Building Lego Robots with Elixir

Torben Hoffmann
CTO @ Erlang Solutions
torben.hoffmann@erlang-solutions.com
@LeHoff

Source: https://www.frontlinegaming.org/wp-content/uploads/2013/01/4e_DnD_Orcs_by_RalphHorsley1.jpeg

Source: https://www.frontlinegaming.org/wp-content/uploads/2013/01/4e_DnD_Orcs_by_RalphHorsley1.jpeg

Syntax Is Irrelevant

Thinking Is Everything

Language Shapes Thinking

Language is Everything

First Encounter 2008

Source: http://www.parkmania.pl/image.php?type=photo&id=102_6817&size=517x377&method=cutout

First Encounter 2008

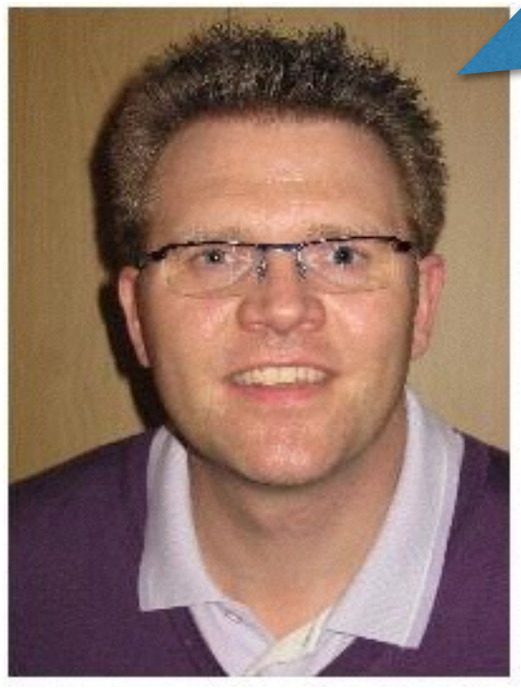
Source: http://static.theglobeandmail.ca/ecb/migration_catalog/article3999938.ece/ALTERNATES/w620/father+son+parent+family+computer+money+advice+play+learn+trust+

First Encounter 2008

Source: https://encrypted.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0CAYQjB0&url=http%3A%2F%2Fwww.tripadvisor.co.uk%2FLocationPhotoDirectLink-g33388-i75792856-Denver_Colorado.html&ei=oq82VYW4MMeUsAG1r4CYDg&bvm=bv.91071109,d.bGg&psig=AFQjCNEfmNbHMXpJyYvCsXIzdHuoLioa-w&ust=1429733539622296

2013 X-mas Wishlist

This time it will work!



Source: https://encrypted.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0CAYQjB0&url=http%3A%2F%2Fwww.tripadvisor.co.uk%2FLocationPhotoDirectLink-g33388-i75792856-Denver_Colorado.html&ei=oq82VYW4MMeUsAG1r4CYDg&bvm=bv.91071109,d.bGg&psig=AFQjCNEfmNbHMXpJyYvCsXIzdHuoLioa-w&ust=1429733539622296

Sisyfos 2013

Source: https://encrypted.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0CAYQjB0&url=http%3A%2F%2Fwww.varmdalkraft.se%2F%3Fp%3D257&ei=C7E2VYjjK4yxsAHmpIDgBg&bvm=bv.91071109,d.bGg&psig=AFQjCNHj2MtLAFZ_9uQokWaJPkjMEx8iYw&ust=1429733958290575

