A PharoThings Tutorial

Allex Oliveira

September 25, 2018

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

Contents

Illustrations

In this booklet we will show you how to develop a little application that collect weather information. We will start to show how we can play with leds and others.

CHAPTER 1

Installations

The first step you need to get started with PharoThings is to install an Operating System in your Raspberry Pi. When you buy a Raspberry Pi, the OS is not factory installed.

1.1 Installating OS on RASPBERRY (RASPBIAN)

In this tutorial, we will download and install NOOBS (New Out Of the Box Software). NOOBS is an easy operating system installer which contains Raspbian (https://www.raspberrypi.org/downloads/raspbian/) and LibreELEC (https://libreelec.tv).

Raspbian is the Foundation's official supported operating system, a Linux OS based on Debian Stretch to run in ARM processors.

Download

You can download an official image from the Raspberry Pi website Noobs downloads page (https://www.raspberrypi.org/downloads/noobs/). You will download a zip file and extract the files to your SD card.

Copying

You will need a computer with an SD card reader to install the image. This process basically extracts the files from the zip file downloaded into an SD card formatted and start the Raspberry Pi with this SD card.

You can go directly to your operating system by clicking on the links below:

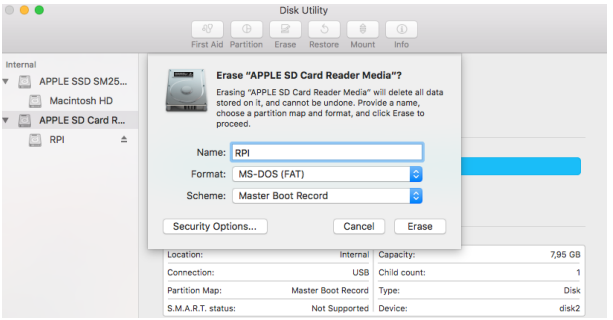

Figure 1-1 Preparing SD Card.

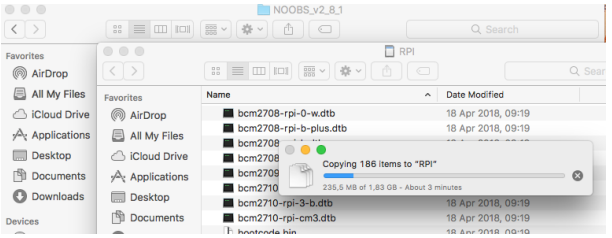

Figure 1-2 Copying NOOBS.

1.2 Copying Raspbian files on MAC OSX

- Open “disk utility”, select the SD Card and Erase (Format MS-DOS FAT) as shown in Figure 1-1.
- Copy the files from folder NOOBS_xxx to SD Card as shown in Figure 1-2.

1.3 Copying Raspbian files on Linux

1.4 Copying Raspbian files on Windows

1.5 Installing the Raspbian in Raspberry Pi

Insert the SD Card on Raspberry and turn it on. Select “Raspbian” and “Yes” as shown by Figure 1-3.

In a few minutes, you will have your Raspberry Pi running Raspbian OS. Now you can install PharoThings and control devices remotely.

Figure 1-3 Installing Raspbian.

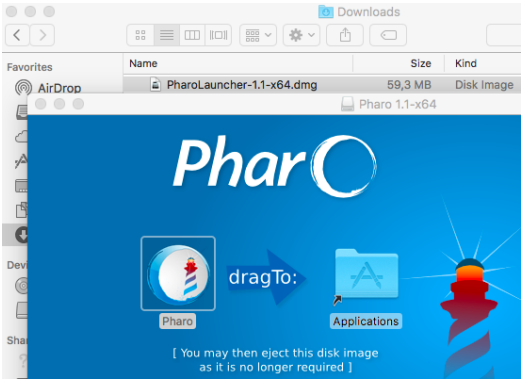

Figure 1-4 Installing PharoLauncher.

1.6 Installing PharoThings on Raspberry Pi

Install PharoThings requires to get Pharo, PharoThings and an ARM virtual machine.

Download PharoLauncher

Use the PharoLauncher (an application to help you running multiple versions and images of Pharo) and install Pharo 6.1. You can get the launcher from http://pharo.org/download. You can also directly install a version of Pharo from the same place.

