CSCI 1108
Introduction to Experimental Robotics

Robotics Overview

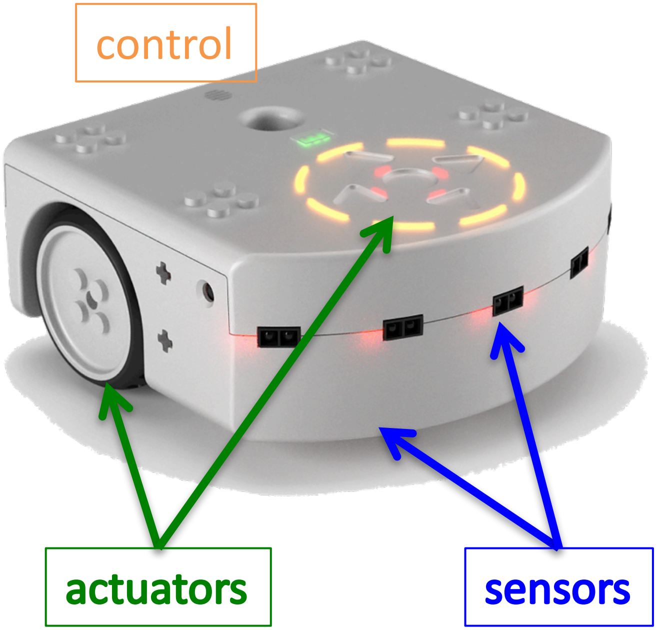
What is a Robotics

- "Robotics is the science of perceiving and manipulating the physical world through computer-controlled devices"
  Probabilistic Robotics
  S. Thrun, W. Burgard, and D. Fox
  MIT press 2006

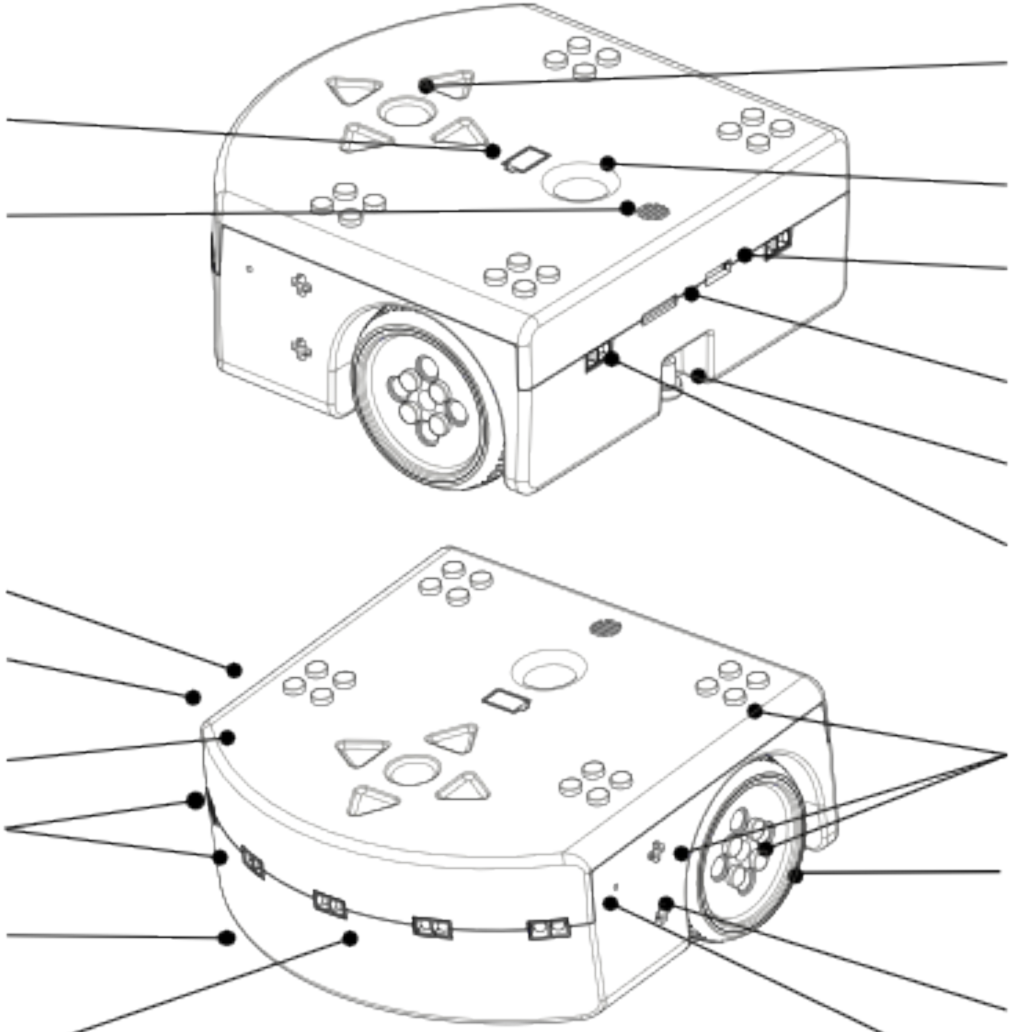
- Word robot first used in 1920
  play R.U.R. by the Czech writer Karel Čapek

