fast line following

- **baseline**
  - constant speed
  - PD controller for steering
  - PIDs for motors controll

- **neural network**
  - automatic target speed estimating using convolutional neural network
  - real time curve shape classification (200Hz)
  - robot can accelerate on straight line

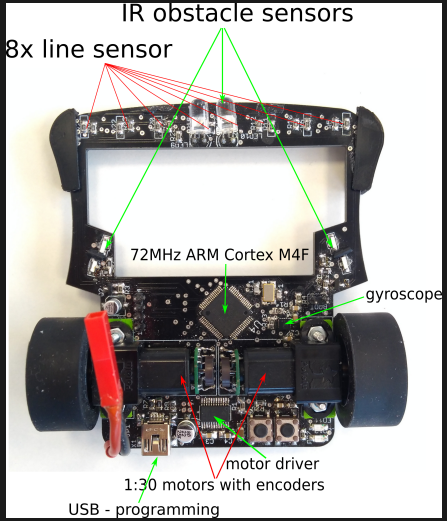
steering controll quality

- PD controller for steering
- two PIDs for motors controll

other features

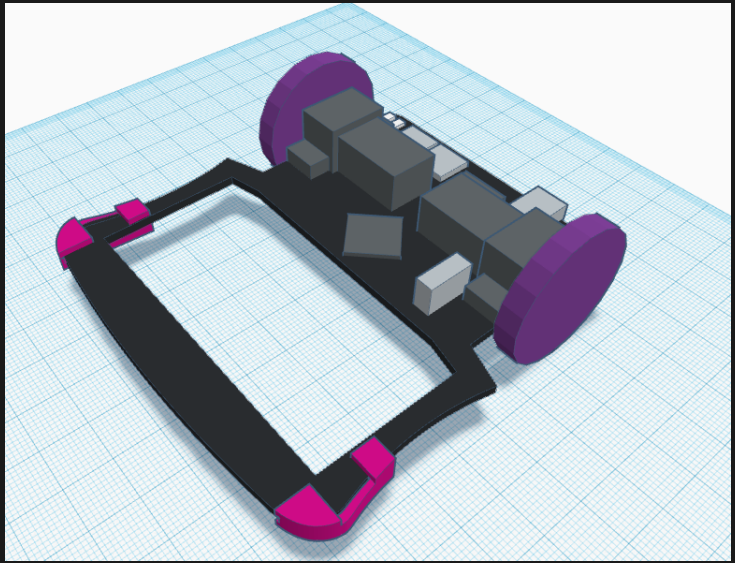
- broken line search
- color line following

hardware

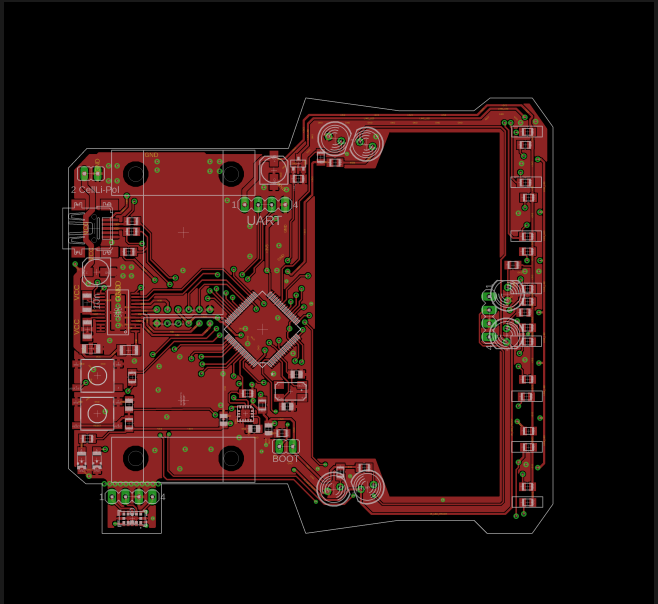


- MCU stm32f303, ARM Cortex M4F
  - 72MHz
  - 32bit floating point unit
  - SIMD DSP instructions for deep learning
- POLOLU micro metal gear motors
  - gear ratio 1:30
  - 180 pulses quadrature encoders
  - DRV8834 H-Bridge driver
- Sensors
  - 8x phototransistor + white LED, 540nm
  - 3xIR leds for obstacle detection
  - gyroscope + accelerometer, LSM6DS0

