http://pharo.org

http://rmod.inria.fr

PharoThings

The live programming IoT platform

Denis Kudriashov dionisiydk@gmail.com

https://github.com/pharo-iot/PharoThings

What is PharoThings?

- A Pharo image **running** on IoT device
    - Raspberry Pi
    - Beaglebone
    - other small machines

*Follow github page for installation https://github.com/pharo-iot/PharoThings

Pharo on ARM machines

- ArmVM: http://files.pharo.org/vm/pharo-spur32/linux/armv6/latest.zip
    - JIT
    - FFI
    - OSProcess/OSSubprocess
        - https://github.com/marianopeck/OSSubprocess

Raspberry GPIO libraries

- WiringPI bindings from Jean Baptiste
  - used by PharoThings
  - https://github.com/pharo-iot/WiringPi
    - docs http://wiringpi.com
- Pigpio bindings from Tim Rowledge
  - http://www.squeaksource.com/HardwarePeripherals.html
  - docs http://abyz.co.uk/rpi/pigpio/
  - Required old compiler to install in Pharo

What is PharoThings?

- A Pharo image running on IoT device
- A Pharo image **controlling remote** IoT device

*Follow github page for installation https://github.com/pharo-iot/PharoThings

TelePharo

- Remote access to running Pharo images
    - Remote image scripting from playground
    - Remote object inspector
    - Remote process browser
- Complete toolset for remote development
    - Remote code browser
    - Remote debugger
- https://github.com/dionisiydk/TelePharo

What is PharoThings?

- A Pharo image running on IoT device
- A Pharo image **controlling remote** IoT device

*Follow github page for installation https://github.com/pharo-iot/PharoThings

Direct access to IoT device

- Low level libraries/protocols to access devices
  - Arduino by Firmata https://github.com/pharo-iot/Firmata

*Follow github page for installation https://github.com/pharo-iot/PharoThings

What is PharoThings?

- A Pharo image running on IoT device
- A Pharo image controlling remote IoT device
- An object board model

The board model

- Pins are objects
  - gpio4 beDigitalOutput
  - gpio4 toggleDigitalValue

The board model

- Hierarchy of boards with specific configuration of pins
  - RpiBoardBRev1 with single connector P1 with 26 pins
  - RpiBoardBRev2 with two connectors P1 with 26 pins and P5 with 8 pins

The board model

- High level peripheral model to program connected devices

```
button := board installIDevice: (PotButton named: 'button green' fromPowerTo: gpio3).

button when: PotButtonReleased send: #toggleDigitalValue to: gpio4.
```

The board model

- Persistent by default
- Save the image: the board state will be recovered after restart
  - pins are in same state
  - connected devices continue working

What is PharoThings?

- A Pharo image running on IoT device
- A Pharo image controlling remote IoT device
- An object board model
- The advanced board model inspector

The board model inspector

- Pins schema like in docs
- live pin state
- interactive

The board model inspector

- Evaluation pane with gpio bindings
- script pins by #doIt/#printIt

Inspector on a PotRemoteBoard (a RpiBoardBRev1 in #[169

a PotRemoteBoard (a RpiBoardBRev1 in #[169 254 0 2]:40423)

P1 | Devices | Raw | Meta

Id	Value	Name	Pin#	Pin#	Name	Value	Id
		3.3v	1	2	5v		
0		SDA (I2C)	3	4	5v		
1		SCL (I2C)	5	6	Ground (0v)		
4		GPIO7	7	8	SerialPortTXD		14
		Ground (0v)	9	10	SerialPortRXD		15
17		GPIO0	11	12	GPIO1		18
21		GPIO2	13	14	Ground (0v)		
22	in	GPIO3	15	16	GPIO4	out	23
		3.3v	17	18	GPIO5		24
10		MOSI (SPI)	19	20	Ground (0v)		
9		MISO (SPI)	21	22	GPIO6		25
11		SCLK (SPI)	23	24	CE (SPI)		8
		Ground (0v)	25	26	CE (SPI)		7

