Final Raspberry Pi lab

This activity puts together everything we have done with the Raspberry Pi:

- basic electronics;
- running GNU/Linux applications in the LXDE environment and the LXTerminal;
- using the Python IDLE editor and shell, and the Turtle Graphics module.

The goal is to write a simple interactive Python program to control LEDs with mouse clicks.

Step 1: draw a traffic light control panel

1. Open the IDLE 3 program.
2. Click the menu **File : New Window** to create a blank file.
3. Save the file to **/home/pi** with the name **trafficpanel1.py**
4. Type the code below and save it.
5. Hit the **F5** key to test it.

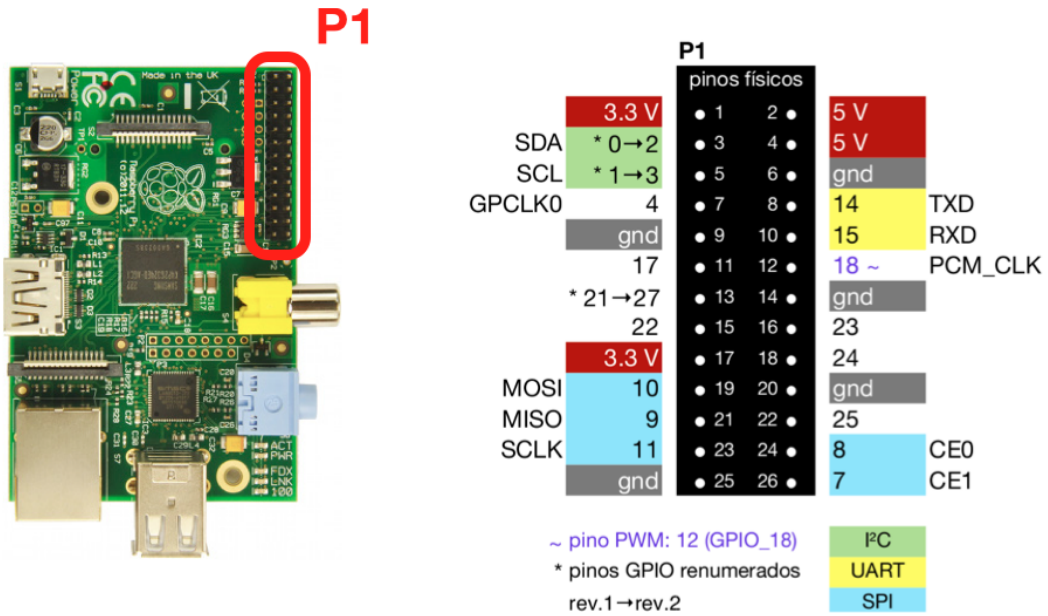
```
from turtle import *

def square(color):
    fillcolor(color)
    begin_fill()
    fd(100)
    rt(90)
    fd(100)
    rt(90)
    fd(100)
    rt(180)
    end_fill()

def draw_panel():
    setworldcoordinates(0, 0, 300, 300)
    speed(0)
    penup()
    goto(100, 300)
    square('red')
    square('yellow')
    square('green')


draw_panel()

mainloop()
```

Step 2: make the control panel interactive

1. Use the menu command **File : Save as** to save the **trafficpanel1.py** file as **trafficpanel2.py**
2. Add the code between the `# step 2` comments and save to **trafficpanel2.py**.
3. Hit the **F5** key to test it. Click on the colored squares and note the output in the shell.

```
from turtle import *

def square(color):
    fillcolor(color)
    begin_fill()
    fd(100)
    rt(90)
    fd(100)
    rt(90)
    fd(100)
    rt(180)
    end_fill()

def draw_panel():
    setworldcoordinates(0, 0, 300, 300)
    speed(0)
    penup()
    goto(100, 300)
    square('red')
    square('yellow')
    square('green')

# step 2: type code FROM this line
def switch(x, y):
    print('click at y = ', y)
    if 0 <= y < 100:
        bgcolor('dark green')
    elif 100 <= y < 200:
        bgcolor('#880')
    else:
        bgcolor('dark red')

onscreenclick(switch)
# step 2: type code TO this line

draw_panel()

mainloop()
```

Step 3: build the traffic light circuit



Step 3.1: connect wires from Raspberry Pi to breadboard

1. Refer to the picture above to identify the Raspberry Pi GPIO physical pins **3, 5, 7** and **25**. Note: all the odd pins are on the left, away from the edge, starting with pin **1** (labelled **3.3V**).
2. Connect a **red** wire from the Raspberry Pi GPIO **pin 3** to **column 4** of the breadboard.
3. Connect a **yellow** wire from the Raspberry Pi GPIO **pin 5** to **column 8** of the breadboard.
4. Connect a **green** wire from the Raspberry Pi GPIO **pin 7** to **column 12** of the breadboard.
5. Connect a **blue** wire from the Raspberry Pi GPIO **pin 25** to the **blue line** of the breadboard.