Elixir to the Rescue
2015

Getting Closer…

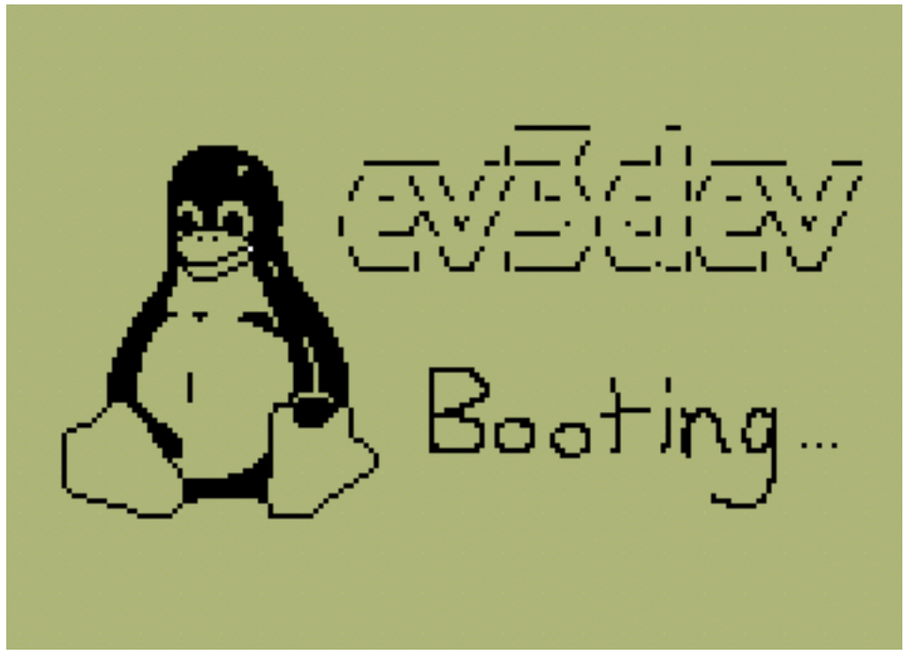
My son's reaction:

"I can understand this code!"

My conclusion:

Elixir fits… when you can program a bit!!

Getting Elixir on EV3



[ev3dev.org](ev3dev.org)

Linux on a SD Card

Install Erlang...

```
apt-get install Erlang
```

Unzip precompiled Elixir

File Alchemy

ev3dev provides a file based interface

When devices are connected they pop up in the file system

E.g., connect a motor:

```
/sys/class/tacho-motor/motor0
```

Which files?

```
root@ev3dev:/sys/class/tacho-motor/motor0# ls
device          polarity_modes     pulses_per_second_sp  run_modes             subsystem
duty_cycle      port_name          ramp_down_sp          speed_regulation_D    time_sp
duty_cycle_sp   position           ramp_up_sp            speed_regulation_I    type
encoder_mode    position_mode      regulation_mode       speed_regulation_K    uevent
encoder_modes   position_modes     regulation_modes      speed_regulation_P
estop           position_sp        reset                 state
log             power              run                   stop_mode
polarity_mode   pulses_per_second  run_mode              stop_modes
root@ev3dev:/sys/class/tacho-motor/motor0#
```

Most of this is documented on [ev3dev.org]

Get Your Motors Running

Write a number -100..100 to

`duty_cycle_sp`

Write 1 to

`run`

Pain Killer

Macro to define properties

```
def_motor_property "run",          writable: true, min: 0, max: 1
def_motor_property "duty_cycle", writable: false
def_motor_property "stop_mode",
      writable: true, modes: [:coast, :brake, :hold]
```

Reading Values

```
# define get function
    getter = quote do
      def unquote(get_function_name)(port) do
        unquote(port_check)
        EV3.Motors.get(port)
        |> elem(1)
        |> Path.join(unquote(property))
        |> EV3.Util.read!
        |> EV3.Util.string_to_integer_or_atom
      end
  end
```

Look, Ma! No Hands!

```
EV3.Motor.forward(:outB, 50)
EV3.Motor.forward(:outC, 30)
color = EV3.ColorSensor.get_color(set_mode: false)
```

Show me some Lego!

Source:
http://youthvoices.net/sites/default/files/image/26182/jan/chicken001.jpg