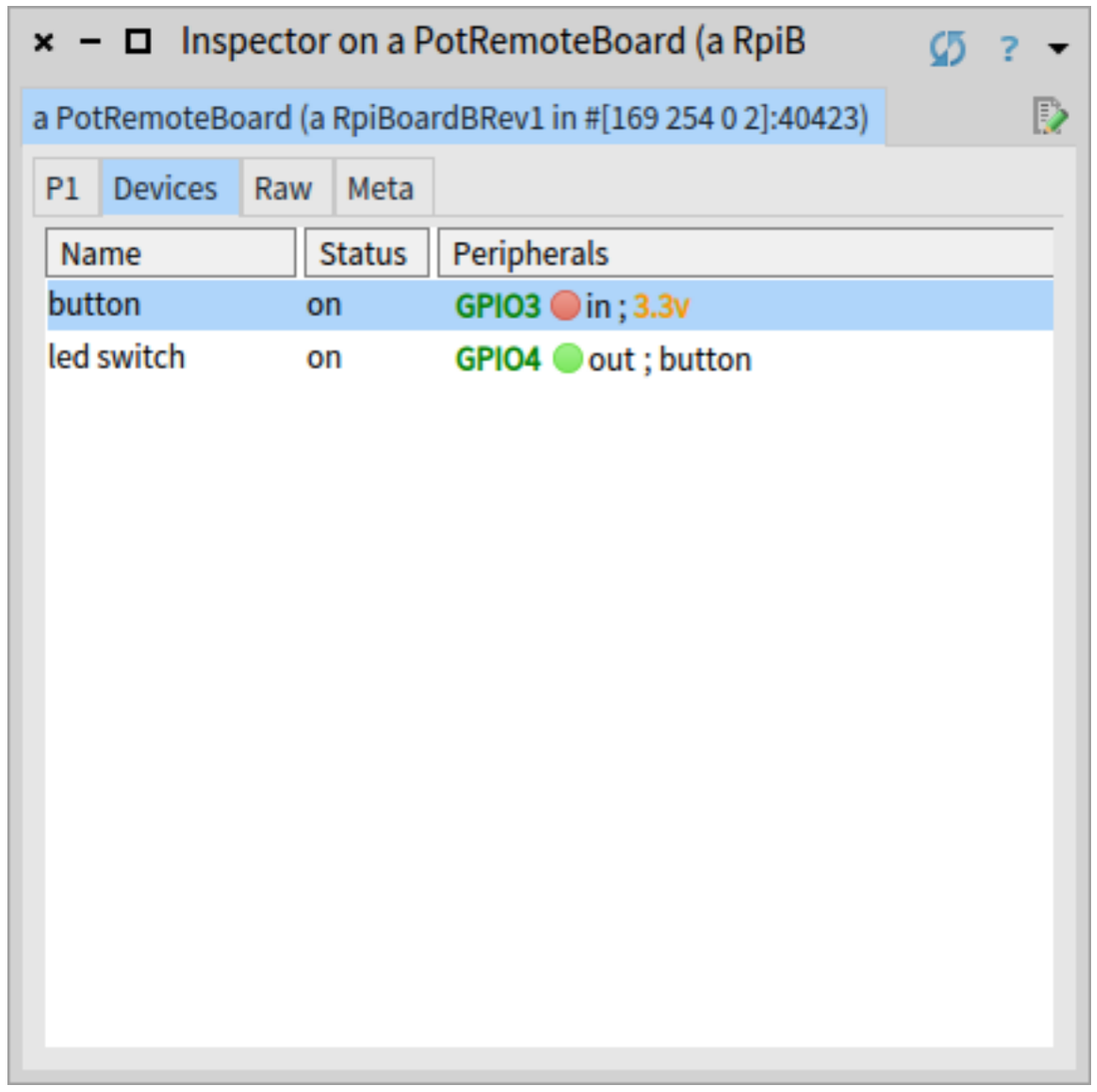
```
"a PotBoardConnector(P1): gpio0..gpio7 vars are bound to pins"
led := gpio4.
led beDigitalOutput.
led value: 1.
led value: 0.

button := gpio3.
button beDigitalInput. "button"
button enablePullDownResister.
button value.

buttonProcess := [ [100 milliSeconds wait.
   led value: (button value=1) asBit
      ] repeat
    ] fork.
```

The board model inspector

- Extra device management tab

```
button := board installDevice: (PotButton named: 'button green' fromPowerTo: gpio3).
board installDevice: (PotSwitch named: 'led switch' for: gpio4 using: button).
```

