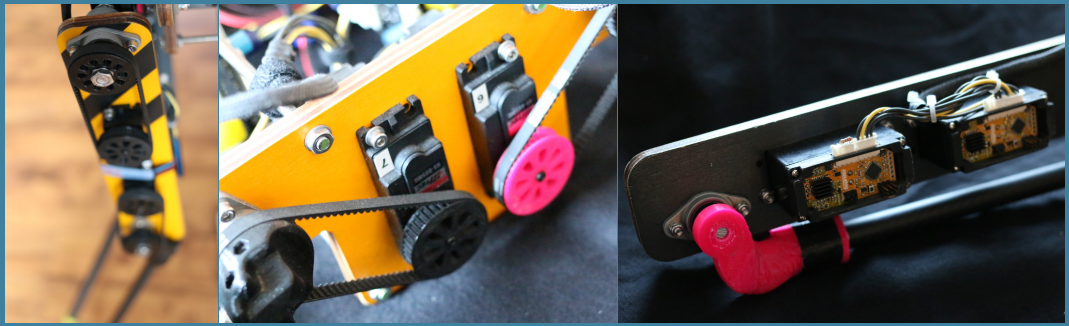
2x 5mm Plywood, glued together and painted, parts could also be milled.

3D printed parts

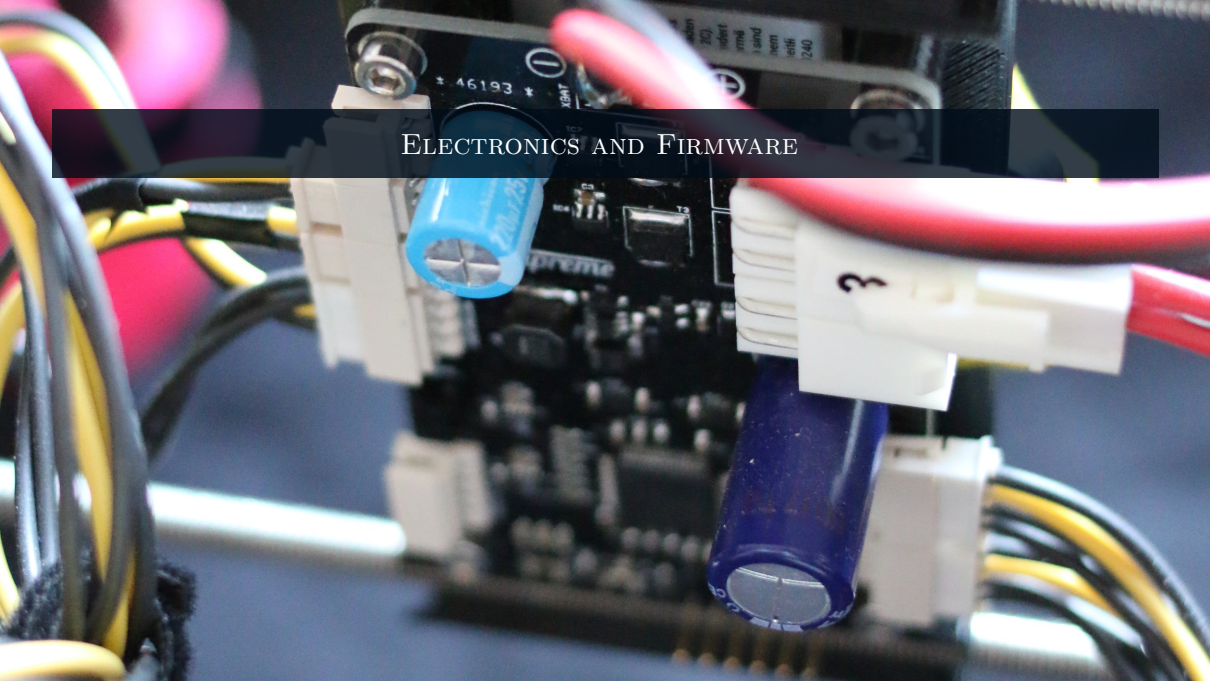
Low-cost home-quality or lab printers suffice. Made of PLA or *tough PLA*, 0.1 – 0.2 mm.



