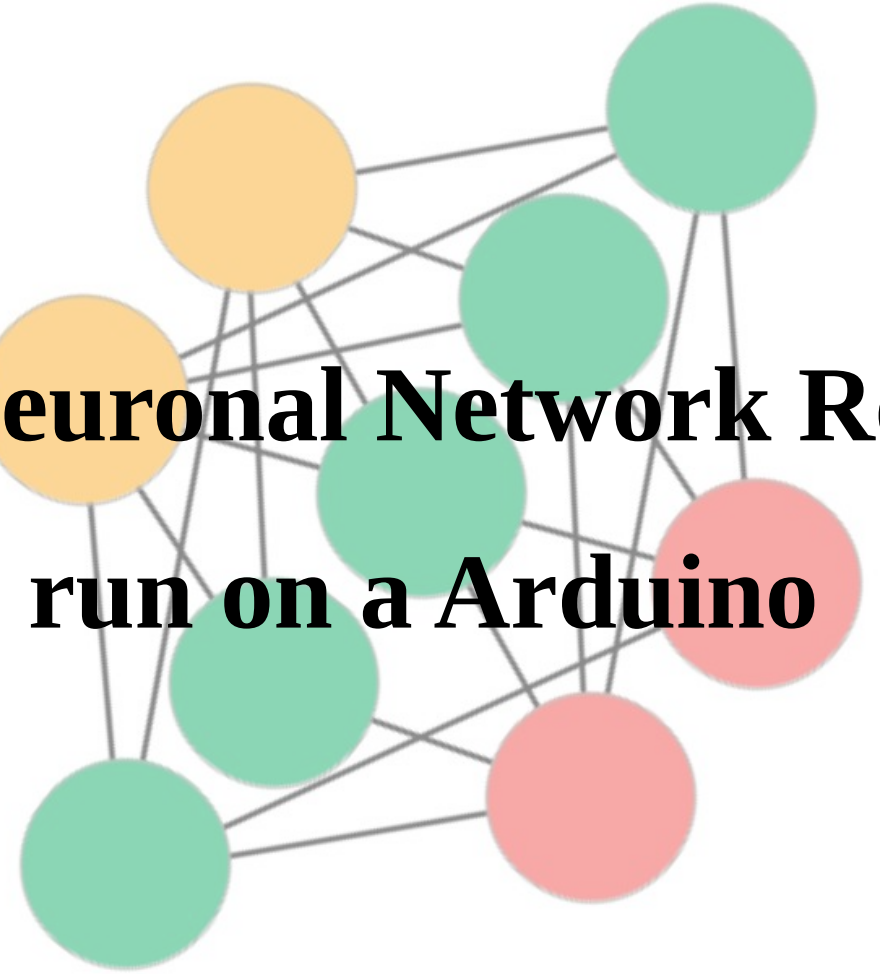
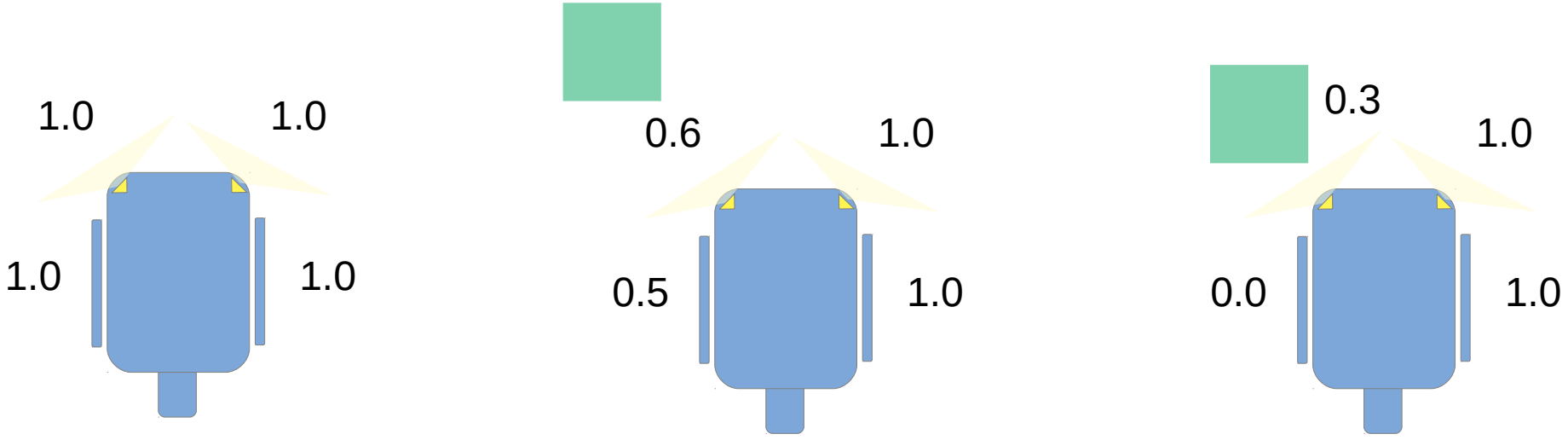