Anatomy of a Robot

- Thymio II robot
  - https://aseba.wikidot.com
- Components:
  - Sensors
  - Controller
  - Actuators



Sensors and Actuators



Li-Po battery level

loudspeaker

5 capacitive touch buttons
activity display
and ON-OFF function

pencil support

USB connection
programming and recharging

memory card slot

hook for trailer

2 proximity sensors

microphone

infrared remote
control receiver

3-axis accelerometer

5 proximity sensors
obstacle detection

2 ground sensors
line following

39 LED
visualize sensors
and interactions

mechanic attachment

2 wheels
speed control

temperature sensor

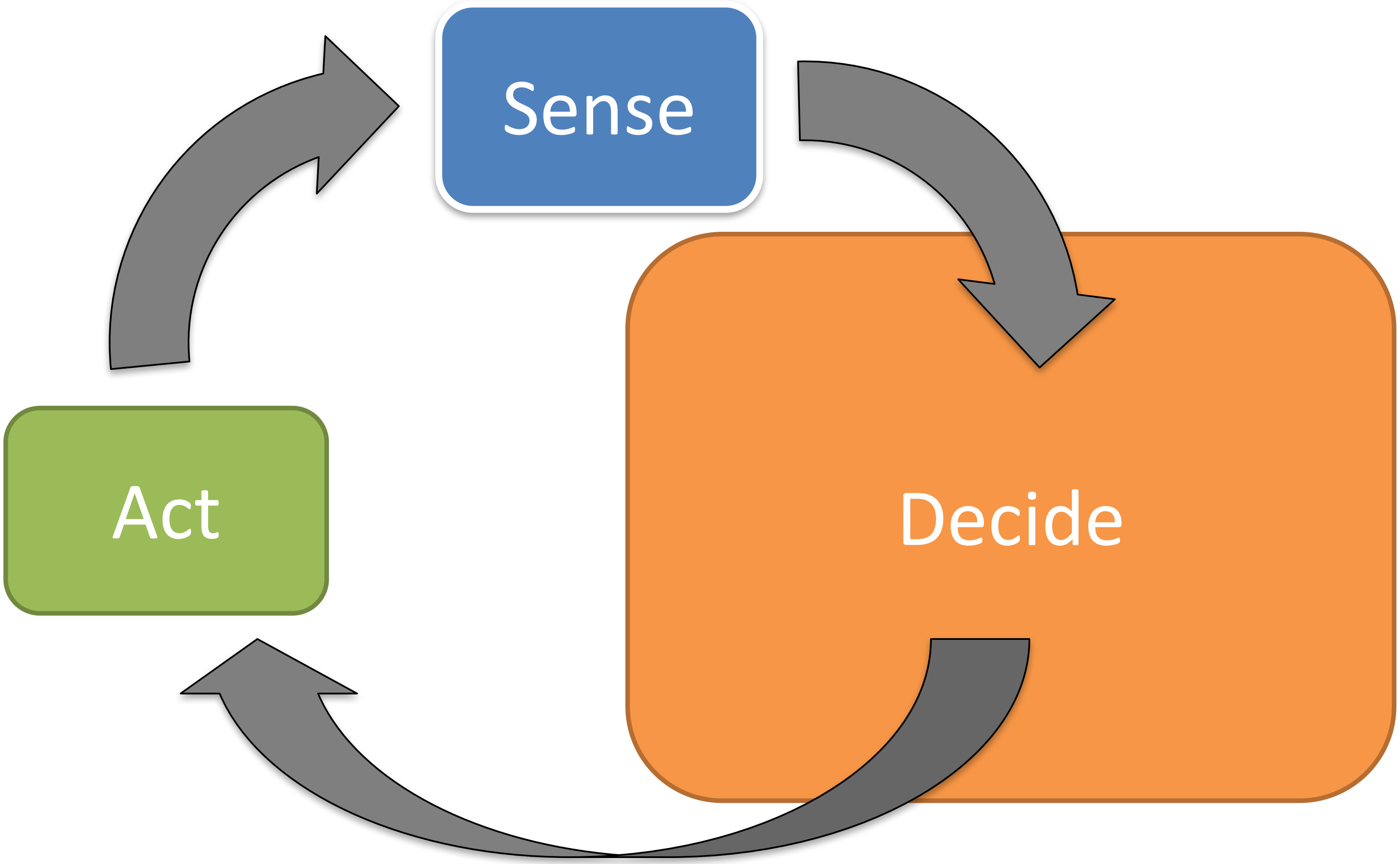
reset button

actuators

sensors

support

The Sense-Decide-Act Framework


Sense

Decide

Act

Controller:

A controller decides what action to take based on input from sensors. Our task is to write a control program for the Thymio II.

This is done in a special programming language called ASEBA

Classic Robotics themes

Actuators and movements:

Kinematics –basic movement geometry

Differential movements - change in position (Jacobian)

Dynamics –movement mechanics with forces

Sensor and object recognition:

Computer Vision

Localization:

Bayes (Kalman) filtering, SLAM, etc

Motion planning

A*, tangent bug, obstacle avoidance, etc

Other robotics terms we will be using

Pose:

Describes the configuration (e.g. position and direction) of a robot

Model:

A (simplified) description of system

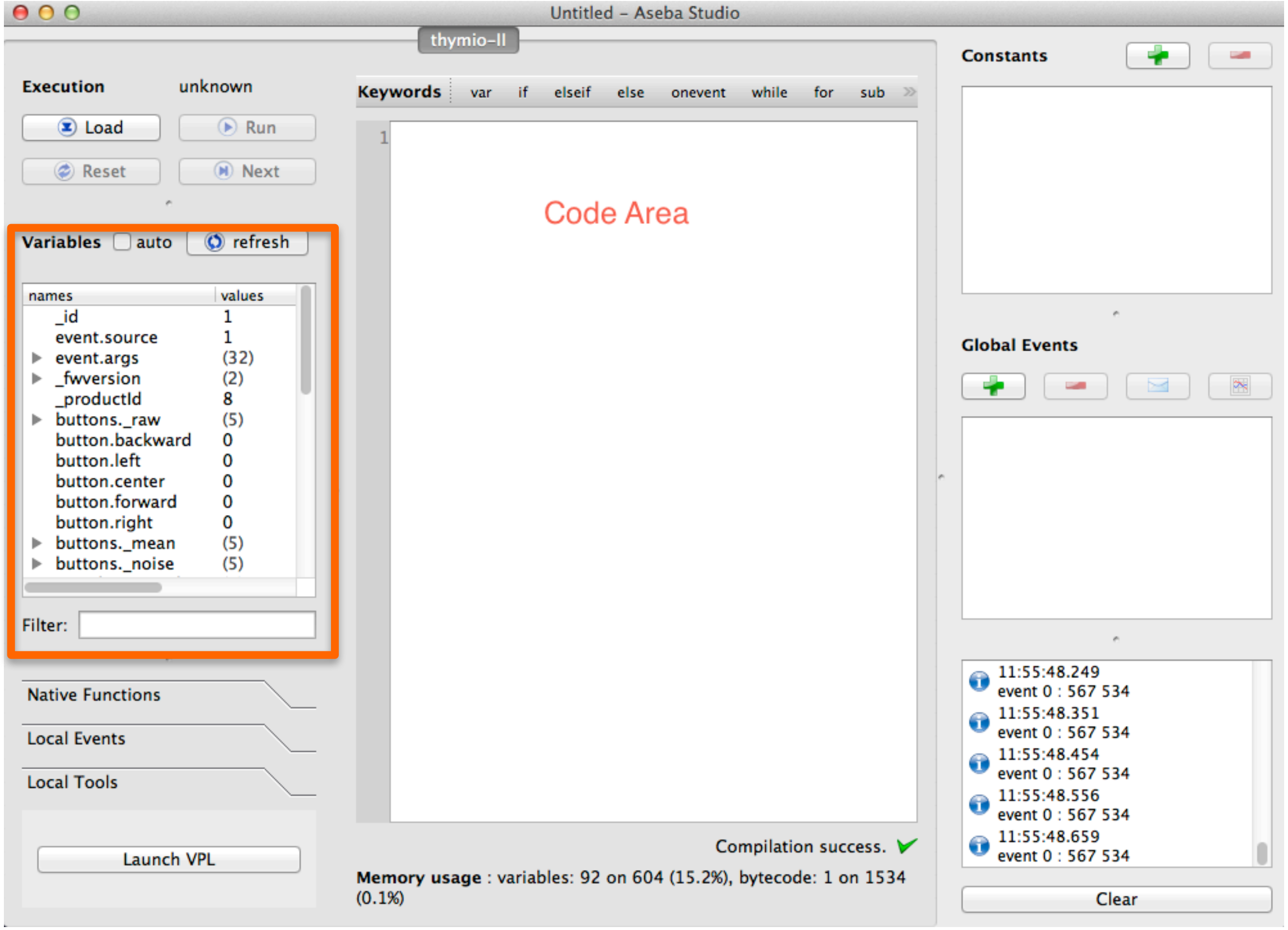
We will specifically study **Sensor models** & **Motion models**

Autonomous:

Acting independently

(as opposed to a ROVER: Remote Operated Vehicle)

Aseba Studio



names	values
_id	1
event.source	1
▶ event.args	(32)
▶ _fwversion	(2)
_productId	8
▶ buttons._raw	(5)
button.backward	0
button.left	0
button.center	0
button.forward	0
button.right	0
▶ buttons._mean	(5)
▶ buttons._noise	(5)

Compilation success. ✓

Memory usage : variables: 92 on 604 (15.2%), bytecode: 1 on 1534 (0.1%)

```
11:55:48.249
event 0 : 567 534
11:55:48.351
event 0 : 567 534
11:55:48.454
event 0 : 567 534
11:55:48.556
event 0 : 567 534
11:55:48.659
event 0 : 567 534
```

https://aseba.wikidot.com/en:thymioapi

Programming in Aseba

- Programs are text-based
- Programing environment called Aseba Studio
- Key Ideas: Event-based programming
  - Events are triggered by sensors
  - Events are handled by event handlers for which we write the code: **`onevent`** …
  - Common programing model for interactive programs (e.g. www, computer interface, etc)

A Sample Program

```
var speed = 100


motor.left.target = 0
motor.right.target = 0


onevent button.forward
  motor.left.target = speed
  motor.right.target = speed

onevent button.backward
  motor.left.target = 0
  motor.right.target = 0

onevent button.left
  motor.left.target = -speed
  motor.right.target = speed

onevent button.right
  motor.left.target = speed
  motor.right.target = -speed
```

Key Idea: Actuators are controlled by setting variables that represent them

The Four Parts of an Aseba Program

- Variable declarations
  - Begin with the **`var`** keyword
- Initialization code
  - Anything except declarations
- Event handlers
  - Begin with the **`onevent`** keyword
- Subroutines
  - Begin with the **`sub`** keyword

Basic Aseba

- Variables

```
var name
var list[]
```

- Event Handler

```
onevent prox
```

- Conditional

```
if      then

end
```

Sensors and Actuators in Aseba

- Key Idea: All sensors and actuators are accessed via predefined variables, e.g.,
  - to control motors, assign values to motor variables

    ```
    motor.left.target = 100
    motor.right.target = 100
    ```

  - to check if an object is close, read proximity variable

    ```
    if prox.horizontal[2] > 1000 then
      ...
    end
    ```

- Where are all the predefined variables listed?
- When do we check variables?