Parts, assembled



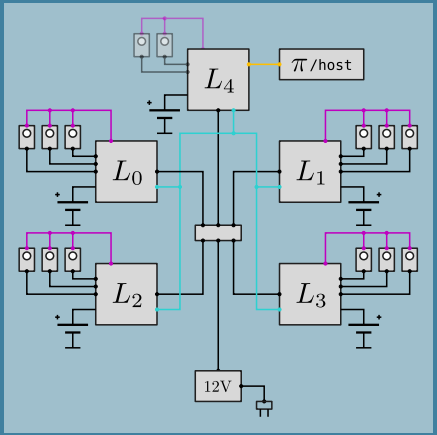
Electronics and Firmware



Design Principles

. . . same as for mechanical parts, but additionally:

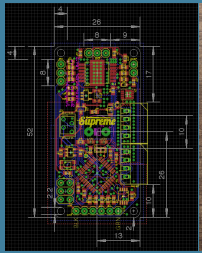
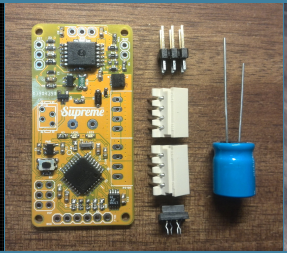
- modular, flexible for different morphologies
- distributed computing and power supply
- lightweight, lean, local



Sensorimotor, local motor control

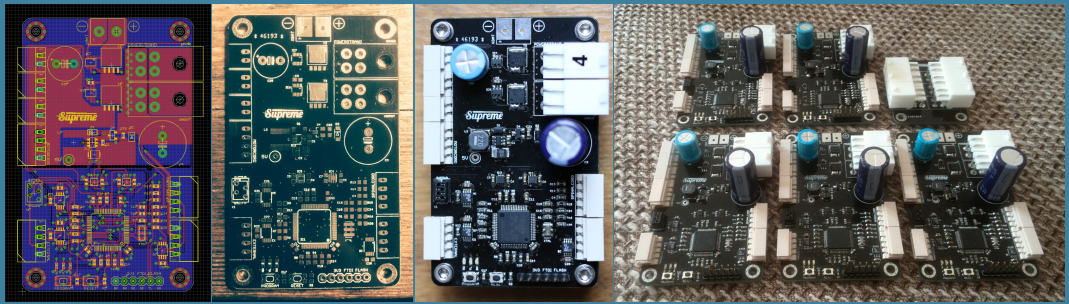
- Atmega 328p
- RS485 communication, 1 MBaud, 256 units
- 6 A integrated h-bridge
- 6–12 VDC input

- position, velocity, current, voltage and temperature sensing
- lean client/server communication protocol
- additional sensors via i2c

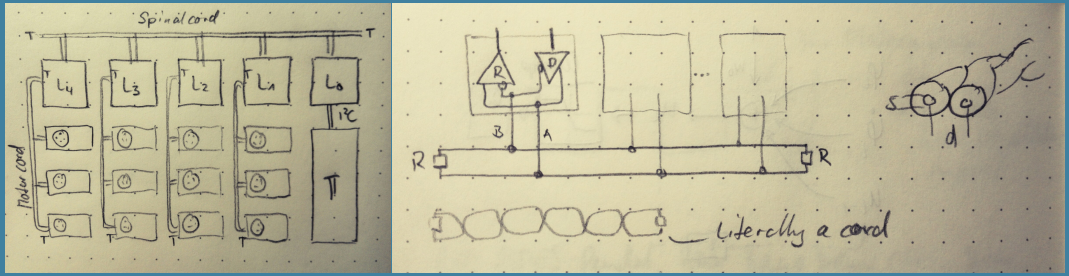



Limbcontroller, local leg control

- STM32F411 with FPU
- 3x RS485 communication (SC, MC, HOST)
- local battery connection through idealized diode
- 6–12 VDC input
- battery voltage sensing
- lean multimaster communication protocol
- additional sensors via i2c
- ~~motors power switchable~~