Download Pharo 61

Run the Pharo Launcher. Double click the distribution you want to create a image and give a name to image (see Figure 1-5). A short name and without spaces is recommended, because we will type this name and path in command line on Linux.

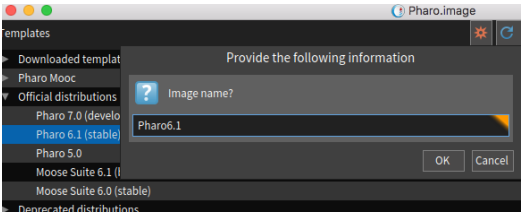

Figure 1-5 Download Pharo 61.

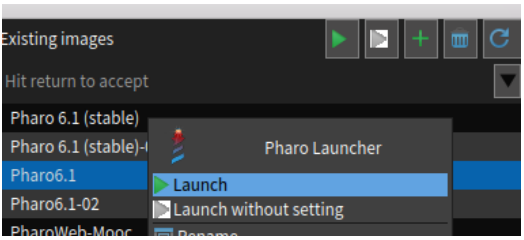

Figure 1-6 Open your Pharo image.

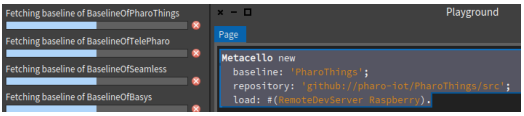

Figure 1-7 Loading PharoThings.

Execute your Pharo image

Launch the image as shown in Figure 1-6. A folder with the image name will be created inside the folder Pharo: `/Users/your_user_name/Documents/Pharo/`

In this example the folder is `/Users/my_user_name/Documents/Pharo/Pharo6.1`

Load PharoThings

Open Playground and execute this command to install the server part of PharoThings (as shown in Figure 1-7):

```
Metacello new
  baseline: 'PharoThings';
  repository: 'github://pharo-iot/PharoThings/src';
  load: #(RemoteDevServer Raspberry).
```

Then configure image to disable slow browser plugins (instead remote browser

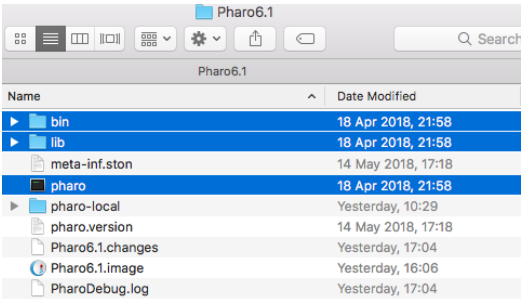

Figure 1-8 Copying PharoARM.

will be much slower):

```
ClySystemEnvironmentPlugin disableSlowPlugins.
```

Snapshot your Image

In Pharo, click and "Save and Quit". This way all your code and configurations are saved and ready to be reused.

Download the VM

- Download ArmVM from http://files.pharo.org/vm/pharo-spur32/linux/armv6/latest.zip.
- Unzip it
- Copy the files shown in FIgure 1-8 to Pharo folder `/Users/your_user_name/Documents/Pharo/pharo_image_folder`

Copying Sources

Copy the file sources from the folder `/Applications/Pharo.app/Contents/MacOS` to folder `/Users/your_user_name/Documents/Pharo/images/pharo_image_folder/lib/pharo/5.0-201804182009/`

Copy to the Raspberry

Copy this folder to your Raspberry Pi (via flashdrive, network etc). The folder must have the structure shown in Figure 1-9.

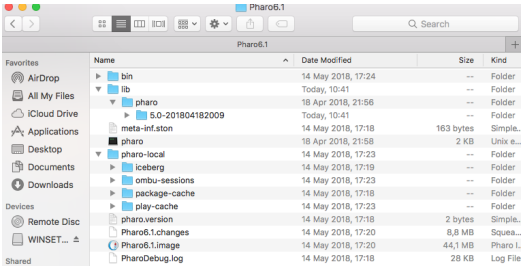

Figure 1-9 Copying the folder on your Raspberry.

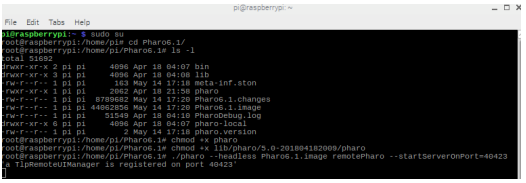

Figure 1-10 Server up and running.