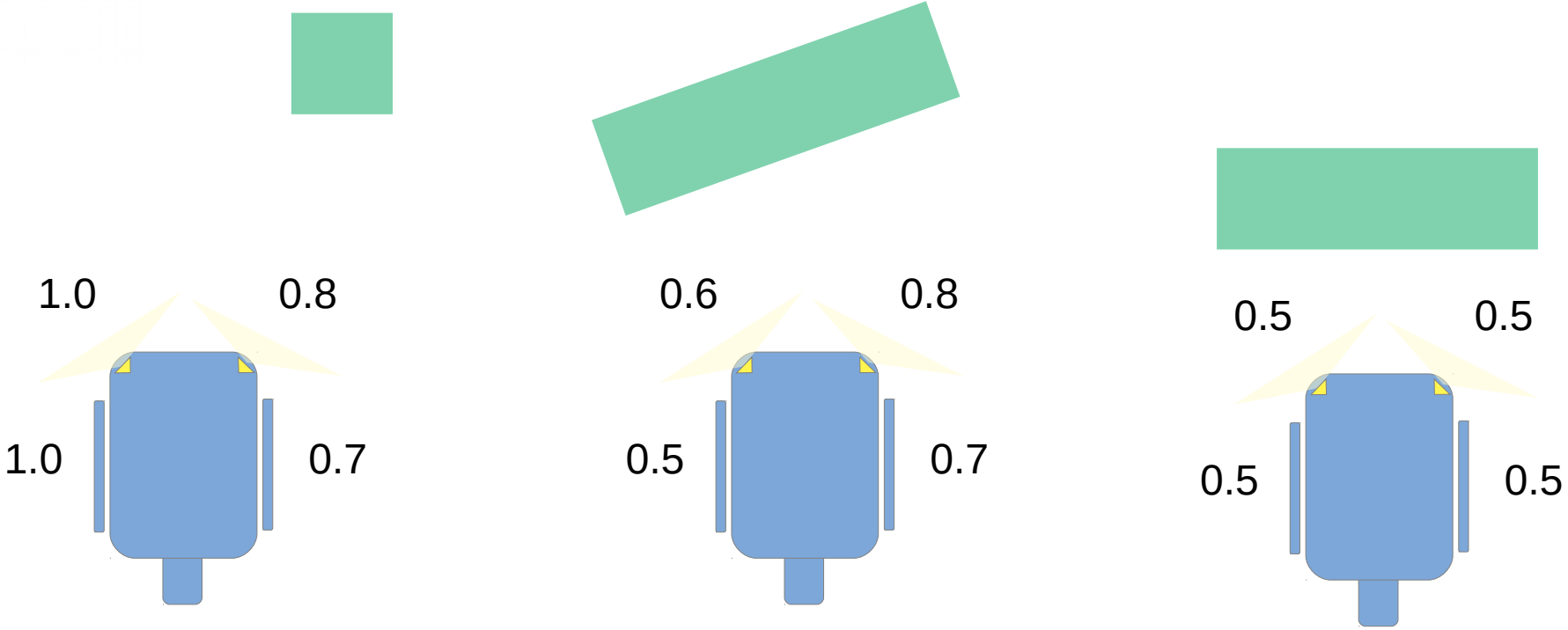
<u>Simple</u> Neuronal Network Robot Bot run on a Arduino