Device management tab



The board model inspector

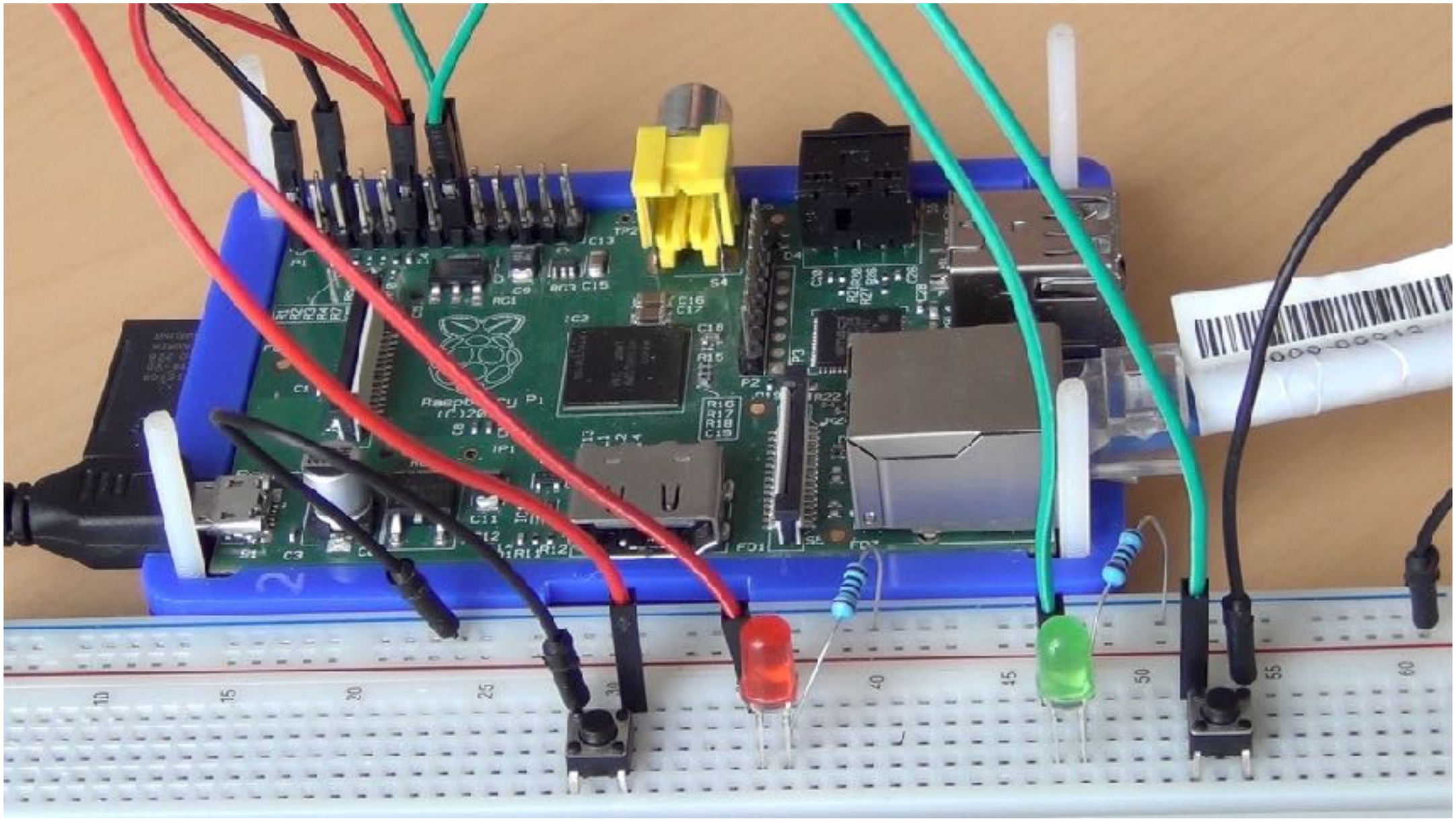
- Works remotely

```
remoteBoard := remotePharo evaluate: [ RpiBoardBRev1 current].
remoteBoard inspect
```