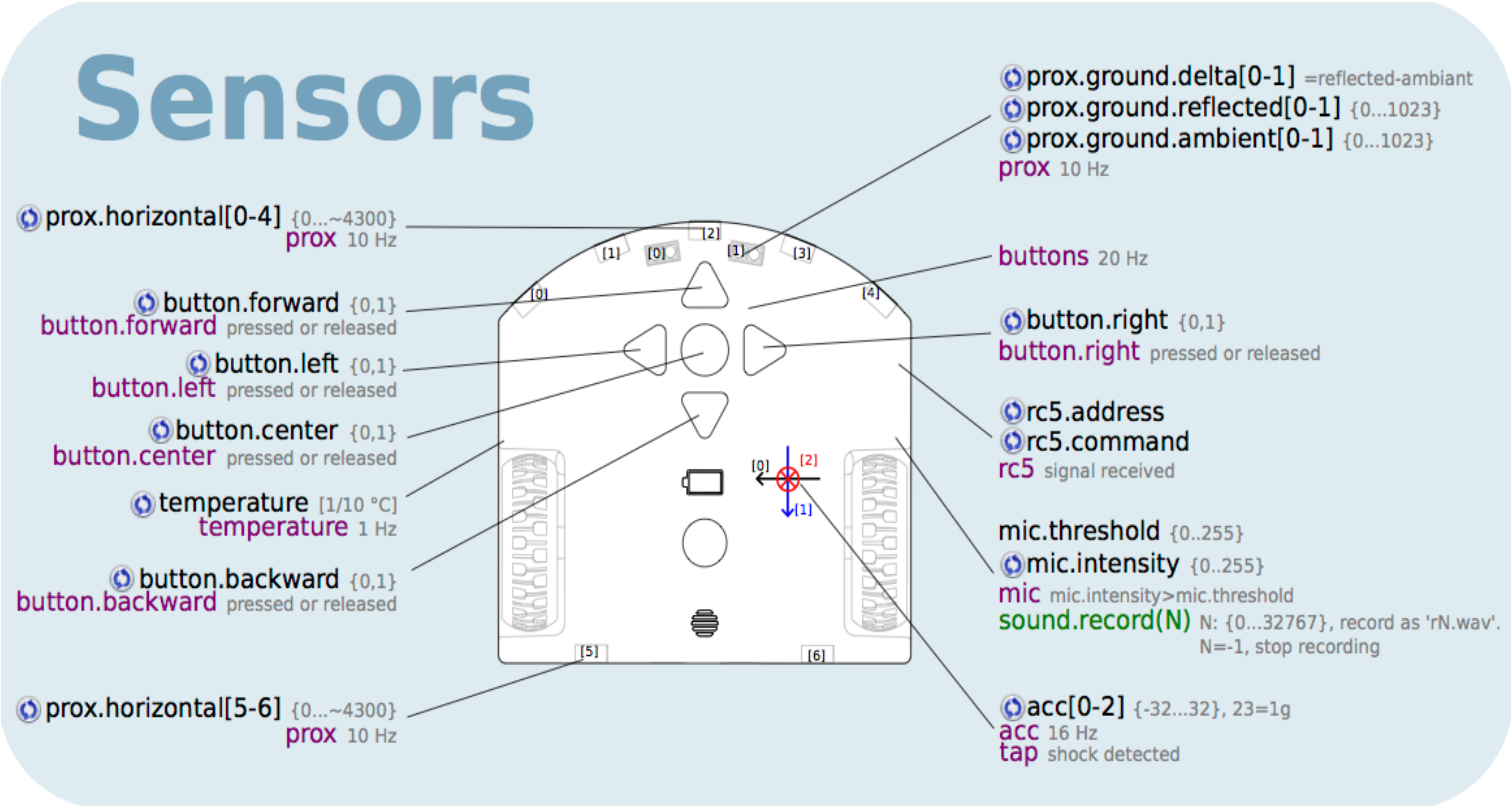
When do We Check the Sensors?

