# A PharoThings Tutorial

Allex Oliveira

January 8, 2019

Copyright 2017 by Allex Oliveira.

The contents of this book are protected under the Creative Commons Attribution-ShareAlike 3.0 Unported license.

You are **free**:

- to **Share**: to copy, distribute and transmit the work,
- to **Remix**: to adapt the work,

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page:
http://creativecommons.org/licenses/by-sa/3.0/

Any of the above conditions can be waived if you get permission from the copyright holder. Nothing in this license impairs or restricts the author's moral rights.

Your fair dealing and other rights are in no way affected by the above. This is a human-readable summary of the Legal Code (the full license):
http://creativecommons.org/licenses/by-sa/3.0/legalcode

Layout and typography based on the `sbabook` LaTeX class by Damien Pollet.

# Contents

**Illustrations** ii

**1 Lesson 5 - LED Flowing Lights using OOP** 1

1.1 The goal . . . 1
1.2 Creating the class . . . 2
1.3 Creating the initialize method . . . 2
1.4 Creating access methods . . . 2
1.5 Creating process control methods . . . 3
1.6 Creating action methods . . . 3

# Illustrations

CHAPTER 1

# Lesson 5 - LED Flowing Lights using OOP

Now we can play with the LEDs, turn them on, off, blink it and manipulate many at the same time. Let's use object-oriented programming, OOP to create methods and classes, to build a simple program, to control the LEDs flow like as we want.

## 1.1 The goal

In this chapter we will create a program to control the LEDs with some features:

- We want to set how many times the LED line will flow;
- We want to set the speed of this flow;
- And we want to set the direction if the flow will start from left or right and which order will be executed. For example, if we want to set to flow starting from the right and after the start from left again, we will use the message 'lrl'. The program needs to able to flow in any combination of directions, like llr, rlllrr, rlrlrlr etc.

The final code executed in the playground will be a code like this:

```
leds := Flowing new.
leds times: 2 delay: 0.1 direction: 'lrl'.
leds flowStart.
leds flowStop.
leds turnOn.
leds turnOff.
```

## 1.2 Creating the class

```
Object subclass: #Flowing
    instanceVariableNames: 'ledArray flowProcess flowDirection
     toggleDelay timesRepeat'
    classVariableNames: ''
    package: 'PharoThings-Lessons'
```

## 1.3 Creating the initialize method

```
initialize
    ledArray := {
    (PotGPIOPin id: 17 number: 0).
    (PotGPIOPin id: 18 number: 1).
    (PotGPIOPin id: 27 number: 2).
    (PotGPIOPin id: 22 number: 3).
    (PotGPIOPin id: 23 number: 4).
    (PotGPIOPin id: 24 number: 5).
    (PotGPIOPin id: 25 number: 6).
    (PotGPIOPin id: 4 number: 7)
    }.
   ledArray do: [ :item | item board: RpiBoard3B current;
    beDigitalOutput; value:0 ].
    timesRepeat := 2.
    toggleDelay := 0.5.
    flowDirection := 'lr'.
```

## 1.4 Creating access methods

```
times: anInteger delay: aNumber direction: aString
    timesRepeat := anInteger.
    toggleDelay := aNumber.
    flowDirection := aString
```

```
flowDirection
    ^flowDirection
```

```
flowTimesRepeat
    ^timesRepeat
```

```
toggleDelay
     ^toggleDelay
```

## 1.5 Creating process control methods

```
flowStart
    flowProcess := [ (self flowTimesRepeat) timesRepeat: [
          self action
      ] ] forkNamed: 'FlowingProcess'.
```

```
flowStop
    flowProcess terminate
```

## 1.6 Creating action methods

```
action
flowDirection do: [ :character | character == $l ifTrue: [ ledArray
    do: self toggleLedArray  ].
                        character == $r ifTrue: [ ledArray
    reverseDdo: self toggleLedArray  ] ]
```

```
toggleLedArray
    ^[ :item | item toggleDigitalValue. (Delay forSeconds: self
    toggleDelay) wait ]
```

```
turnOn
    ledArray do: [ :item | item value:1 ].
```

```
turnOff
    ledArray do: [ :item | item value:0 ].
```