Follower Logic 1/2



Follower Logic 2/2



Follower Logic Target / Formula

Defined Target:

Distance	Speed
1.0	1.0
0.8	0.7
0.5	0.4
0.3	0

Formula Calculation:

Speed = (Distance – 0.3) / 0.7

Distance	Speed
1.00	1.00
0.80	0.71
0.50	0.29
0.30	0.00

Follower Logic Arduino Implementation

follower_bot_formula | Arduino 1.8.9

File Edit Sketch Tools Help

follower_bot_formula §

```
left_ir_f = left_ir_f / 100;

 Serial.print(" R IR: ");
 dtostrf(right_ir_f*100,6,3,f);
 Serial.print(f);

 Serial.print(" L IR: ");
 dtostrf(left_ir_f*100,6,3,f);
 Serial.print(f);

  rspeed = ( right_ir_f - 0.3 ) / 0.7;
  lspeed = ( left_ir_f - 0.3 ) / 0.7;

 /*
```

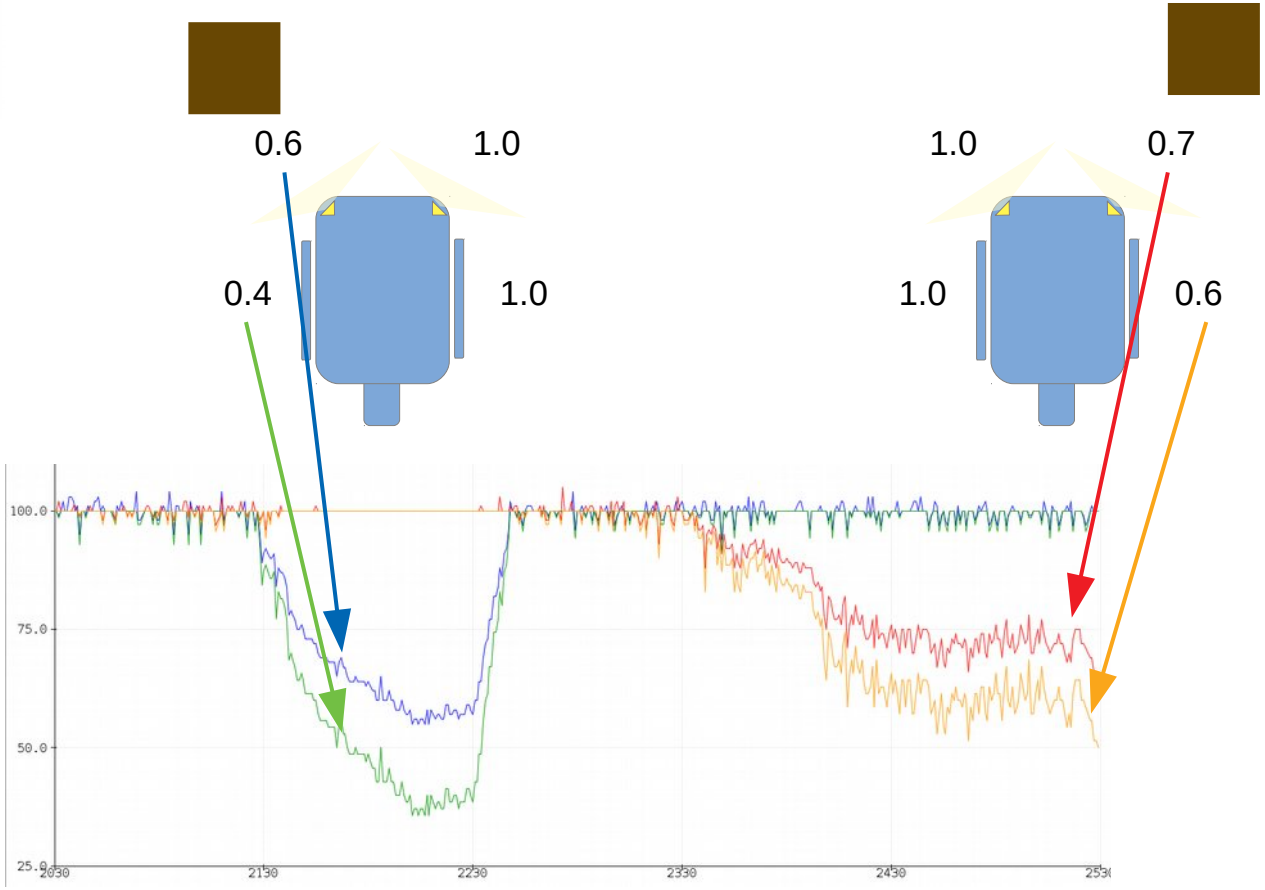
Speed = (Distance – 0.3) / 0.7

```
avrdude: verifying flash memory against /tmp/arduino_build_101252/follower_bot_formula.ino.hex:
avrdude: load data flash data from input file /tmp/arduino_build_101252/follower_bot_formula.ino.hex:
avrdude: input file /tmp/arduino_build_101252/follower_bot_formula.ino.hex contains 7072 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 3.07s
```