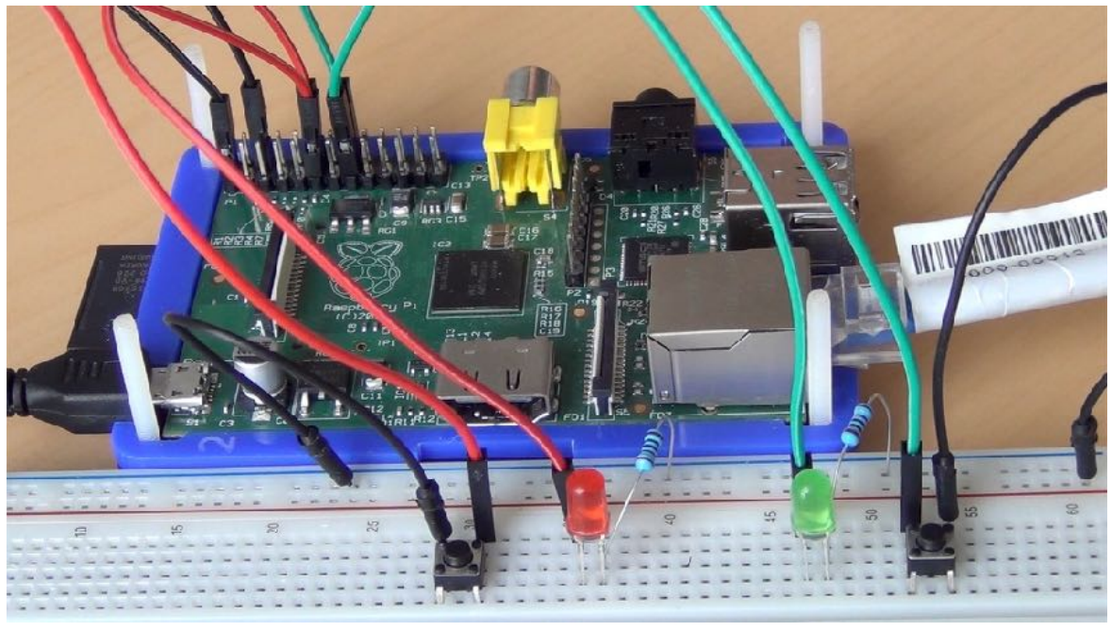
PharoThings

- A Pharo image **running** on IoT device
- A Pharo image **controlling remote** IoT device
- An object board model
- The advanced board model inspector

Demo



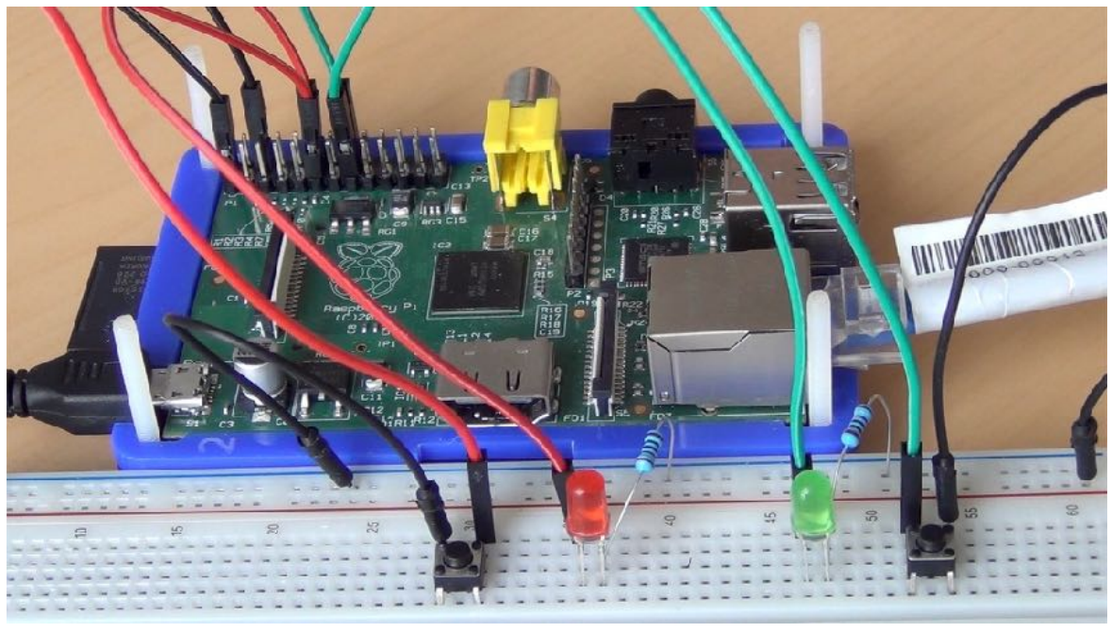
Demo



- Headless Raspberry Pi
- Button connected by red wire from the ground to gpio0
- Red led connected to gpio1
- Green led connected to gpio4
- Button connected by green wire from the power to gpio3

*Follow peripherals on picture from the left to the right

We want



- Switch the green led by green button (green wire)
- Switch the red led by red button (red wire)

