

## Raspberry Pi - Développement d'applications

1. On peut utiliser différents langages de programmation pour afficher *Hello world* :

- C/C++

```
pascalpiszyna — pi@L3P-RPi11: ~/C — ssh pi@10.10.11.217 — 78x13
[pi@L3P-RPi11:~ $ gcc --version
gcc (Raspbian 4.9.2-10) 4.9.2
Copyright (C) 2014 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[pi@L3P-RPi11:~ $ pwd
/home/pi
[pi@L3P-RPi11:~ $ md C
-bash: md: command not found
[pi@L3P-RPi11:~ $ mkdir C
[pi@L3P-RPi11:~ $ cd C
[pi@L3P-RPi11:~/C $ nano hello.c
```

```
pascalpiszyna — pi@L3P-RPi11: ~/C — ssh pi@10.10.11.217 — 78x12
GNU nano 2.2.6 File: hello.c Modified

#include <stdio.h>

main()
{
    printf("Bonjour tout le monde\n");
}

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page  ^U UnCut Text ^T To Spell
```

```
pascalpiszyna — pi@L3P-RPi11: ~/C1 — ssh pi@10.10.11.217 — 80x8
[pi@L3P-RPi11:~/C1 $ ls
hello.c
[pi@L3P-RPi11:~/C1 $ gcc hello.c -o hello
[pi@L3P-RPi11:~/C1 $ ls
hello hello.c
[pi@L3P-RPi11:~/C1 $ ./hello
Bonjour tout le monde
[pi@L3P-RPi11:~/C1 $ ]
```

- Java

```
pascalpiszyna — pi@L3P-RPi11: ~/Java — ssh pi@10.10.11.217 — 78x7
[pi@L3P-RPi11:~ $ mkdir Java ]
[pi@L3P-RPi11:~ $ cd Java ]
[pi@L3P-RPi11:~/Java $ java -version ]
java version "1.8.0_65"
Java(TM) SE Runtime Environment (build 1.8.0_65-b17)
Java HotSpot(TM) Client VM (build 25.65-b01, mixed mode)
pi@L3P-RPi11:~/Java $ nano hello.java
```

```
pascalpiszyna — pi@L3P-RPi11: ~/Java — ssh pi@10.10.11.217 — 78x10
GNU nano 2.2.6 File: Hello.java

public class Hello {
    public static void main(String[] args) {
        System.out.println("Bonjour tout le monde");
    }
}

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

```
pascalpiszyna — pi@L3P-RPi11: ~/Java — ssh pi@10.10.11.217 — 78x5
[pi@L3P-RPi11:~/Java $ nano Hello.java ]
[pi@L3P-RPi11:~/Java $ javac Hello.java ]
[pi@L3P-RPi11:~/Java $ java Hello ]
Bonjour tout le monde
pi@L3P-RPi11:~/Java $
```

- Python

```
pascalpiszyna — pi@L3P-RPi11: ~ — ssh pi@10.10.11.217 — 78x13
[pi@L3P-RPi11:~ $ python
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
[>>> a=5
[>>> b=9
[>>> c=a+b
[>>> c
14
[>>> quit
Use quit() or Ctrl-D (i.e. EOF) to exit
[>>> quit()
pi@L3P-RPi11:~ $ ]
```

- Javascript (node.js)

```
pascalpiszyna — pi@L3P-RPi11: ~/JS — ssh pi@10.10.11.217 — 78x8
[pi@L3P-RPi11:~/C $ cd ..
[pi@L3P-RPi11:~ $ pwd
/home/pi
[pi@L3P-RPi11:~ $ mkdir JS
[pi@L3P-RPi11:~ $ cd JS
[pi@L3P-RPi11:~/JS $ node --version
v0.10.29
pi@L3P-RPi11:~/JS $ nano mynode.js ]
```

```
pascalpiszyna — pi@L3P-RPi11: ~/JS — ssh pi@10.10.11.217 — 78x8
GNU nano 2.2.6 File: mynode.js Modified
console.log('Bonjour tout le monde');