284 Arduino Duemilanove or Diecimila on /dev/ttyUSB0

Follower Testrun / Formula



Follower Logic Target / Neuronal Network

Defined Target (same):

Distance	Speed
1.0	1.0
0.8	0.7
0.5	0.4
0.3	0

Follower Logic Target / Neuronal Network

Manual Training Data

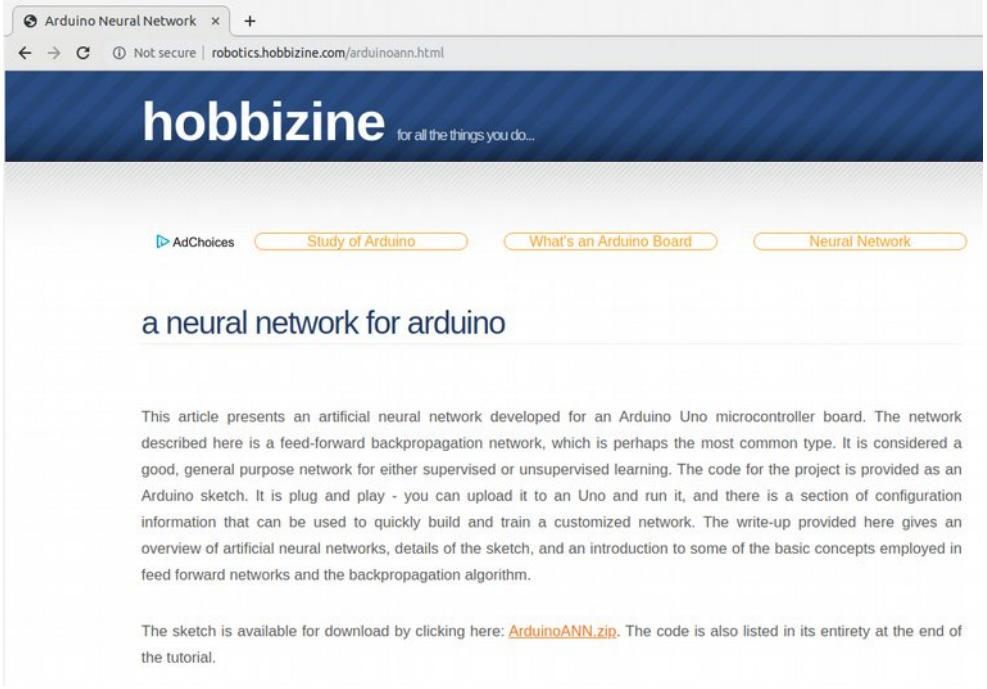
Distance		Motor		Detail
Left	Right	Left	Right	
1.00	1.00	1.00	1.00	No Obstaceles
0.80	0.80	0.70	0.70	Obstacle in the middle but far
0.50	1.00	0.40	1.00	Obstacle at the left
1.00	0.50	1.00	0.40	Obstacle at the right
0.30	0.30	0.10	0.10	Obstacle in the middle nearby
0.00	0.00	0.00	0.00	Obstacle in the middle very close

Neuronal Network Setup

- Input Nodes = 2
- Output Nodes = 2
- Number of Trainings records = 6
- Hidden Nodes = 5
- Learning Rate = 0.3
- Momentum = 0.9