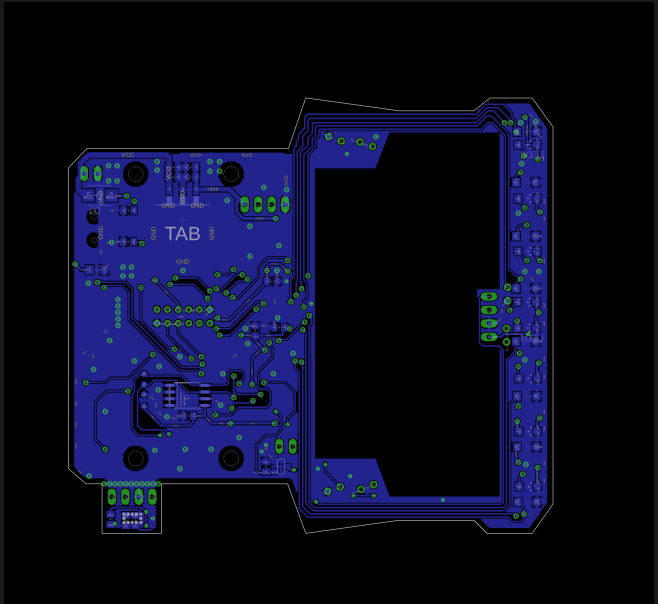
3D model for simulations



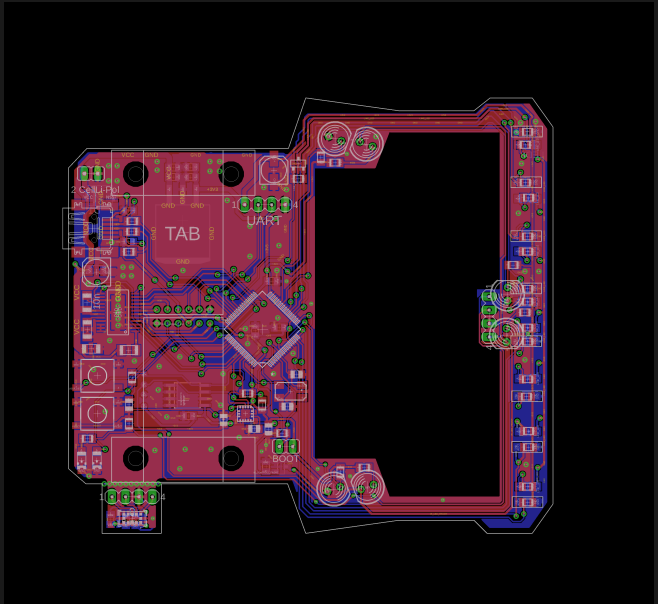
PCB - top layer



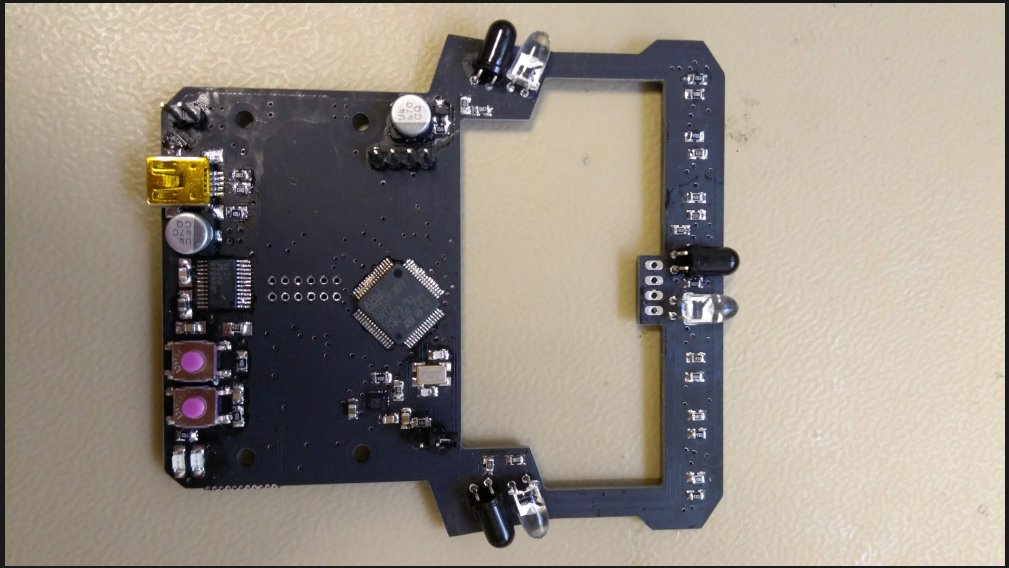
PCB - bottom layer



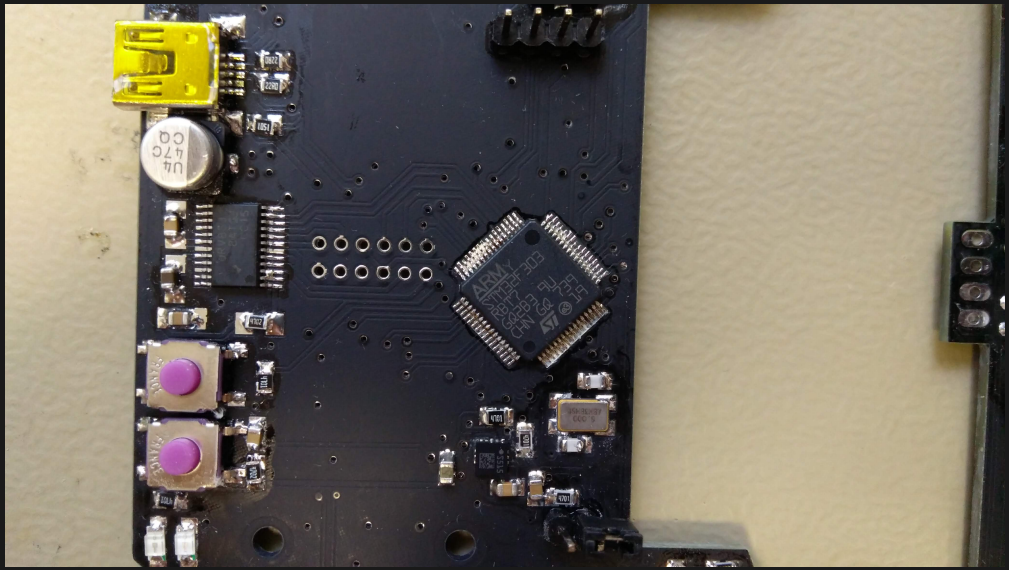
PCB - design



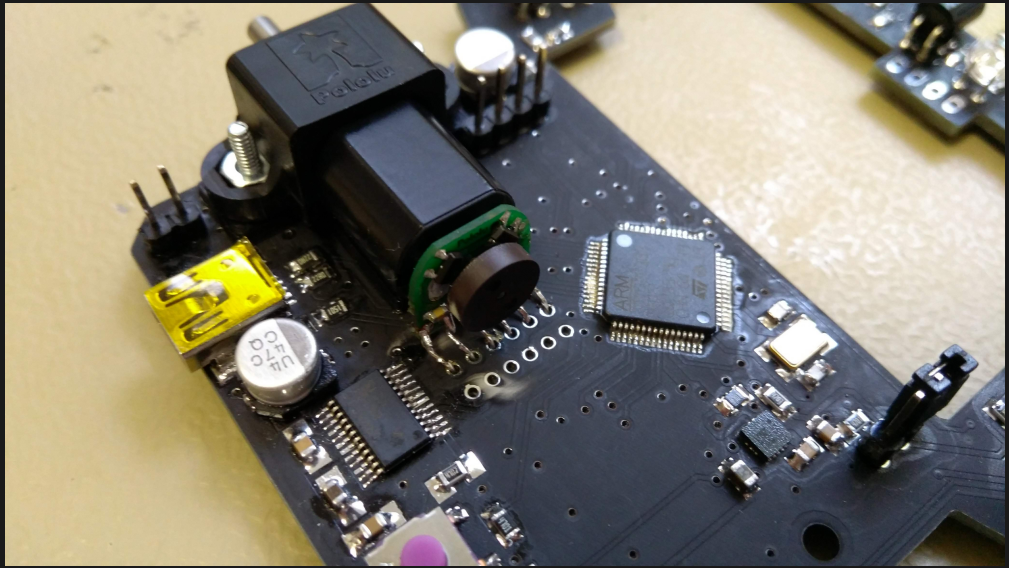
assembling process



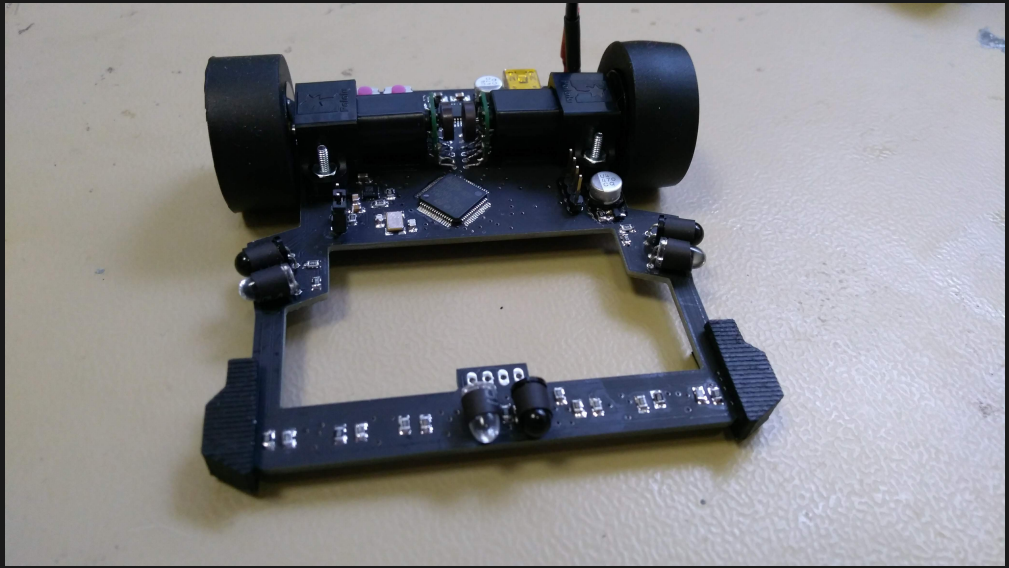
assembling process - soldering detail



assembling process - motor connection