1.7 Execute PharoThings on Raspberry

Turn on your Raspberry and connect it to the network.

In this example, the folder Pharo6.1 was copied to folder `/home/pi/`.

Is necessary apply execute permissions on the Pharo files, using the command chmod +x

```
chmod +x /home/pi/Pharo6.1/pharo

chmod +x /home/pi/Pharo6.1/lib/pharo/5.0-201804182009/pharo
```

Start Server

Start the Pharo typing the following command in the Terminal on your Raspberry :

```
pharo --headless Server.image remotePharo --startServerOnPort=40423
```

If all is right, you will see the answer:

```
'a TlpRemoteUIManager is registered on port 40423'
```

So now we have the Raspberry running the TelePharo on TCP port 40423 (as shown in Figure 1-10) and we can connect into it from another computer.

1.8 Connecting Pharo client on remote Pharo

Open again the Pharo on your local computer and execute this command to install the PharoThings client:

```
Metacello new
baseline: 'PharoThings';
repository: 'github://pharo-iot/PharoThings/src';
load: 'RemoteDev.
```

Type this command to connect to the remote TelePharo on Raspberry (change the IP to your Raspberry IP):

```
remotePharo := TlpRemoteIDE connectTo: (TCPAddress ip: #[193 51 236
    167] port: 40423)
```

Here we are using specialized Raspberry tools. They require the auto-refresh feature of inspector which is not enabled by default in Pharo 6. To activate it evaluate:

```
GTInspector enableStepRefresh
```

So for your board model you need to choose an appropriate board class. For Raspberry, it will be one of the RpiBoard subclasses. Currently, you can use the following classes according to the models:

- RpiBoardBRev1: Raspberry Pi Model B Revision 1
- RpiBoardBRev2: Raspberry Pi Model B Revision 2
- RpiBoard3B: Raspberry Pi Model B+, Pi2 Model B, Pi3 Model B, Pi3 Model B+

With the chosen class evaluate the following code to open an inspector:

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

And will be open the inspector showing the PIN scheme (as show in Figure 1-11)

The board inspector provides a scheme of pins similar to Raspberry Pi docs. But here it is a live tool which represents the current pins state.

In the picture the board is shown with two configured pins: gpio3 and gpio4 which are connected to physical button and led accordingly.

Digital pins are shown with green/red icons which represent high/low (1/0) values. In case of output pins you are able to click on the icon to toggle the value. Icons are updated according to pin value changes. If you click on physical button on your board the inspector will show the updated pin state by changing its icon color.

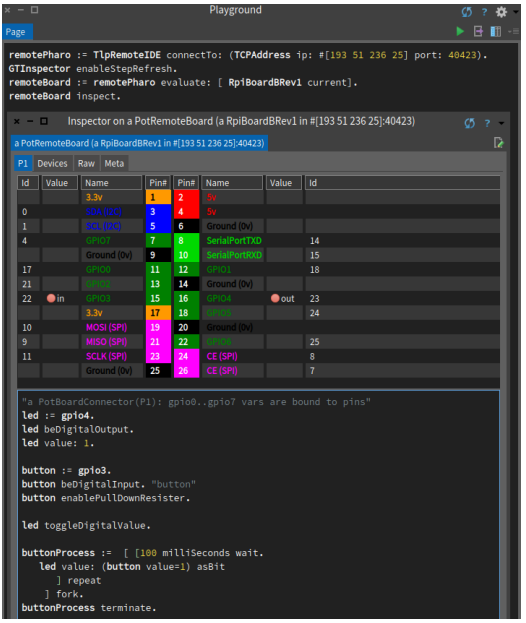

Figure 1-11 Remote GPIO inspector

The evaluation pane in the bottom of the inspector provides bindings to gpio pins which you can script by #doIt/printIt commands. The example shows expressions which were used to configure a button and led.

1.9 In the next chapter

Now that we have installed the Operation System and PharoThings in the Raspberry Pi, we can play with LEDs, sensors, LCD displays and more.

In the next chapter we will see how turing on/off a LED using PharoThings.

CHAPTER 2

Lesson 1 – Turning on/off LED

One of the classic analogies in electronics to "Hello World" is turn on a led or lamp. In this first lesson, we will learn how to connect correctly an LED in your Raspberry Pi and how to use PharoThings to control this led by turning it on and off.

