A PharoThings Tutorial

Allex Oliveira

November 7, 2018

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

Contents

Illustrations

CHAPTER 1

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.

1.1 What we need?

In this lesson we will use a very simple setup.

Components

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

1.2 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 shown in the Picture 1-1.

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.

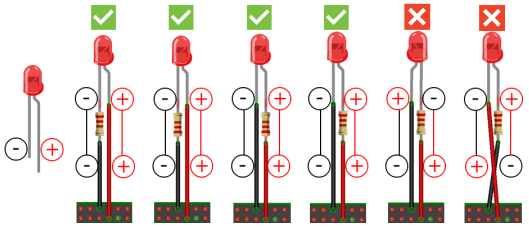

Figure 1-1 Led polarity and resistors.

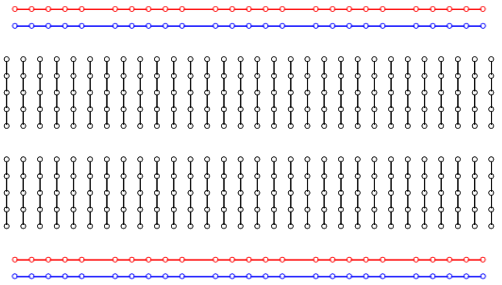

Figure 1-2 Breadboard scheme.

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 1-2.

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

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

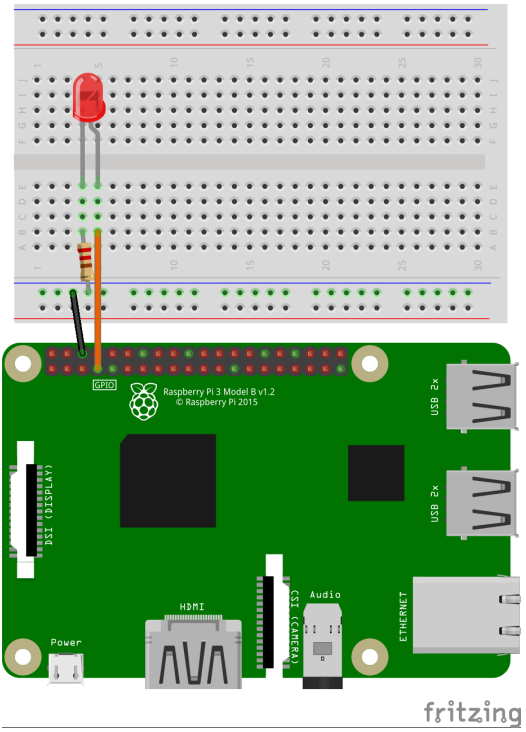

Figure 1-3 Physical connection LED.

- 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 1-3 shows how the electric connection is made:

1.4 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;
- Use TelePharo to connect from your computer into the Raspberry and do all the work remotely.

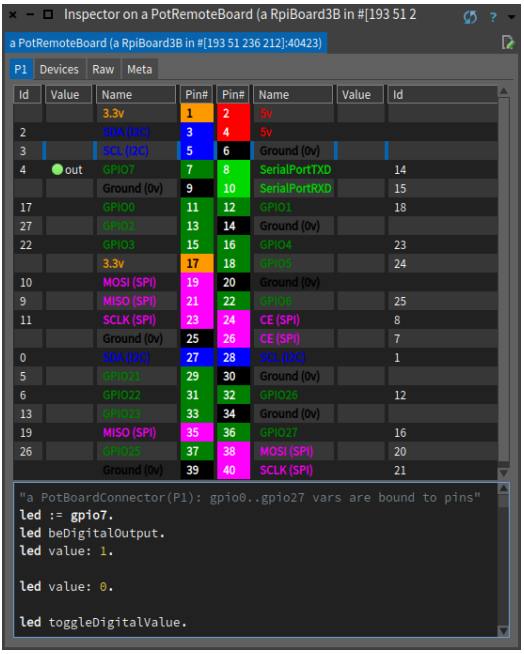

Figure 1-4 Remote Board Inspector.

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 the Chapter 1: Instalattions.

Connecting remotely

Through your local Pharo image, let's connect in the Pharo image running on Raspberry, enable the auto-refresh feature of the inspector and open the inspector. Run this code in your local playground:

```
remotePharo := TlpRemoteIDE connectTo: (TCPAddress ip: #[193 51 236
    167] port: 40423)
GTInspector enableStepRefresh
remoteBoard := remotePharo evaluate: [ RpiBoardBRev1 current].
remoteBoard inspect.
```

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.
```

1.5 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.

1.6 In the next lesson

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