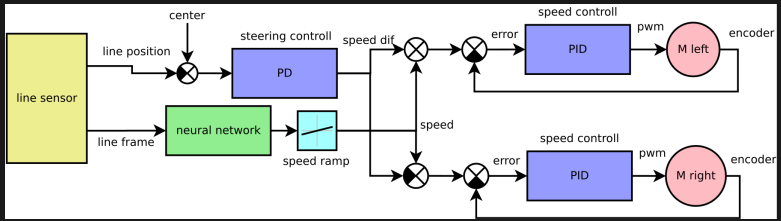
assembling process - complete robot



software

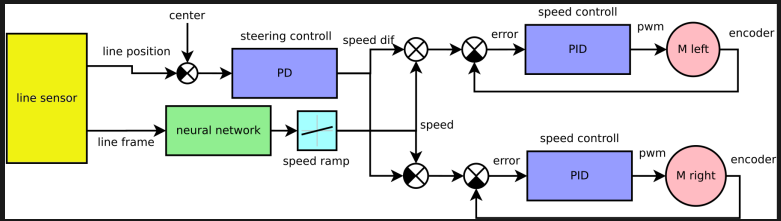
- robot firmware is very complex - **thousands of lines** of C++ code
- there is custom C++11 templates hardware abstraction layer
- custom event driven operating system - all sensors tasks are running on background
- programmed without IDE, just simple text editor,