Step 3.2: install resistors on breadboard

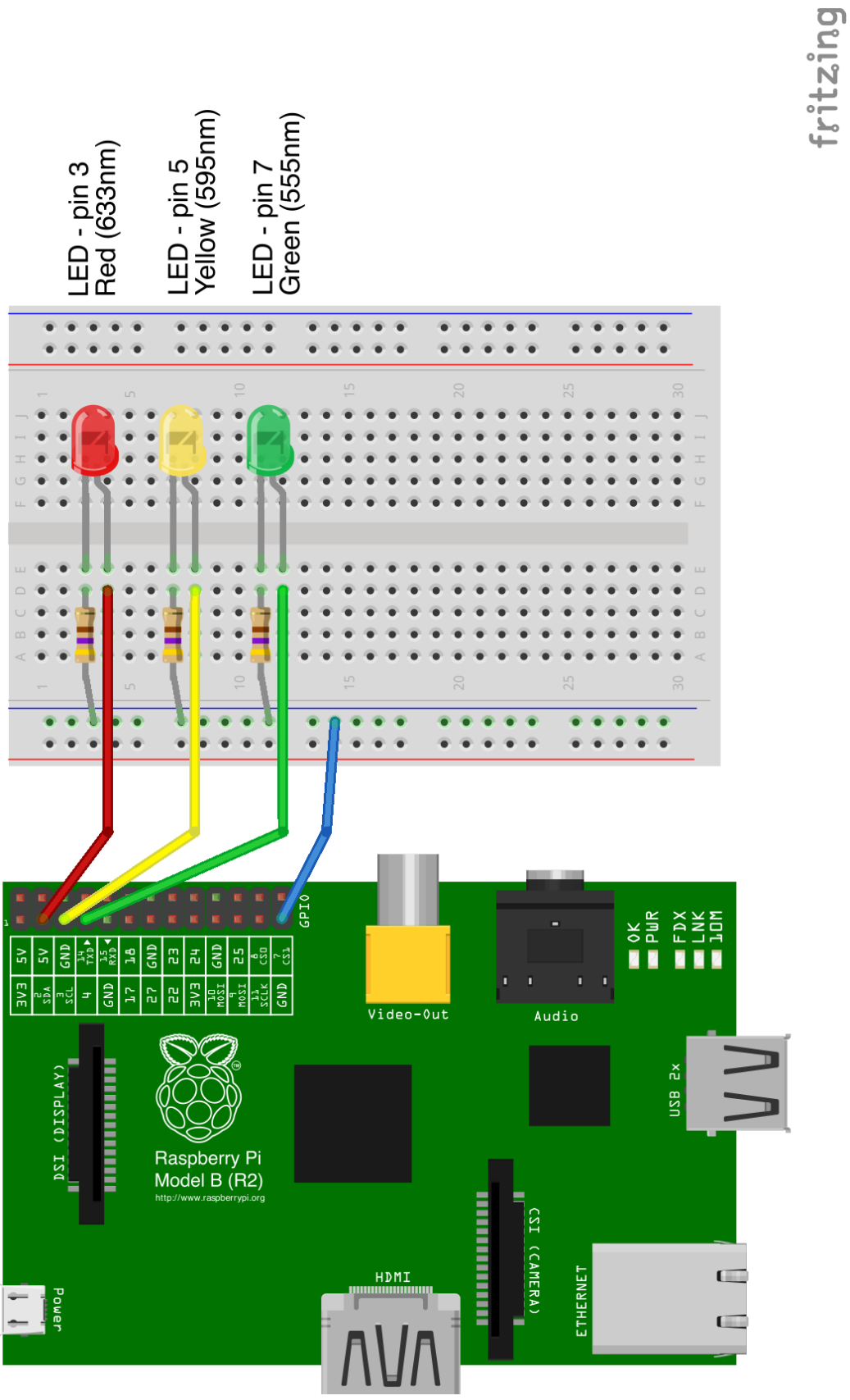
1. Connect a **470Ω** (or similar) resistor from **column 3** to the **blue line** of the breadboard.
2. Connect a **470Ω** (or similar) resistor from **column 7** to the **blue line** of the breadboard.
3. Connect a **470Ω** (or similar) resistor from **column 11** to the **blue line** of the breadboard.