*Follow demo videos at github page https://github.com/pharo-iot/PharoThings

Few pictures of remote tools

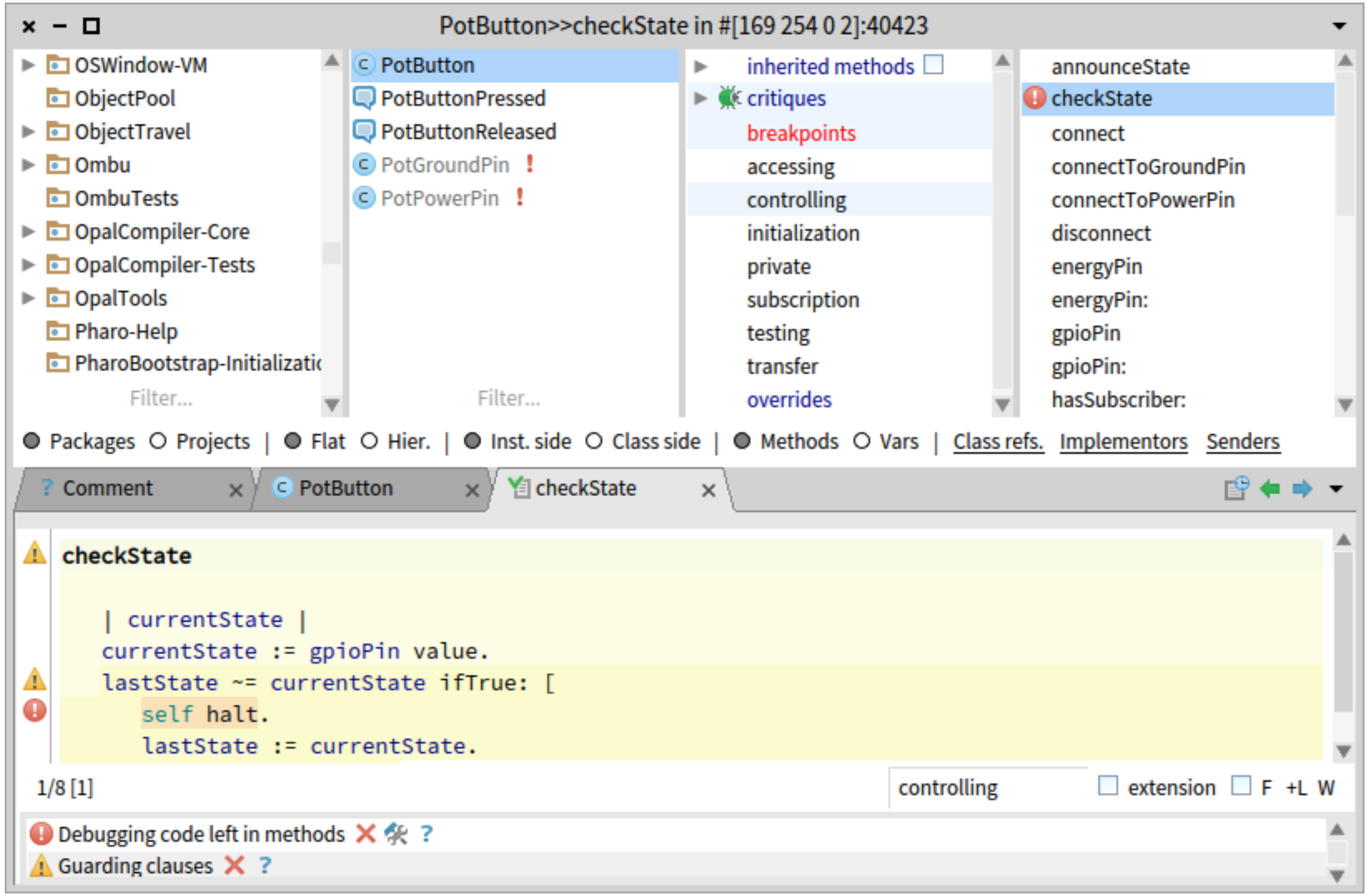
Remote playground

RPlayground#[169 254 0 2]:40423

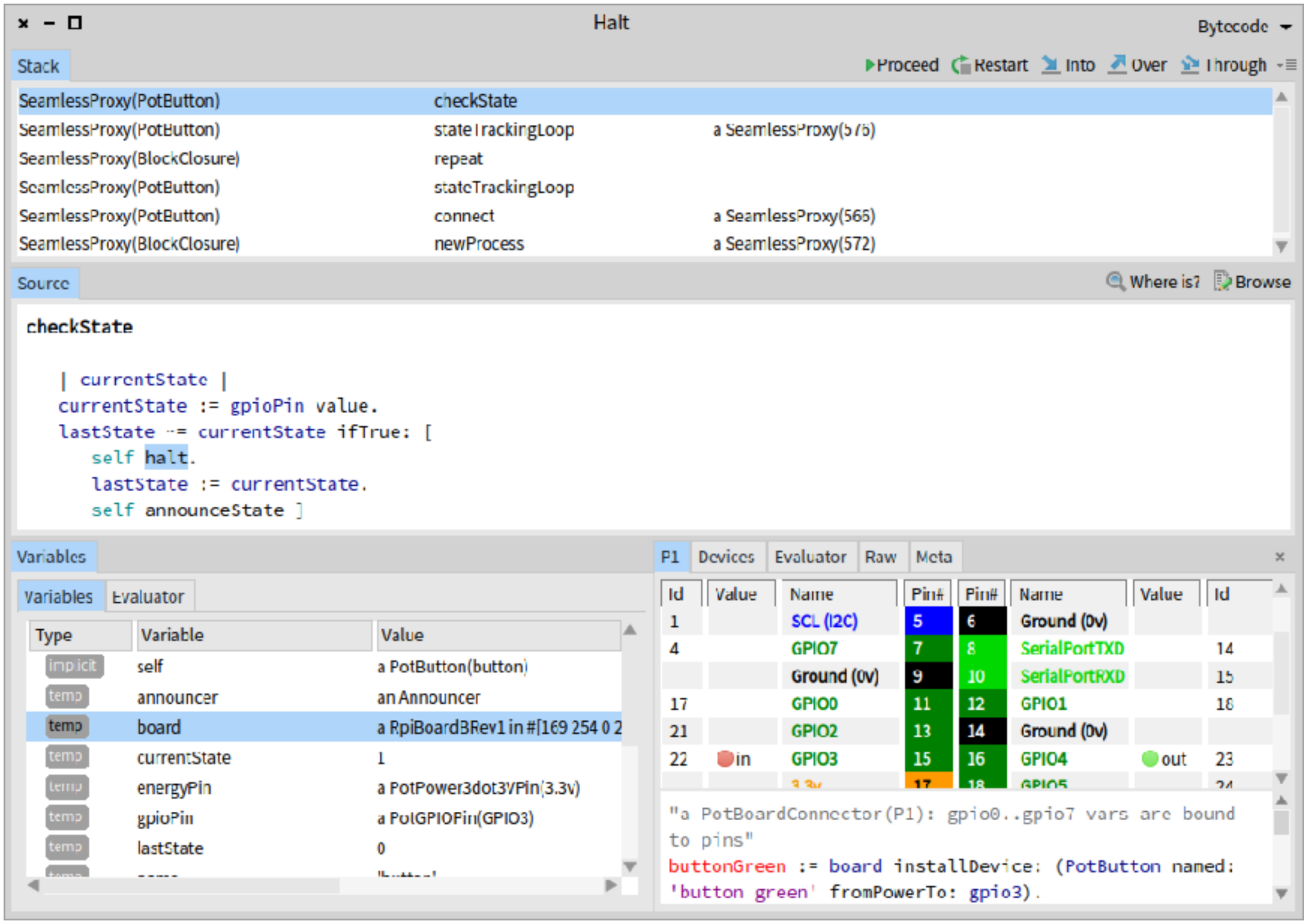
Page

```
OSSUnixSubprocess new
    command: 'uname';
    arguments: #('-a');
    redirectStdout;
    runAndWaitOnExitDo: [ :process :outString | ^ outString ].
 "'Linux raspberrypi 3.12.28+ #709 PREEMPT Mon Sep 8 15:28:00 BST 2014 armv6l GNU/Linux'"

lib := WiringPiLibrary uniqueInstance.
lib wiringPiSetupGpio.
lib pin: 18 mode: 1. "led output"
lib pin: 18 write: 1.
lib pin: 18 write: 0.
```

Remote browser



Remote debugger



Future

- More RaspberryPI models
- Beaglebone models
- Arduino models
- Deploying as service from image
- Zeroconf for armVM+PharoThings
- General evolution of TelePharo
    - Automatic detection of running images in network
    - Remote refactoring
    - Security
    - many other things

The end