- Key Idea: Sensors generate events. Event handlers check sensors

- Example: Proximity (**prox**) sensors generate 10 events per second

```
onevent prox
  if prox.horizontal[2] > 1000 then
    motor.left.target = 0
    motor.right.target = 0
  else
    motor.left.target = 100
    motor.right.target = 100
  end
```

Sensors



Sensors

prox.horizontal[0-4] {0...~4300}
prox 10 Hz

button.forward {0,1}
button.forward pressed or released

button.left {0,1}
button.left pressed or released

button.center {0,1}
button.center pressed or released

temperature [1/10 °C]
temperature 1 Hz

button.backward {0,1}
button.backward pressed or released

prox.horizontal[5-6] {0...~4300}
prox 10 Hz

prox.ground.delta[0-1] =reflected-ambiant
prox.ground.reflected[0-1] {0...1023}
prox.ground.ambient[0-1] {0...1023}
prox 10 Hz

buttons 20 Hz

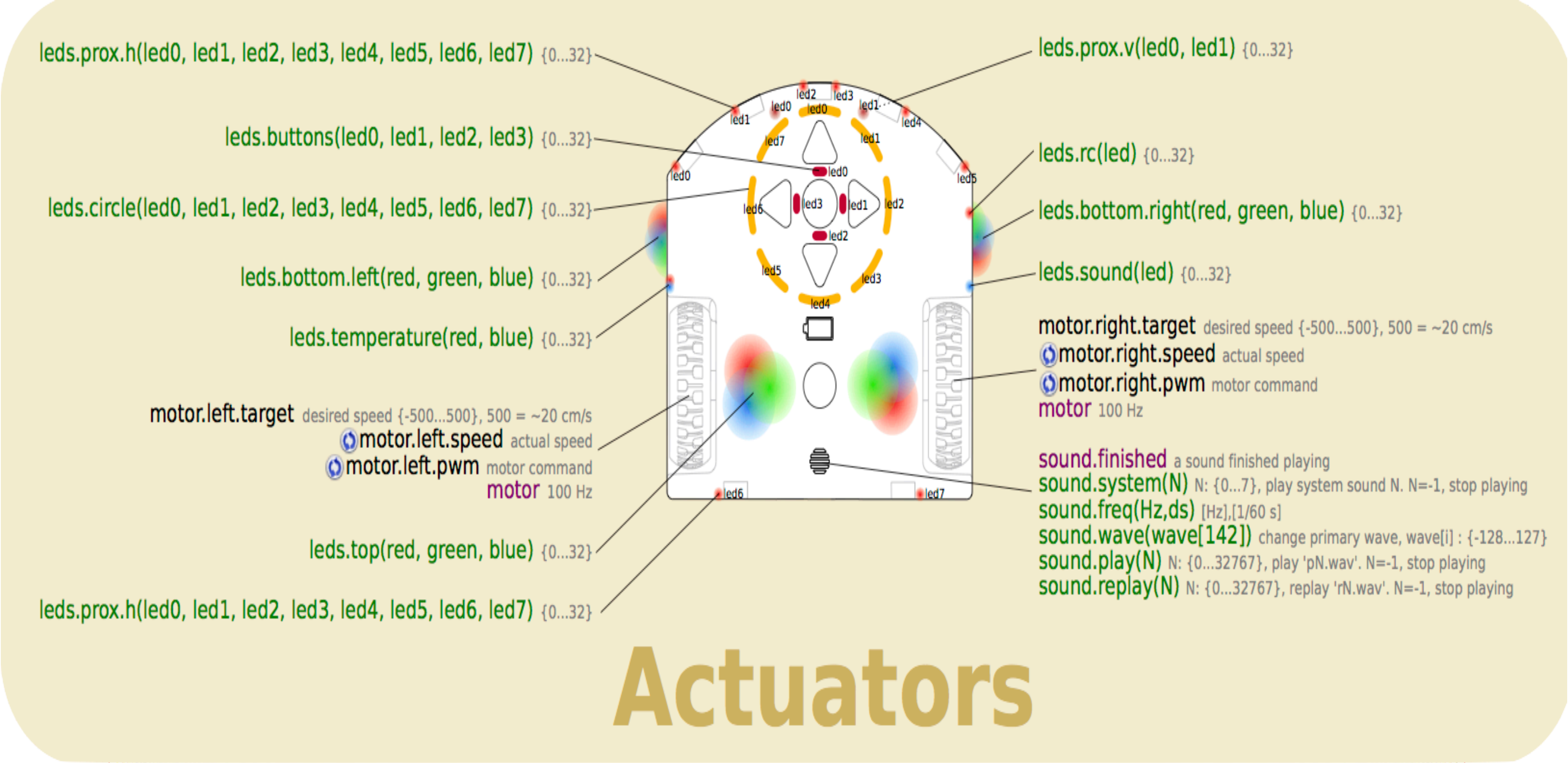
button.right {0,1}
button.right pressed or released