Arduino Implementation

- Coding based on http://robotics.hobbizine.com/arduinoann.html



Neuronal Network Visualization 1/2

three-layer feed-forward Network
algorithm: Gradient descent

Learning Rate: 0.3

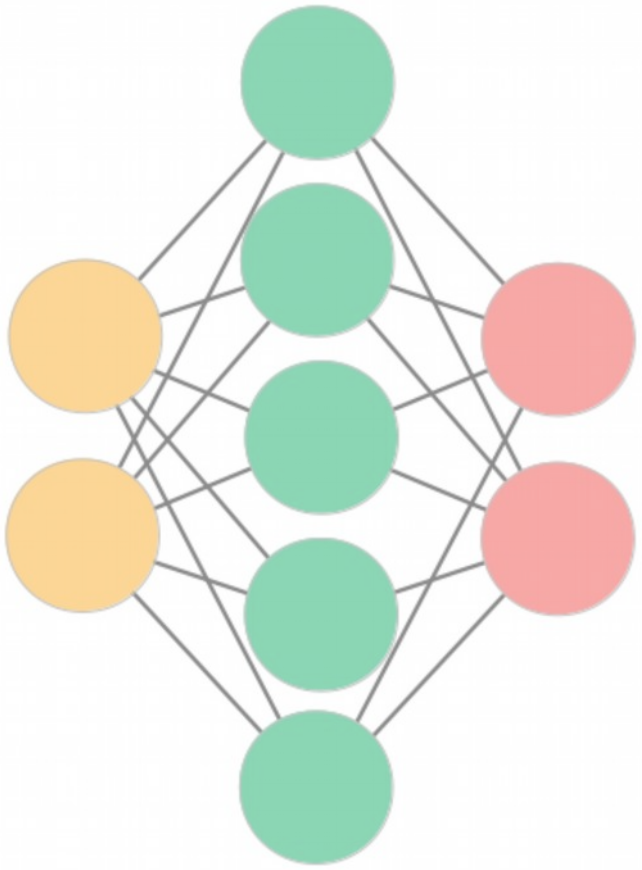
Adjusts how much of the error is actually backpropagated.

Momentum: 0.9

Adjusts how much the results of the previous iteration affect the current iteration.

Minimal Success: 0.02

The threshold for error at which the network will be said to have solved the training set.

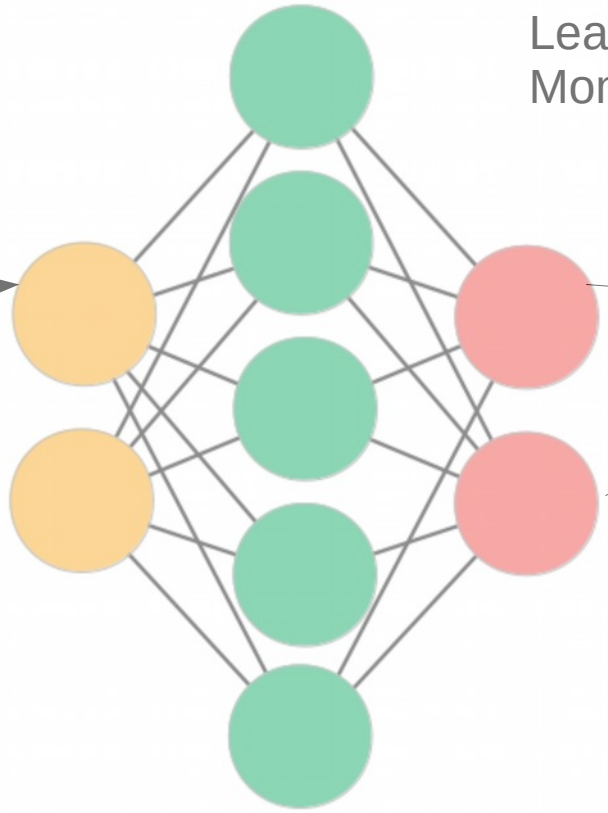


Input (2) Hidden (5) Output (2)

Neuronal Network Visualization 2/2

Learning Rate: 0.3
Momentum: 0.9

Distance	
Left	Right
1.00	1.00
0.80	0.80
0.50	1.00
1.00	0.50
0.30	0.30
0.00	0.00
?	?



Motor	
Left	Right
1.00	1.00
0.70	0.70
0.40	1.00
1.00	0.40
0.10	0.10
0.00	0.00
?	?