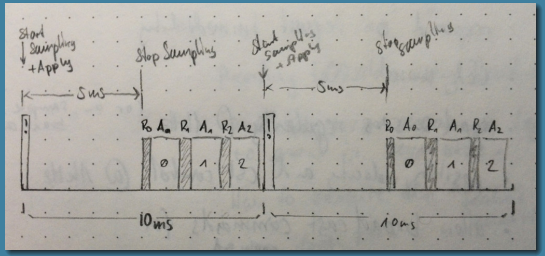


Spinalcord + Motorcord Bus System



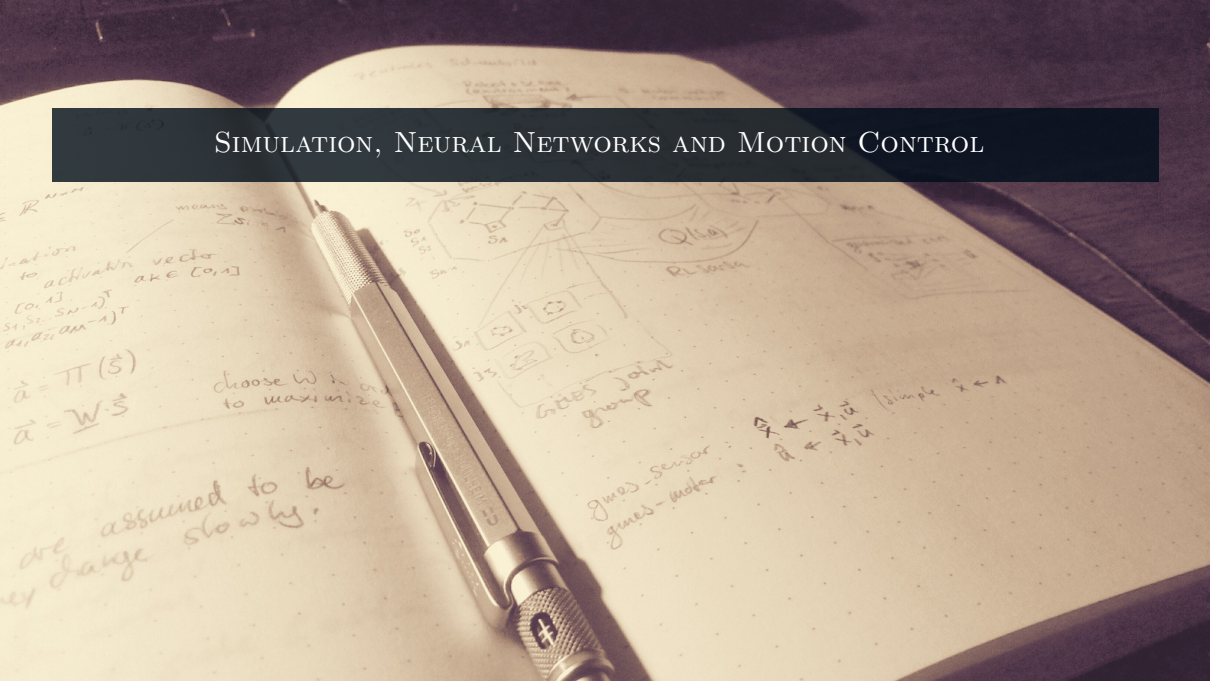
2 symmetric wires (twisted pair), GND + VCC. Termination resistors for reflection absorption.

Firmware + Communication



- firmware cpp11 + scons, using xpcc (modm) framework
- 10ms steps, 100 Hz
- self-synchronizing multimaster com protocol for Limbcontroller
- Request/Response for Sensorimotor

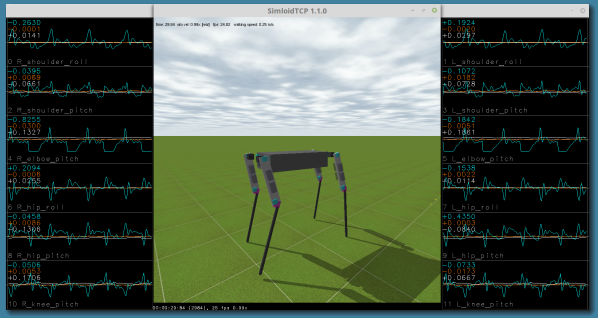
Simulation, Neural Networks and Motion Control



Simulation Environment

Simloid[1]: Robot Simulation Environment

- cycle time 10 ms
- rigid body dynamics
- hinge joints only, hard joint limits
- more realistic joint friction model
- sensor and motor model
- interfaced via TCP/IP commands



[1]Work started by Daniel Hein in 2006.

