A PharoThings Tutorial

Allex Oliveira

January 7, 2019

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

Contents

Illustrations

CHAPTER 1

Lesson 7 - Controlling LED by Button

In the previous lessons, we learned how to control the GPIOs putting them in mode OUT. This means send power to wire connected in on specific GPIO. Now we will put the GPIO in mode IN, to read the pin state. This means that our application can know when a button is pressed. Let's create a shortcode on the remote inspector to turn On one LED each time we press the button.

1.1 What do we need?

We are using the set of the first lesson, but let's use 8 LEDs and 8 resistors and some more jumper wires.

Components

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

1.2 Experimental theory

The Raspberry Pi GPIOs can be set OUT or IN mode. When we set them to IN, the selected pin can have change the state from 0 to 1 or vice-versa when we connect it in the 3.3V or ground PIN.

In this lesson, we will use the pull-up mode with an internal resistor. This means that we will use the 3.3V pin to change the GPIO value. If we used the pull-down mode, we would use the Ground Pin to change the GPIO value. To know more, you can access the Wiring Pi website, http://wiringpi.com/reference/core-functions.

In this scenario, we will put a push button between the 3.3V pin and GPIO pin, to control when to send energy to GPIO.

1.3 Experimental procedure

Now we will build the circuit. This circuit consists of an LED, a resistor to limit current and a push button, to control when to send power to GPIO IN.

- Connect the Ground PIN from Raspberry in the breadboard blue rail (-). 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 right column and the PIN7 (GPIO7);
- Connect the 3.3V PIN in the red rail (+);
- Then insert the push button on the breadboard and connect one leg on PIN11 (GPIO0);
- And another button leg on the red rail (+). Pay attention to the position of the button!

The figure below shows how the electric connection is made:

1.4 Connecting remotely

Through your local Pharo image, let's connect in the Pharo image by 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
    212] port: 40423)
GTInspector enableStepRefresh.
remoteBoard := remotePharo evaluate: [ RpiBoard3B current].
remoteBoard inspect.
```

1.5 Experimental code

In your inspect window (Inspector on a PotRemoteBoard), let's create the instances of the LED and button.

To instantiate the LED, let's do how we did previous, putting it in beDigitalOutput mode:

```
led := gpio7
led beDigitalOutput.
```

To instantiate the button let's do the same, but putting it in beDigitalInput and setting it to enablePullUpResister mode:

```
button := gpio0.
button beDigitalInput.
button enablePullUpResister.
```

This means each time that we connect the 3.3V on this GPIO (pushing the button), the state will be changed to 1, and back to 0 after release the button.

You ask the value of the button using the value method. Do this test running this line with the button pressed and after with the button released. This test is good for you to check if your button is working correctly:

```
button value.
```

Now let's create a process to check when the button is pressed and send this value to led object:

```
[ [100 milliSeconds wait.
    led value: (button value=1) asBit
        ] repeat
     ] forkNamed: 'button process'.
```

Your LED will turn On when the button is pressed!

1.6 Terminating the process

As we saw in the Blinking LED lesson, you can finish this process remotely, case you don't want to wait it finish. To do this, call the Remote Process Browser:

```
remotePharo openProcessBrowser.
```

Search the FlowingProcess and terminate it:

- selecting the process and using the shortcut “Cmd + T”;
- selecting the process and using the button Terminate;
- or right-click and select Terminate.

1.7 In the next lesson

In this lesson, you learned how to configure a push button, take the value of the button and send it to LED value, doing the LED turn On or Off using the button.

Next lesson we will start to use the sensors using the I2C protocol. We will see the BME280 (temperature, humidity, and air pressure), ADXL (accelerometer X, Y, Z) and MCP9808 (temperature).