Step 3.3: install LEDs on breadboard

1. Connect a **red** LED with anode (long lead) in **column 4** and cathode (short lead) in **column 3.**
2. Connect a **yellow** LED with anode (long lead) in **column 8** and cathode (short lead) in **column 7.**
3. Connect a **green** LED with anode (long lead) in **column 12** and cathode (short lead) in **column 11.**

Step 3.4: revise all the connections

Refer to the picture below to make sure everything is connected correctly.



Step 4: write test program to make LEDs blink

Step 4.1: type code for leds.py

1. In the IDLE 3 program, click the menu **File : New Window** to create a new file.
2. Save the file to **/home/pi** with the name **leds.py**
3. Type the code below and save it to **leds.py**.

```
from time import sleep
import atexit
import RPi.GPIO as GPIO

atexit.register(GPIO.cleanup)
GPIO.setmode(GPIO.BOARD)

LED_PINS = [('red', 3), ('yellow', 5), ('green', 7)]


def activate(active_color):
    for color, pin in LED_PINS:
        GPIO.output(pin, color == active_color)


for color, pin in LED_PINS:
    GPIO.setup(pin, GPIO.OUT)
    GPIO.output(pin, 0)


if __name__ == '__main__':

    for color, pin in LED_PINS * 3:
        print color
        activate(color)
        sleep(.5)
```

Step 4.2: test the leds.py program

1. Open the **LXTerminal** program.
2. At the $ prompt, type:
   $ sudo python leds.py

Note: Python programs that control the GPIO pins must be executed by **root** (the super user, or system administrator). Therefore the program above cannot be tested using F5 from IDLE if you are logged in as the **pi** user. You must open the terminal and use the **sudo** command to run the program as **root** (sudo is short for “super-user do”).

Step 5: Integrate the traffic panel to the LEDs

Step 5.1: Edit the traffic panel program

You need to add just 4 lines to the **trafficpanel2.py** program to make **trafficpanel3.py**

1. In the IDLE 3 program, open the **trafficpanel2.py** program.
2. Use the menu command **File : Save as** to save it as **trafficpanel3.py** in the **/home/pi** folder.
3. Add the 4 lines of code marked with `# <-- step 5`

```
from turtle import *
from leds import activate       # <-- step 5

def square(color):
    fillcolor(color)
    begin_fill()
    fd(100)
    rt(90)
    fd(100)
    rt(90)
    fd(100)
    rt(180)
    end_fill()

def draw_panel():
    setworldcoordinates(0, 0, 300, 300)
    speed(0)
    penup()
    goto(100, 300)
    square('red')
    square('yellow')
    square('green')

# step 2: type code FROM this line
def switch(x, y):
    print('click at y = ', y)
    if 0 <= y < 100:
        bgcolor('dark green')
        activate('green')       # <-- step 5
    elif 100 <= y < 200:
        bgcolor('#880')
        activate('yellow')      # <-- step 5
    else:
        bgcolor('dark red')
        activate('red')         # <-- step 5

onscreenclick(switch)
# step 2: type code TO this line

draw_panel()

mainloop()
```

Step 5.2: test the integrated traffic panel

1. Open the **LXTerminal** program.
2. At the $ prompt, type these two lines:
   $ xhost +

   $ sudo python trafficpanel3.py
3. If everything worked, you should be able to control de traffic LEDs by clicking on the colored squares in the graphical interface of **trafficpanel3.py**

Note: this last step shows how to run a graphical program, **trafficpanel3.py** as **root** using **sudo**. The problem is that we are logged in as the **pi** user and when we use **sudo** to run a graphical program as the **root** user, GNU/Linux will not allow it because another user cannot control the current graphics display. The **xhost +** command allows other users, including **root**, to connect to the display.