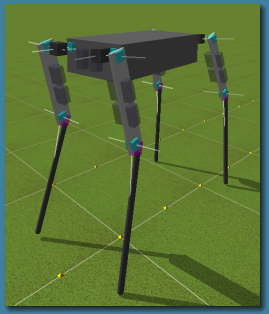
The (Simulated) Fourlegged Robot in Detail

motors:

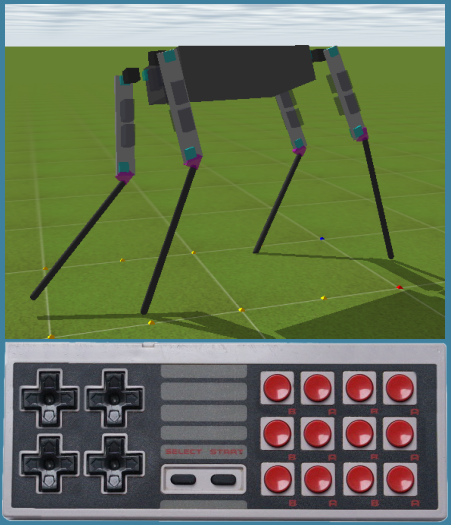
- 12 hinge joints (3 per leg)
- shoulder (2) and knee

sensors:

- 12 joint angles
- 12 angular velocities
- 3-axis acceleration sensor



How to get it controlled?



State and Motor Space Definition

sensor inputs (state): $\vec{x} = \begin{pmatrix} \vec{\varphi} \\ \dot{\vec{\varphi}} \\ \vec{\alpha} \\ 1 \end{pmatrix}$

- raw sensor values
- limited and normalized, $\vec{x} \in [-1, 1]^D$
- $D = 2K + 3 + 1$, with K no. of joints
- $\vec{\varphi} \in [-1, 1]^K$ joint angles
- $\dot{\vec{\varphi}} \in [-1, 1]^K$ joint angular velocities
- $\vec{\alpha} \in [-1, 1]^3$, acceleration sensor, low-pass filtered

motor outputs: $\vec{u}$

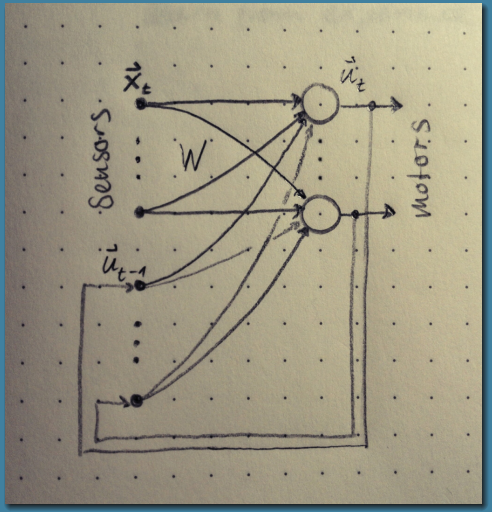
$\vec{u} \in [-1, 1]^K$ as motor voltage

A Generalized Motor Controller

controller equation:

$$\vec{u}_t = f(W\vec{z}_t) \qquad \vec{z}_t = \begin{pmatrix} \vec{x}_t \\ \vec{u}_{t-1} \end{pmatrix}$$

- weight matrix $W \in \mathbb{R}^{K\times(D+K)}$
- transfer function $f(\cdot) = tanh$
- fully connected
- differentiable, i.e. learnable with gradient descend
- closed loop, short term memory
- good tradeoff between simplicity and generality
- model free, platform unspecific



Evolving Locomotion Controller

moving fast:

$$F_s(W_i) = \frac{\Delta s}{T} = \frac{\text{moved distance}}{\text{trial time}}$$

moving efficiently:

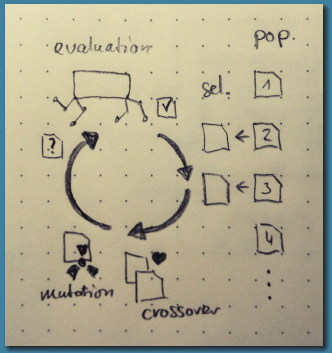
$$F_e(W_i) = \frac{\Delta s}{\sum_t ||u_t||_2} = \frac{\text{moved distance}}{\text{motor outputs}}$$

Constraints:

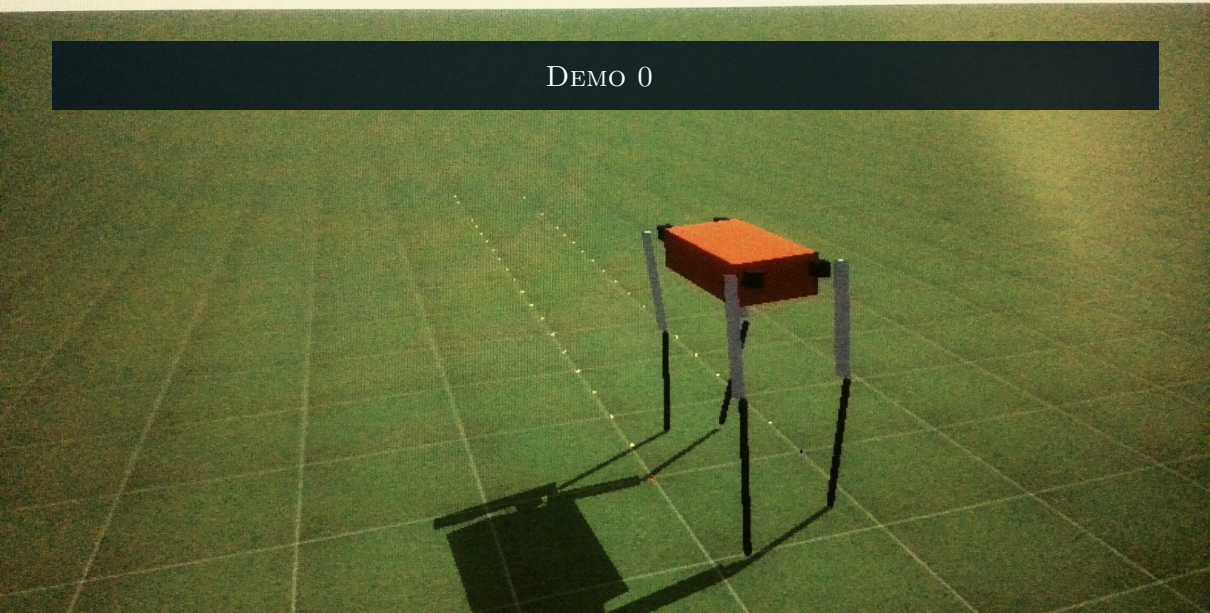
- Dropped?
- Out of track?
- Not Moving at all?

→ Terminate trial, reduce fitness.

Initial Conditions (seed):
Initialize weights as poorly tuned PD controllers (which tend to oscillate).



Demo 0



Transferring Gaits to Real Hardware. Bridging the Simulation Gap.

Domain randomization: For each trial, learn on different simulated robots with randomized parameters and a lot of disturbance, e.g. obstacles, nudges.

Randomize all...

- weights, sizes, motormodel
- friction, stiction, noisy sensors,
- objects in the robots way.

With the hope that the real platform is close enough to the mean of that distribution to walk stably.

Record motor test data with single leg (pendulum).

Next steps, a lot to do.

general: push documentation, increase usablity, simplify everything.

hardware:

- make the platform more robust
- solid mechanical endstops
- stronger motors, better belt tension
- reduce weight