  sources are availible on :
  https://github.com/michalnand/motoko_uprising

controll process



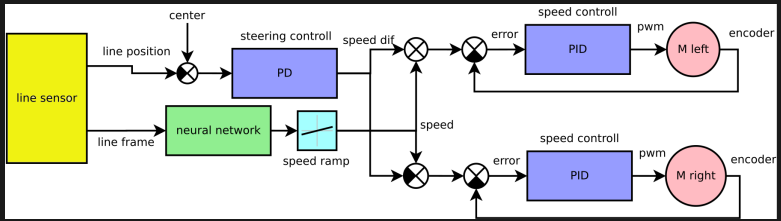
controll process - sensor reading

1. the **line sensors** are readed using ADC



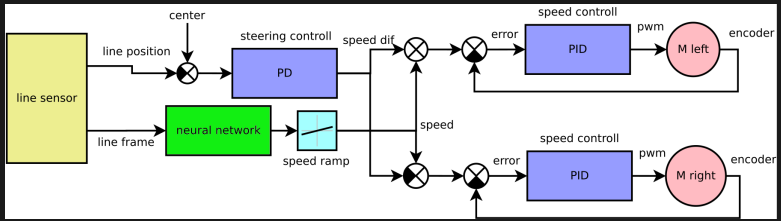
controll process - sensor reading

1. the **line sensors** are readed using ADC
   - thanks quadratic position interpolation, it provides 512 possible line positions