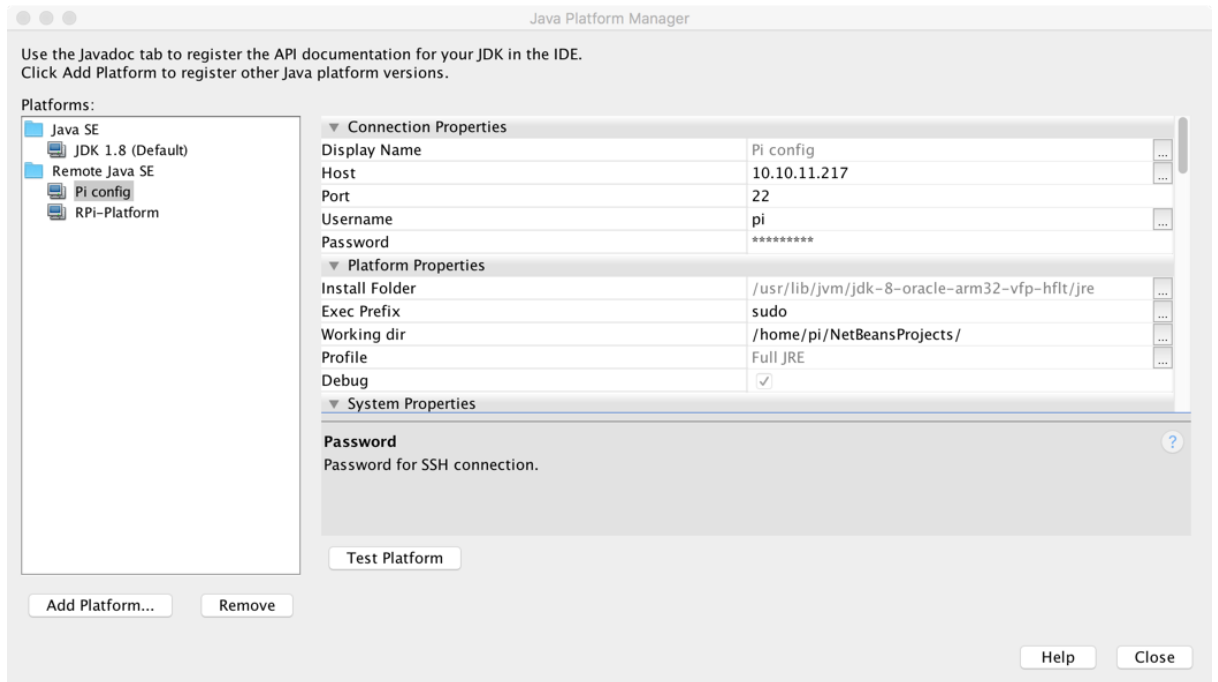
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

```
pascalpiszyna — pi@L3P-RPi11: ~/JS — ssh pi@10.10.11.217 — 78x5
[pi@L3P-RPi11:~/JS $ node mynode.js
Bonjour tout le monde
[pi@L3P-RPi11:~/JS $
[pi@L3P-RPi11:~/JS $
pi@L3P-RPi11:~/JS $ ]
```

2. Avec Netbeans, créer une plateforme Raspberry Pi :

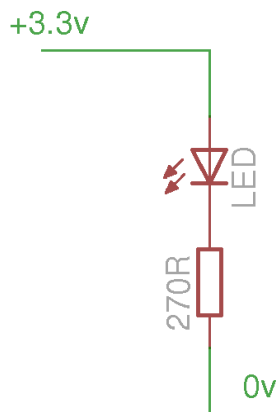
- <https://blog.idrsolutions.com/2014/08/using-netbeans-remotely-deploy-projects-raspberry-pi/>

On obtient au final:



3. Créer une première application *Hello World* pour le Raspberry Pi

4. Câbler un circuit LED + résistance (270 ohms) entre la broche GPIO.1 et la masse :



5. Tester les commandes spécifiques au GPIO :

```

pascalpiszyna — pi@L3P-RPi11: ~/Java — ssh pi@10.10.11.217 — 88x31
[pi@L3P-RPi11:~/Java $ gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 2 | 8 | 3.3v | | | 1 | 2 | | | 5v | | |
| 3 | 9 | SDA.1 | IN | 1 | 3 | 4 | | | 5V | | |
| 4 | 7 | SCL.1 | IN | 1 | 5 | 6 | | | 0v | | |
| 4 | 7 | GPIO. 7 | IN | 1 | 7 | 8 | 1 | ALT0 | TxD | 15 | 14 |
| | | 0v | | | 9 | 10 | 1 | ALT0 | RxD | 16 | 15 |
| 17 | 0 | GPIO. 0 | IN | 0 | 11 | 12 | 0 | OUT | GPIO. 1 | 1 | 18 |
| 27 | 2 | GPIO. 2 | IN | 0 | 13 | 14 | | | 0v | | |
| 22 | 3 | GPIO. 3 | IN | 0 | 15 | 16 | 0 | IN | GPIO. 4 | 4 | 23 |
| | | 3.3v | | | 17 | 18 | 0 | IN | GPIO. 5 | 5 | 24 |
| 10 | 12 | MOSI | IN | 0 | 19 | 20 | | | 0v | | |
| 9 | 13 | MISO | IN | 0 | 21 | 22 | 0 | IN | GPIO. 6 | 6 | 25 |
| 11 | 14 | SCLK | IN | 0 | 23 | 24 | 1 | IN | CE0 | 10 | 8 |
| | | 0v | | | 25 | 26 | 1 | IN | CE1 | 11 | 7 |
| 0 | 30 | SDA.0 | IN | 1 | 27 | 28 | 1 | IN | SCL.0 | 31 | 1 |
| 5 | 21 | GPIO.21 | IN | 1 | 29 | 30 | | | 0v | | |
| 6 | 22 | GPIO.22 | IN | 1 | 31 | 32 | 0 | IN | GPIO.26 | 26 | 12 |
| 13 | 23 | GPIO.23 | IN | 0 | 33 | 34 | | | 0v | | |
| 19 | 24 | GPIO.24 | IN | 0 | 35 | 36 | 0 | IN | GPIO.27 | 27 | 16 |
| 26 | 25 | GPIO.25 | IN | 0 | 37 | 38 | 0 | IN | GPIO.28 | 28 | 20 |
| | | 0v | | | 39 | 40 | 0 | IN | GPIO.29 | 29 | 21 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
[pi@L3P-RPi11:~/Java $ gpio mode 1 out
[pi@L3P-RPi11:~/Java $ gpio write 1 1
[pi@L3P-RPi11:~/Java $ gpio write 1 0
[pi@L3P-RPi11:~/Java $ ]

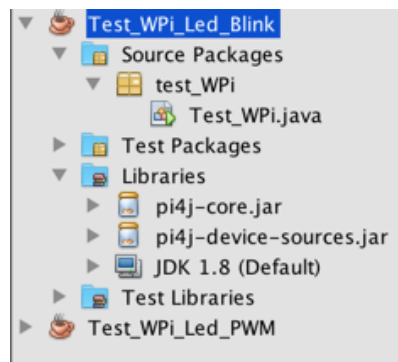
```