Naïve Approach

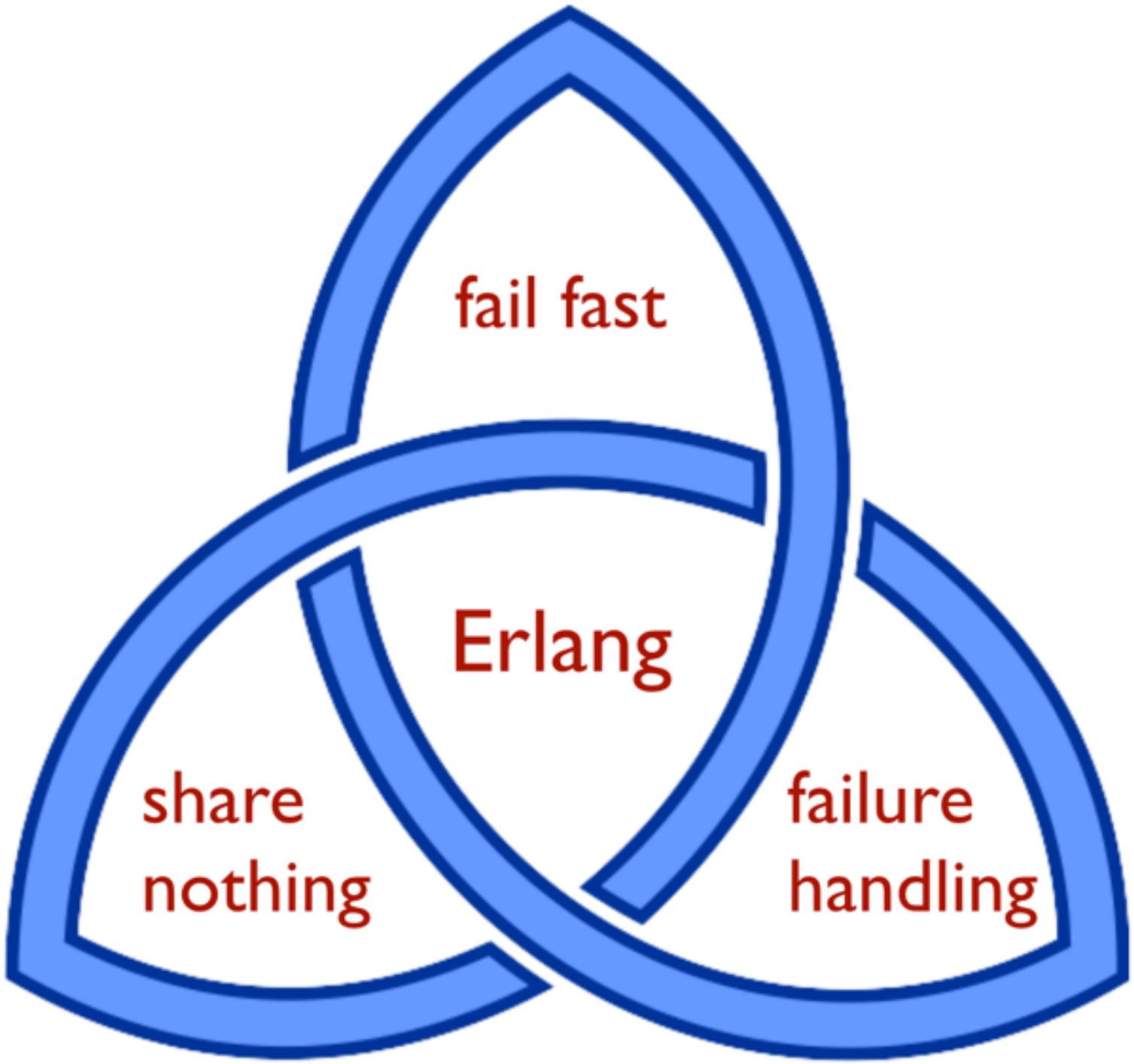
One process as controller

Timer to trigger reading sensors

Change the motor settings

No leverage of processes

The Golden Trinity Of Erlang



Sensor Proxies

Process as proxy for each sensor

Reading timer inside that process

Send message when change in value

Color Sensor

Read the color

```
def handle_info(:check, state) do
    color = EV3.ColorSensor.get_color(set_mode: false)
    new_history = Enum.take([color | state.history], @threshold)
    all_same = Enum.all?(new_history, fn c -> c == color end)
    new_state = case all_same and color != state.cur_col do
      true ->
        LineFollower.notify(state.controller, {:color, color})
        %{state | history: new_history, cur_col: color}
      false ->
        %{state | history: new_history}
  end
  :timer.send_after(@interval, :check)
  {:noreply, new_state}
end
```