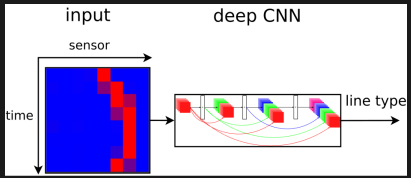
controll process - neural network

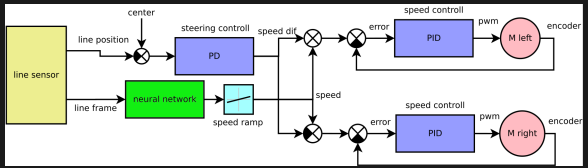
2. the **neural network** performs curve shape classification



controll process - neural network

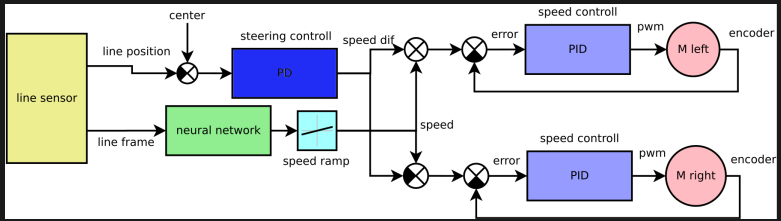
2. the **neural network** performs curve shape classification
   - input is matrix, 8x8, eight past line sensors readings
   - output is one of 5 classes :
     sharp left, soft left, center line, soft right, or sharp right
   - network runtime is only 2 miliseconds





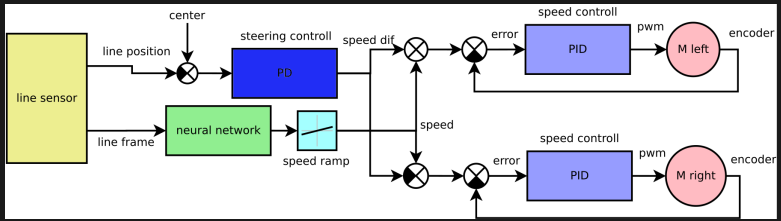
controll process - steering

3. the **steering** PD controller compute speed difference



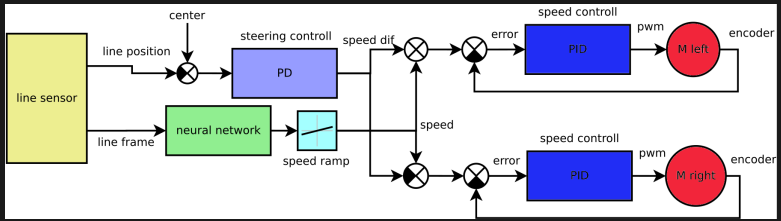
controll process - steering

3. the **steering** PD controller compute speed difference
   - of course, the filtered derivative term is used



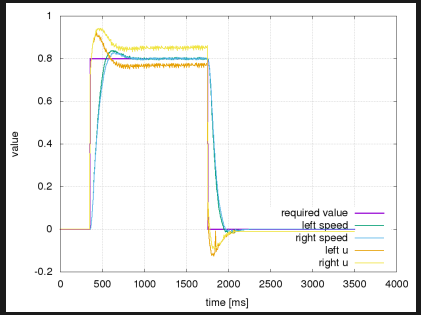
controll process - motors controll

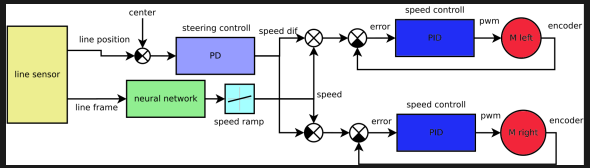
3. the **motors speed** is controlled with two PIDs



controll process - motors controll

3. the **motors speed** is controlled with two PIDs
   - PIDs are using antiwindup, so the response looks quick





end to end line follower

- **reinforcement learning** line follower

end to end line follower

- **reinforcement learning** line follower
- **no PD, PIDs, no dataset**
  -just learned from rewards and pushments

end to end line follower

- **reinforcement learning** line follower
- **no PD, PIDs, no dataset**
  -just learned from rewards and pushments
- **self understand** what the line is, what the goal is ...

end to end line follower

- **reinforcement learning** line follower
- **no PD, PIDs, no dataset**
  -just learned from rewards and pushments
- **self understand** what the line is, what the goal is ...
- **advantage actor critic** algorithm

end to end line follower

- **reinforcement learning** line follower
- **no PD, PIDs, no dataset**
  -just learned from rewards and pushments
- **self understand** what the line is, what the goal is ...
- **advantage actor critic** algorithm
- here is the magic

$$\begin{aligned}\mathcal{L} =& (V - \hat{V})^2 + \\ & - log(p(A|S))(V - \hat{V}) \\ & - \beta \mathcal{H}(A)\end{aligned}$$

Thanks for watching