Input Hidden Output

Arduino Neuronal Network Training

/dev/ttyUSB0

```
Robot Starting with training
Initial/Untrained Outputs:

  Training Pattern:  0.000  Input: 1.00 1.00   Target: 1.00 1.00   Output 0.23 0.58
  Training Pattern:  1.000  Input: 0.80 0.80   Target: 0.70 0.70   Output 0.23 0.58
  Training Pattern:  2.000  Input: 0.50 1.00   Target: 0.40 1.00   Output 0.23 0.59
  Training Pattern:  3.000  Input: 1.00 0.50   Target: 1.00 0.40   Output 0.23 0.58
  Training Pattern:  4.000  Input: 0.30 0.30   Target: 0.10 0.10   Output 0.23 0.58
  Training Pattern:  5.000  Input: 0.00 0.00   Target: 0.00 0.00   Output 0.23 0.59 TrainingCycle:  1.000  Error:  1.240

  Training Pattern:  0.000  Input: 1.00 1.00   Target: 1.00 1.00   Output 0.41 0.57
  Training Pattern:  1.000  Input: 0.80 0.80   Target: 0.70 0.70   Output 0.41 0.57
  Training Pattern:  2.000  Input: 0.50 1.00   Target: 0.40 1.00   Output 0.41 0.57
  Training Pattern:  3.000  Input: 1.00 0.50   Target: 1.00 0.40   Output 0.41 0.57
  Training Pattern:  4.000  Input: 0.30 0.30   Target: 0.10 0.10   Output 0.41 0.57
  Training Pattern:  5.000  Input: 0.00 0.00   Target: 0.00 0.00   Output 0.41 0.57 TrainingCycle: : 208.000  Error:  0.020

  Training Pattern:  0.000  Input: 1.00 1.00   Target: 1.00 1.00   Output 0.92 0.92
  Training Pattern:  1.000  Input: 0.80 0.80   Target: 0.70 0.70   Output 0.77 0.78
  Training Pattern:  2.000  Input: 0.50 1.00   Target: 0.40 1.00   Output 0.37 0.92
  Training Pattern:  3.000  Input: 1.00 0.50   Target: 1.00 0.40   Output 0.92 0.37
  Training Pattern:  4.000  Input: 0.30 0.30   Target: 0.10 0.10   Output 0.10 0.10
  Training Pattern:  5.000  Input: 0.00 0.00   Target: 0.00 0.00   Output 0.02 0.02

Training Set Solved!  Error:  0.020
```

Trainings:

1.

208

3.

Output Test

Last Error Rate 0.020

Arduino Neuronal Network Results

Distance		Target Motor		NN Motor	
Left	Right	Left	Right	Left	Right
1.00	1.00	1.00	1.00	0.92	0.92
0.80	0.80	0.70	0.70	0.77	0.78
0.50	1.00	0.40	1.00	0.37	0.92
1.00	0.50	1.00	0.40	0.92	0.37
0.30	0.30	0.10	0.10	0.10	0.10
0.00	0.00	0.00	0.00	0.02	0.02

Last Error Rate 0.020

Distance		Target Motor		NN Motor	
Left	Right	Left	Right	Left	Right
1.00	1.00	1.00	1.00	0.97	0.97
0.80	0.80	0.70	0.70	0.71	0.710
0.50	1.00	0.40	1.00	0.40	0.98
1.00	0.50	1.00	0.40	0.99	0.40
0.30	0.30	0.10	0.10	0.09	0.09
0.00	0.00	0.00	0.00	0.02	0.03