Tell controller

Repeat

Bumper

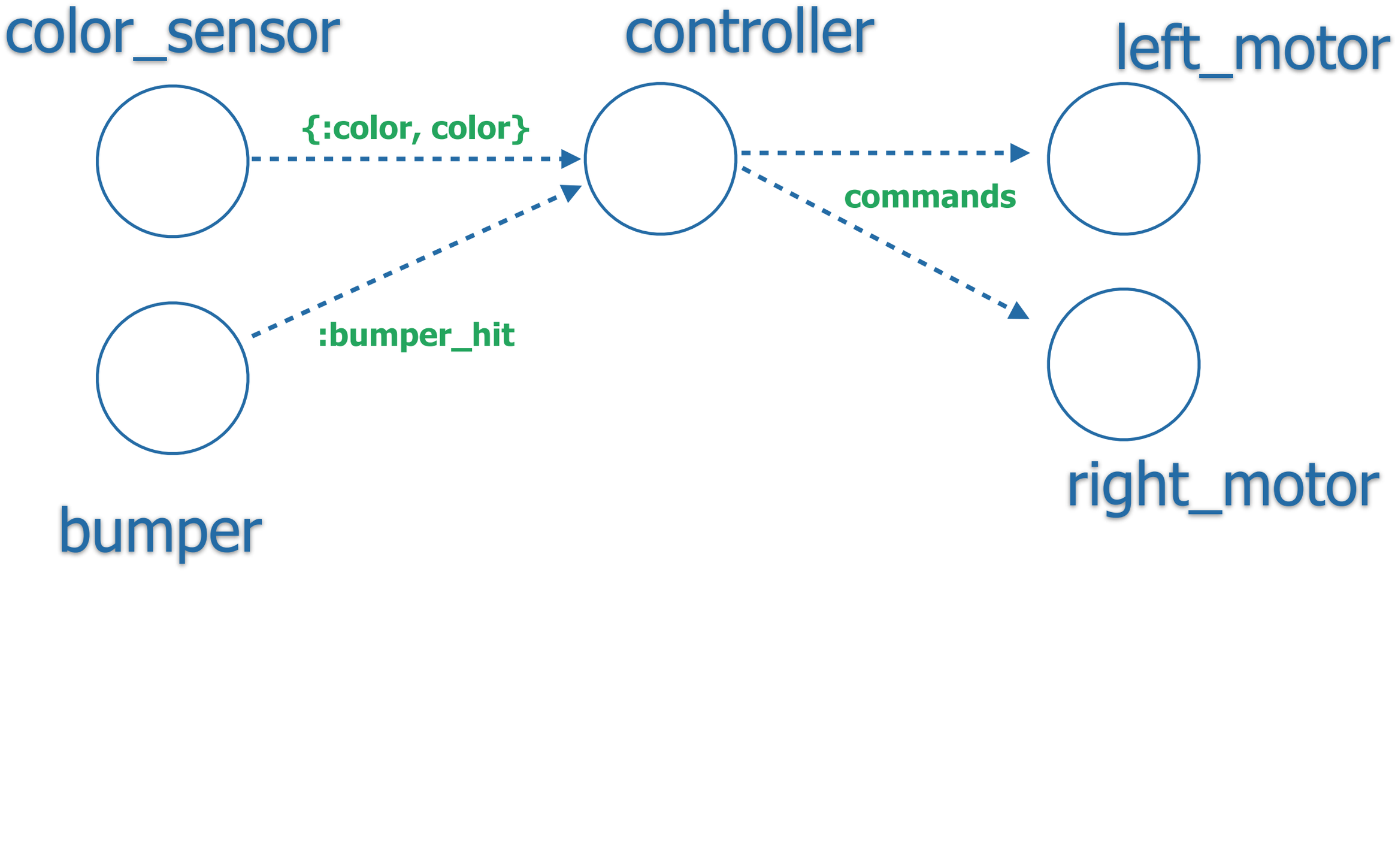
Read sensor

```
def handle_info(:check, state) do
    if EV3.TouchSensor.value == :pressed do
      LineFollower.notify(state.controller, :bumper_hit)
    end
    :timer.send_after(@interval, :check)
    {:noreply, state}
end
```

Tell controller

Repeat

Architecture



Line Follower

- Follows the left edge of the line
- Implemented as a gen_fsm
- Many small functions to control the motors

State Machine

```
def on_left_side(%State{lost_timeout: lost_timeout} = state_data) do
    forward_slight_right(state_data)
    {:next_state, :on_left_side, state_data, lost_timeout}
  end

def forward_slight_right(%State{forward_speed: forward_speed,
                                slight_turn_factor: slight_turn_factor}) do
    EV3.Motor.forward(@left_motor, forward_speed)
    EV3.Motor.forward(@right_motor, round(forward_speed * slight_turn_factor))
  end

def on_left_side({:color, target},
                 %State{target_c: target} = state_data) do
    on_right_side(state_data)
end

def on_left_side(:timeout, state_data) do
    stop_when_lost()
    {:stop, :normal, state_data}
end
```

All State Events

```
def handle_event(event, state_name, state_data) do
    Logger.info("Received all states event '#{event}'" <>
                " when in state '#{state_name}'")
    case event do
      :bumper_hit ->
        stop_motors_and_event_generators()
        {:stop, :normal, state_data}
      :stop ->
        stop_motors_and_event_generators()
        {:stop, :normal, state_data}
    end
end
```

Niceties

Protection of hardware in the process

…can be added, if needed

Reaction to the bumper is easy

Smoothening of sensor readings

A more direct approach

Fuddling around with Linux…

no comments

Main goal: write some control software

What if…

we could run the control software from the laptop?