rc5.address
rc5.command
rc5 signal received

mic.threshold {0..255}
mic.intensity {0..255}
mic mic.intensity>mic.threshold
sound.record(N) N: {0...32767}, record as 'rN.wav'.
N=-1, stop recording

acc[0-2] {-32...32}, 23=1g
acc 16 Hz
tap shock detected

Actuators



leds.prox.h(led0, led1, led2, led3, led4, led5, led6, led7) {0...32}

leds.buttons(led0, led1, led2, led3) {0...32}

leds.circle(led0, led1, led2, led3, led4, led5, led6, led7) {0...32}

leds.bottom.left(red, green, blue) {0...32}

leds.temperature(red, blue) {0...32}

motor.left.target desired speed {-500...500}, 500 = ~20 cm/s
motor.left.speed actual speed
motor.left.pwm motor command
motor 100 Hz

leds.top(red, green, blue) {0...32}

leds.prox.h(led0, led1, led2, led3, led4, led5, led6, led7) {0...32}

leds.prox.v(led0, led1) {0...32}

leds.rc(led) {0...32}

leds.bottom.right(red, green, blue) {0...32}

leds.sound(led) {0...32}

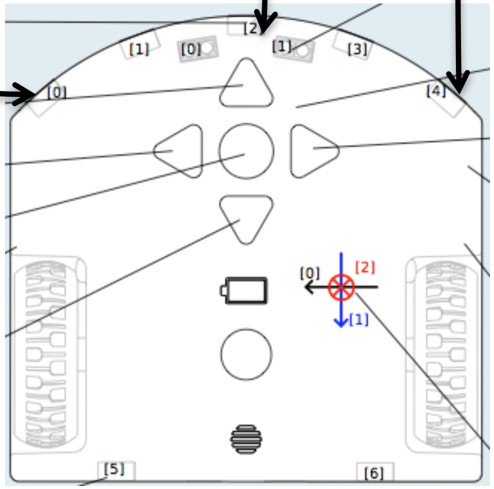
motor.right.target desired speed {-500...500}, 500 = ~20 cm/s
motor.right.speed actual speed
motor.right.pwm motor command
motor 100 Hz

sound.finished a sound finished playing
sound.system(N) N: {0...7}, play system sound N. N=-1, stop playing
sound.freq(Hz,ds) [Hz],[1/60 s]
sound.wave(wave[142]) change primary wave, wave[i] : {-128...127}
sound.play(N) N: {0...32767}, play 'pN.wav'. N=-1, stop playing
sound.replay(N) N: {0...32767}, replay 'rN.wav'. N=-1, stop playing

Actuators

Last Example

```
onevent prox
  if prox.horizontal[2] > 1000 then
    motor.left.target = 0
    motor.right.target = 0
  elseif prox.horizontal[4] > 1000 then
    motor.left.target = -100
    motor.right.target = 100
  elseif prox.horizontal[0] > 1000 then
    motor.left.target = 100
    motor.right.target = -100
  else
    motor.left.target = 100
    motor.right.target = 100
  end
```


[0]
[1]
[0]
[2]
[1]
[3]
[4]
[0]
[2]
[1]
[5]
[6]