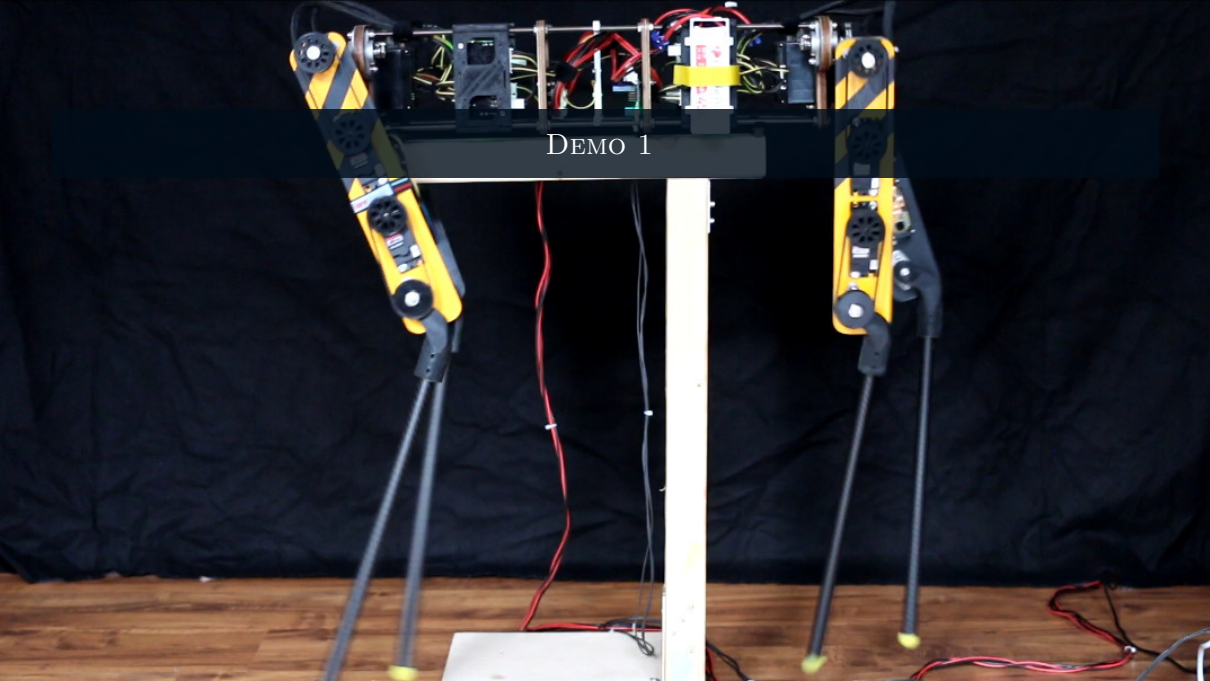
electronics:

- Sensorimotor 'cargo' (40 Amps, same interface)
- magnetic position/velocity encoders
- foot-contact sensors
- one accel sensor on each sensorimotor pcb

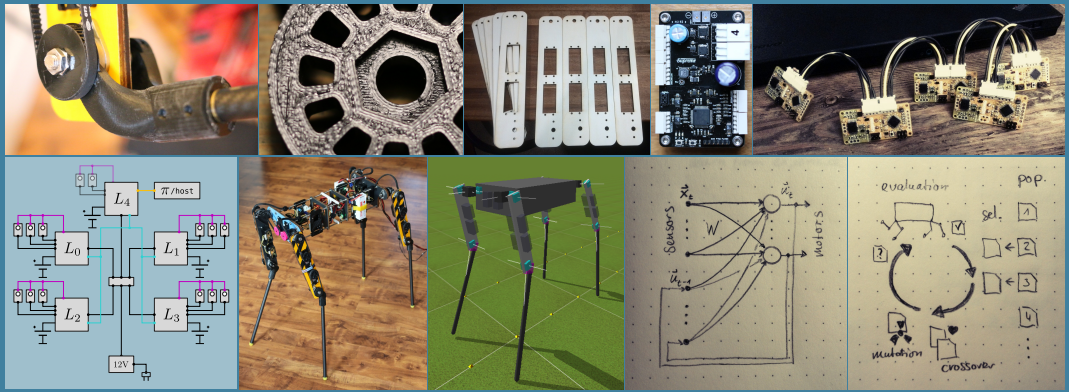
sim+ctrl:

- morphological evolution to find better body proportions (e.g. leg length)
- increase walking speed, different behaviors, closing simulation gap
- make networks run on limbcontroller
- autocalibration method using gravity as reference

But the most desirable thing is,
to get the robot up and running.


Demo 1

Q&A


L_4
π/host
L_0
L_1
L_2
L_3
12V
Sensors
Motors
W
evaluation
pop.
sel.
mutation
crossover