Coding in Pharo is very simple, but it is very powerful and you can control all the GPIOs of your Raspberry Pi remotely.

If you didn't see how to install the PharoThings on your Raspberry Pi and how to control it remotely, you can find the instructions on Chapter 1.

2.1 What we need?

In this lesson we will use a very simple setup.

Components

- 1 Raspberry Pi (any model) connected to your network (wired or wireless)
- 1 Breadboard
- 1 LED
- 1 Resistor (330ohms)
- Jumper wires

2.2 Experimental theory

Before constructing any circuit, you must know the parameters of the components in the circuit, such as their operating voltage, operating circuit, etc.

The LED

To turn on the LED, we need to send the correct voltage and current to it. The voltage and current can't be high, otherwise, the led will burn, or in some cases, damage the Raspberry.

Typically, the forward voltage of an LED is between 1.8 and 3.3 volts. It varies by the color of the LED. A red LED typically drops 1.8 volts, but voltage drop normally rises as the light frequency increases, so a blue LED may drop from 3 to 3.3 volts[1].

Most 3mm and 5mm LEDs will operate close to their peak brightness at a drive current of 20 mA. This is a conservative current: it doesn't exceed most ratings (your specs may vary, or you may not have any specs–in this case, 20 mA is a good default guess [2])

In this experiment, the operating voltage of the LED is between 1.5V and 2.0V and the operating current is between 10mA and 20mA.

The Resistor

We must always use resistors to connect LEDs up to the GPIO pins of the Raspberry Pi to limit the voltage and current between the LED and the Raspberry to a safe value.

A small change in voltage can produce a huge change in current (see more: LED Current vs. Voltage [2])

In this experiment, we will use a 330ohm resistor. To identify the correct resistor, follow one of the following color sequences, depending on the number of bands [3]:

- If there are four colour bands, they will be Orange, Orange, Brown, and then Gold;
- If there are five bands, then the colours will be Orange, Orange, Black, Black, Brown.

It does not matter which way round you connect the resistors (in this experiment). Current flows in both ways through them. This means that you can connect the resistor at the positive pole or the negative pole of the LED, as well as starting with the first or last color, as show in the Picture 2-1.

But the LEDs will only work if the power is supplied correctly (if the polarity is correct). You will not burn the LEDs if they connect the wrong way – they just will not turn on.

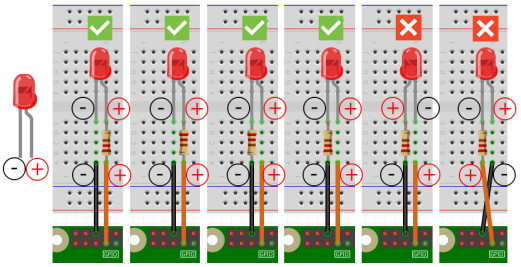

Figure 2-1 Led polarity and resistor.

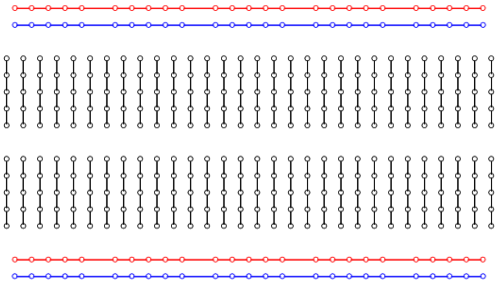

Figure 2-2 Breadboard scheme.

2.3 The Breadboard

A breadboard is used to build prototyping of electronics. With a breadboard, it is not necessary to use solder, this way you can reuse the board. This makes it easy to use to create temporary prototypes and experiment with circuit design.

The holes in the breadboard are connected following a pattern, as shown in the Picture 2-2.

- The red (+) and blue (-) rail are connected horizontally;
- The holes in the middle are connected vertically.

2.4 Experimental procedure

Now we will build the circuit. This circuit consists of an LED that lights up when power is applied, a resistor to limit current and a power supply (the Rasp).