Direct Commands

Normal software on the EV3

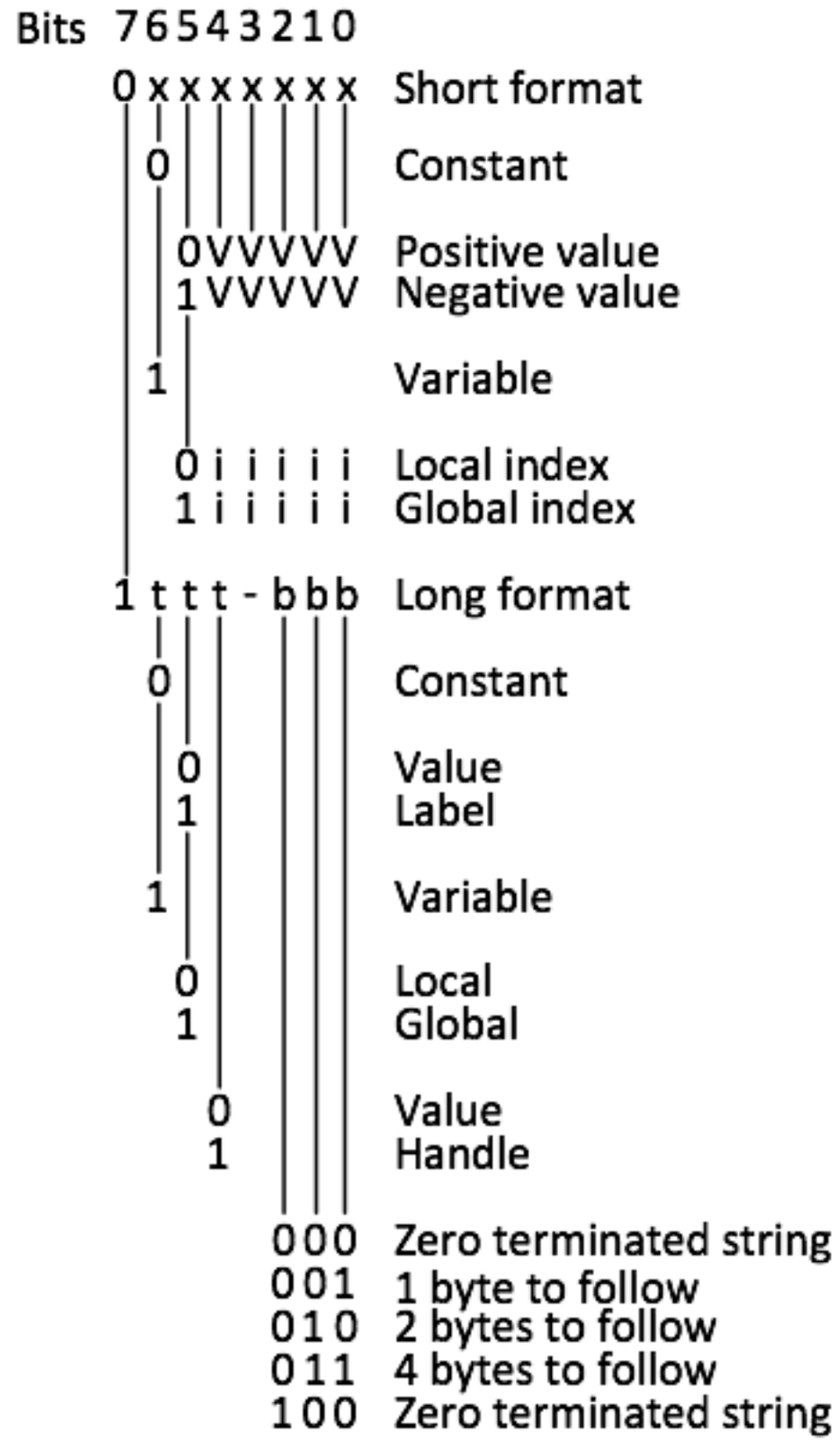
Send commands over Bluetooth/WiFi

Program runs on your own machine

Instruction	**opOutput_Stop (LAYER, NOS, BRAKE)**
Opcode	0xA3
Arguments	(Data8) LAYER – Specify chain layer number [0 - 3]
	(Data8) NOS – Output bit field [0x00 – 0x0F]
	(Data8) BRAKE – Specify break level [0: Float, 1: Break]
Dispatch status	Unchanged
Description	This function enables restting the tacho count for the individual output ports