Last Error Rate 0.002

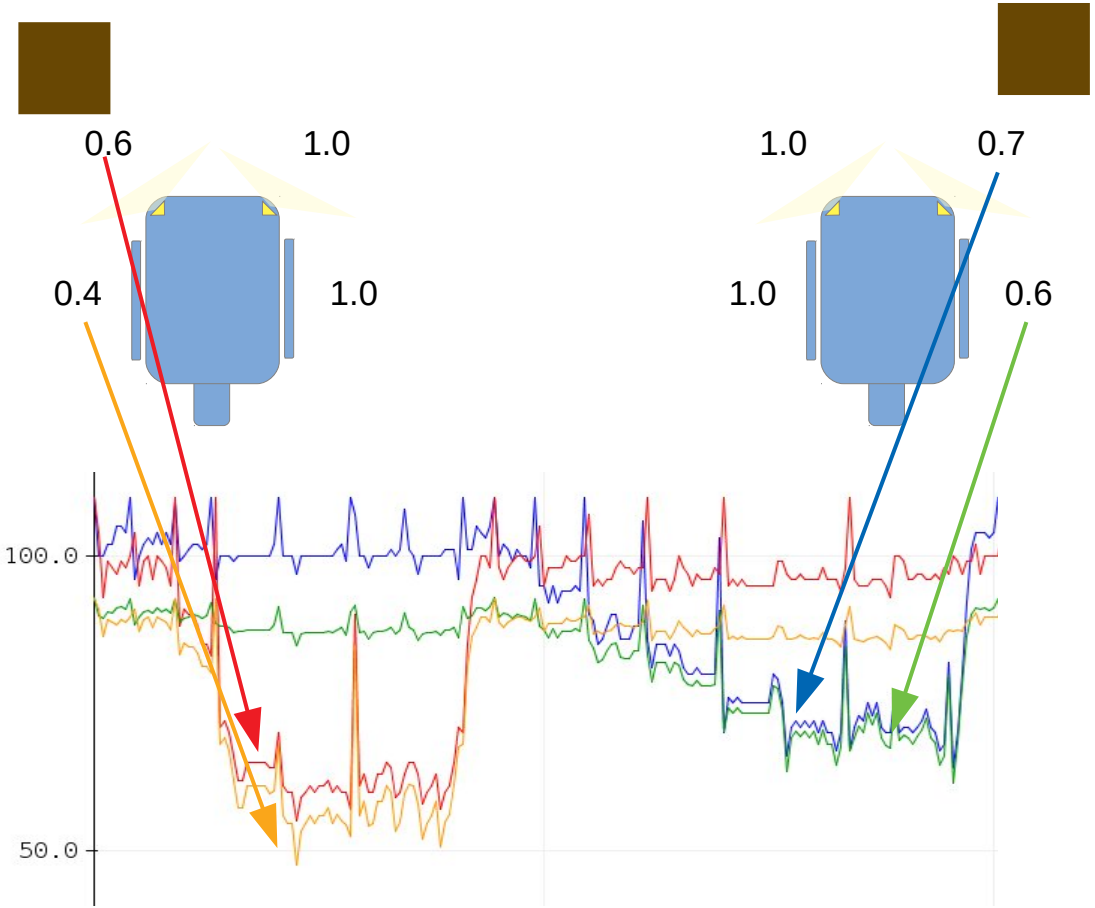
Arduino Neuronal Network Tests

Real values not in Training Data:

```
Test Run:
Test Input:  L IR:  0.900 R IR:  0.900 Out L Mot:  0.890 R Mot:  0.892
Test Input:  L IR:  0.400 R IR:  0.900 Out R Mot:  0.270 L Mot:  0.947
Test Input:  L IR:  0.700 R IR:  0.300 Out L Mot:  0.688 R Mot:  0.148
```

Distance		Target Motor Formula Speed = (Distance – 0.3) / 0.7		NN Motor		Accurancy	
Left	Right	Left	Right	Left	Right	Left	Right
0.90	0.90	0.86	0.86	0.89	0.89	0.03	0.03
0.40	0.90	0.14	0.86	0.27	0.95	0.13	0.09
0.70	0.30	0.57	0.00	0.69	0.15	0.12	0.15

Follower Testrun / Neuronal Network



Neuronal Network change Functionality: “Evasion”

Motor speed definition:
1.0 => Full forward
0.5 => stop
0.0 => Full backwards

Distance		Motor		Detail
Left	Right	Left	Right	
1.00	1.00	1.00	1.00	No Obstaceles => full forward
0.80	0.80	0.90	0.90	Obstacle in the middle but far => slow down
0.40	1.00	0.70	0.20	Obstacle at the left => Right turn
1.00	0.40	0.20	0.70	Obstacle at the right => Left turn
0.30	0.30	0.10	0.10	Obstacle in the middle nearby => backwards
0.00	0.00	0.00	0.00	Obstacle in the middle very close => full backwards

Challenges

- Build samples with some kind of learing drive
- Use this Neuronal Network for balancing with a accelerator
- Try other functionality

Summarize

- A simple Neuronal Network works on Arduino
- No need of a huge set of testdata
- Change of “Functionality” easy: Follower => Evasion