- Connect the Ground PIN from Raspberry in the breadboard blue rail (-). Raspeberry Pi models with 40 pins has 8 GPIO ground pins. You

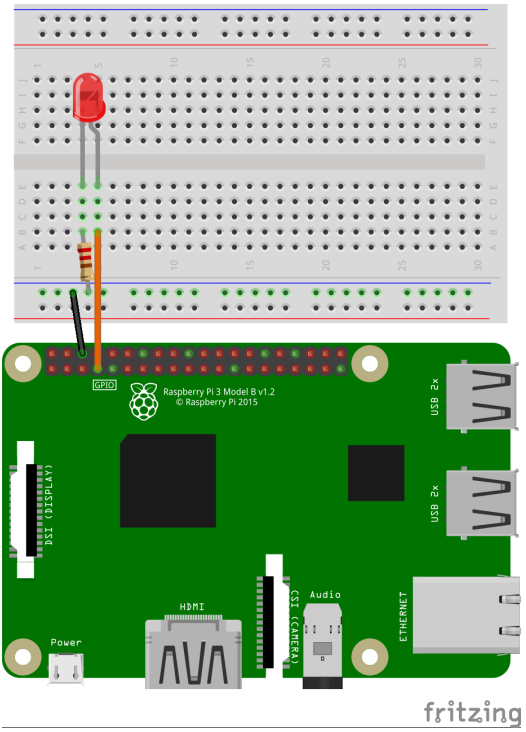

Figure 2-3 Physical connection LED.

can connect with anyone. In this experiment we will use the PIN6 (Ground);

- Then connect the resistor from the same row on the breadboard to a column on the breadboard, as shown below;
- Now push the LED legs into the breadboard, with the long leg (with the kink) on the right;
- And insert a jumper wire connecting the rigth column and the PIN7 (GPIO7).

The Figure 2-3 shows how the electric connection is made:

2.5 Experimental code

Now, we can write some code in Pharo to control the GPIOs using PharoThings and turn the LED on. We have 2 options to do this:

- Write the code locally on the Raspberry;

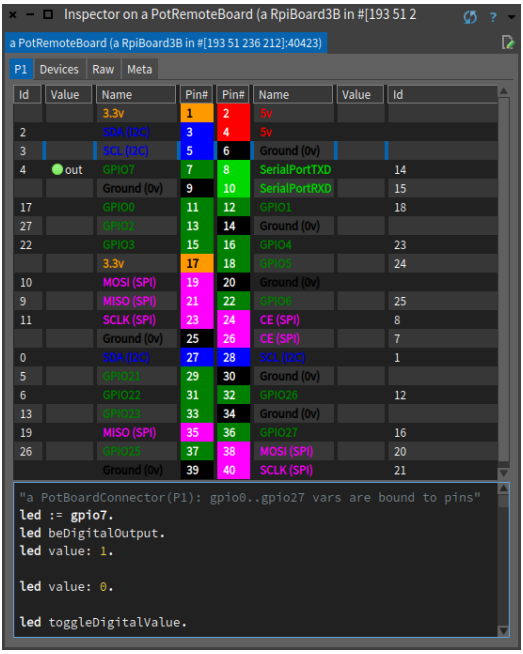

Figure 2-4 Remote Board Inspector.

- Use TelePharo to connect from your computer into the Raspberry and do all the work remotely.

In this experiment, we will use the second option: connect and do all the work remotely.

If you didn't see how to install the PharoThings on your Raspberry Pi and how to control it remotely, take a look in this tutorial: Installing PharoThings on your Raspberry Pi

In your inspect window (Inspector on a PotRemoteBoard) you can see a scheme of pins similar to Raspberry Pi docs. But here it is a live tool which represents the current pins state[4].

Digital pins are shown with green/red icons which represent high/low (1/0) values. In case of output pins you are able to click on the icon to toggle the value.

For control the led we first introduced named variable #led which we assigned to GPIO7 pin instance:

```
led := gpio7.
```

Then we configured the pin to be in digital output mode and set the value:

```
led beDigitalOutput.
led value: 1.
```

It turned the led on.

You can notice that gpio variables are not just numbers/ids. PharoThings models boards with first class pins. They are real objects with behaviour. For example you can ask pin to toggle a value:

```
led toggleDigitalValue.
```

Or ask a pin for current value if you want to check it:

```
led value.
```

2.6 What did we learn?

With PharoThings you can remotely control all the GPIOs in your running board!

You can:

- Interact remotely with pins and boards;
- See the current pins state in real time;
- Run the code dynamically.
- Easy, powerful.

2.7 In the next lesson...

Let's use what we learned in this lesson and write a simple code to blink the LED.

CHAPTER 3

Lesson 2 – Blinking LED