Command Encoding

opAdd8 (ParCode1, ParCode2, ParCodeN)



Command Encoding

```
def encode(cmd_type, cmd, opts \\ @encode_defaults) do
    options       = Keyword.merge(@encode_defaults, opts)
    msg_counter   = options[:msg_counter]
    alloc_local   = options[:alloc_local]
    alloc_global = options[:alloc_global]

  << byte_size(cmd) + 5      :: size(16)-little,        # Byte 0 & 1
     msg_counter             :: size(16)-little,        # Byte 2 & 3
     @cmd_types[cmd_type]    :: size(8),                # Byte 4
     rem(alloc_global, 256) :: size(8),                 # Byte 5
     alloc_local             :: size(6),   # Bits 7-2 of byte 6
     div(alloc_global, 256) :: size(2),   # Bits 1-0 of byte 6
     cmd                     :: binary >>               # Byte 7 - n
end
```

Parameter Encoding

```
def lc(x) when abs(x) < 32,     do: lc0(x)
def lc(x) when abs(x) < 128,    do: lc1(x)
def lc(x) when abs(x) < 32768,  do: lc2(x)
def lc(x),                      do: lc4(x)

def lc0(x) do
  << @short_format          :: size(1),
     @constant              :: size(1),
     x                      :: size(6) >>
end

def lc1(x) do
  << @long_format           :: size(1),
     @constant              :: size(1),
     @constant_value        :: size(1),
     0                      :: size(2),
     FollowType.one_byte    :: size(3),
     x                      :: size(8) >>
end
```

Bluetooth Issues

Latency

…grows when sending too much :-(

Might work over WiFi

EV3 Library

Jimmy Zöger

Use with ev3dev:

https://github.com/lehoff/ev3

Direct commands:

https://github.com/lehoff/ev3bt

Warning: does not work with the current version of ev3dev

He's da man!

Teaching Programming

EV3 visual environment:

very difficult

EV3 lib for Elixir:

better, but you have to learn Elixir first

Junior Beamster

Teaching Elixir to 13-15 year olds

All material available for free

beamster.org/jb

Junior Beamster Initial Learnings

Using the command line is alien

Using an editor is a different experience

Functions are easy

Recursion comes along slowly

Need to do some UI

New Visual Language

What if…

You could start in a nice visual environment

Gradually open the boxes

Stay with the same concepts

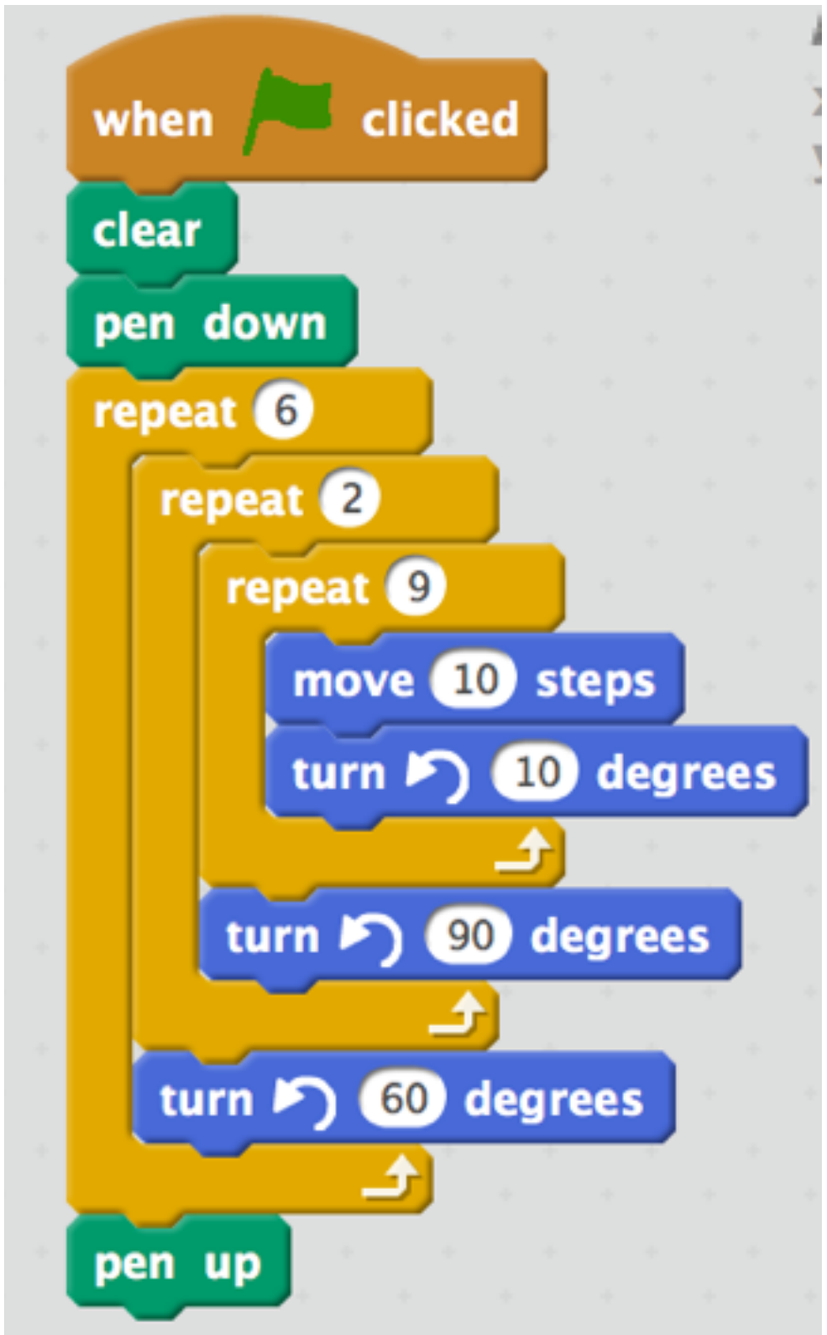
Eventually code everything in text

Scratch

Nice environment

Intuitive blocks

for sequential programming



Elixir

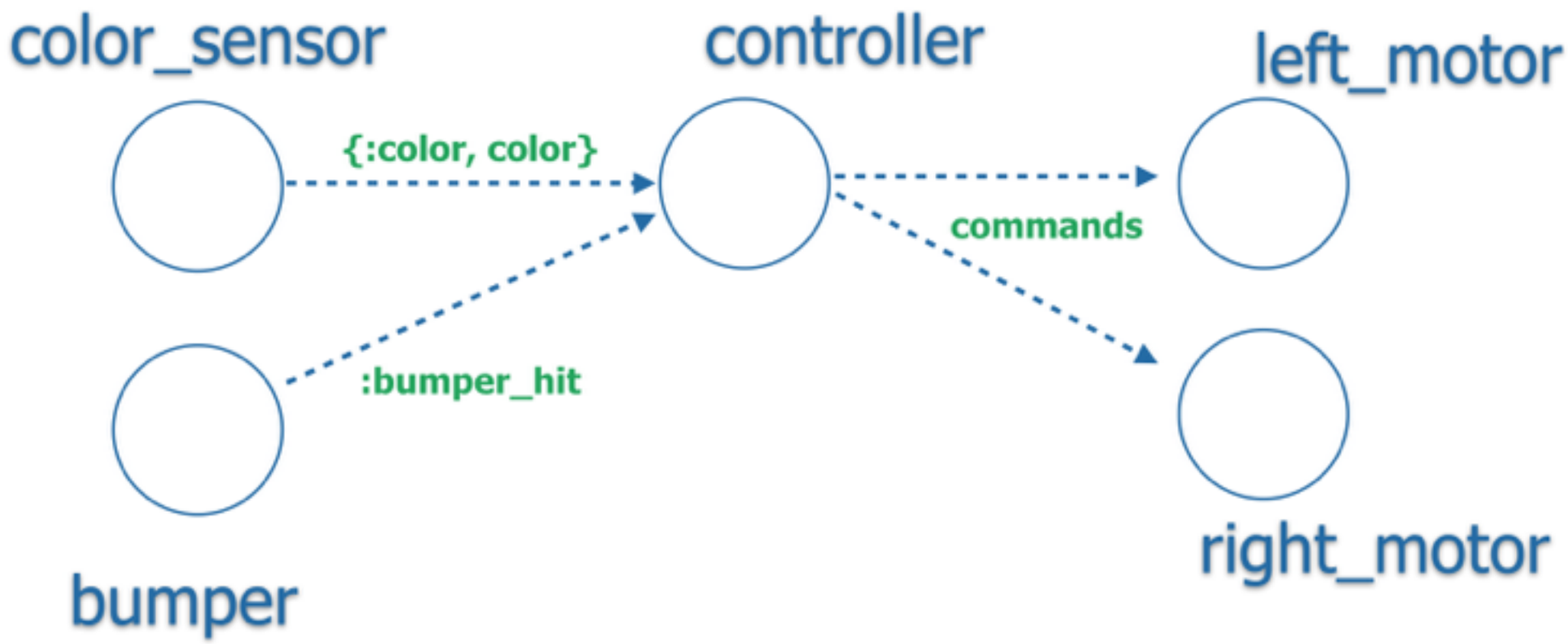
Good concepts for robots

- Processes and message passing

Visual Elements

Each sensor/actuator is just a process

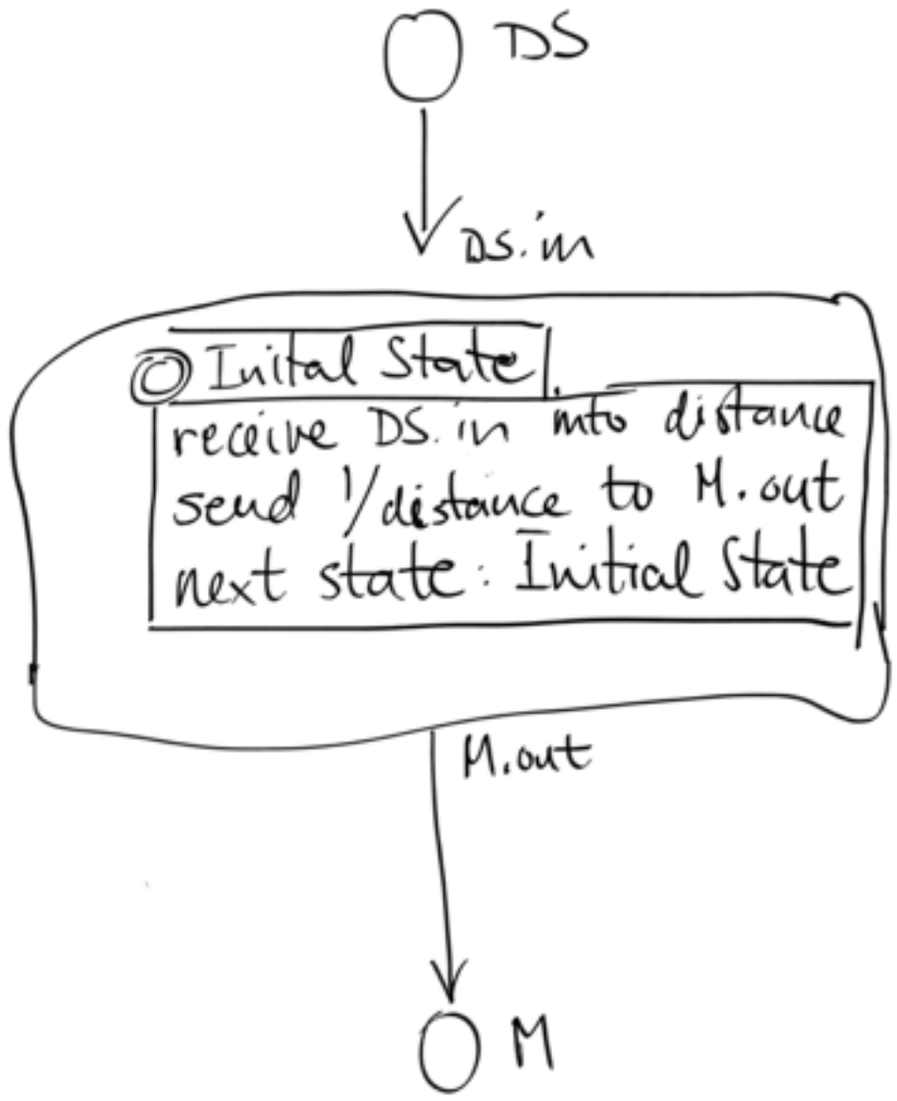
Wires = communication



Going Deeper

All elements come with a default behaviour

You can open up a box and see the code



Learning Curve

Very simple visual environment

=> you hit the ceiling fast

Complex visual environment

=> learning curve is insane

Same concepts from start to end

=> smooth learning experience

Thinking

Basic concepts

- Processes
- Message passing

Modelling is straightforward (mostly)