6. Télécharger la librairie Pi4J :

- <http://pi4j.com/index.html>
- <http://pi4j.com/apidocs/>
- <http://wiringpi.com/>

7. Créer une première application (Led clignotante) intégrant la librairie Pi4J :

Il faut inclure les fichiers JAR suivants dans les librairies du projet:



- 1° méthode : avec la librairie Wiring Pi :

```

1  package test_WPi;
2
3  import com.pi4j.wiringpi.Gpio;
4
5  public class Test_WPi {
6
7      public static void main(String[] args) throws InterruptedException {
8          System.out.println("<--WiringPi--> Led Example ... started.");
9          // initialize wiringPi library
10         Gpio.wiringPiSetup();
11         Gpio.pinMode(1, Gpio.OUTPUT);
12         for (int counter = 0; counter < 10; counter++) {
13             // LED ON
14             Gpio.digitalWrite(1, Gpio.HIGH);
15             Thread.sleep(1000);
16
17             // LED OFF
18             Gpio.digitalWrite(1, Gpio.LOW);
19             Thread.sleep(1000);
20         }
21     }
22 }

```

- 2° méthode : avec la librairie Pi4J :

```

1  package test_WPi;
2
3  import com.pi4j.io.gpio.GpioController;
4  import com.pi4j.io.gpio.GpioFactory;
5  import com.pi4j.io.gpio.GpioPinDigitalOutput;
6  import com.pi4j.io.gpio.PinState;
7  import com.pi4j.io.gpio.RaspiPin;
8
9  public class Test_WPi {
10
11     public static void main(String[] args) throws InterruptedException {
12         System.out.println("<--WiringPi--> Led Example ... started.");
13         // initialize wiringPi library, this is needed for PWM
14         final GpioController gpio = GpioFactory.getInstance();
15         final GpioPinDigitalOutput pin = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01, "PinLED", PinState.LOW);
16
17         for (int counter = 0; counter < 10; counter++) {
18             // LED ON
19             pin.high();
20             Thread.sleep(1000);
21
22             // LED OFF
23             pin.low();
24             Thread.sleep(1000);
25
26             // release the GPIO controller resources
27             gpio.shutdown();
28         }
29     }
30 }

```

- Créer une deuxième application (Modulation d'intensité lumineuse) intégrant la librairie Wiring Pi :

```
1 package test_pi4j;
2
3 import com.pi4j.wiringpi.Gpio;
4 import com.pi4j.wiringpi.SoftPwm;
5
6 public class Test_Pi4J {
7
8     private static final int PIN_NUMBER = 1;
9
10    public static void main(String[] args) throws InterruptedException {
11        System.out.println("<---Pi4J--> Led PWM Example ... started.");
12        // initialize wiringPi library, this is needed for PWM
13        Gpio.wiringPiSetup();
14        // softPwmCreate(int pin, int value, int range)
15        // the range is set like (min=0 ; max=100)
16        SoftPwm.softPwmCreate(PIN_NUMBER, 0, 100);
17        int counter = 0;
18        while (counter < 4) {
19            // fade LED to fully ON
20            for (int i = 0; i <= 100; i++) {
21                // softPwmWrite(int pin, int value)
22                // This updates the PWM value on the given pin. The value is
23                // checked to be in-range and pins
24                // that haven't previously been initialized via softPwmCreate
25                // will be silently ignored.
26                SoftPwm.softPwmWrite(PIN_NUMBER, i);
27                Thread.sleep(25);
28            }
29            // fade LED to fully OFF
30            for (int i = 100; i >= 0; i--) {
31                SoftPwm.softPwmWrite(PIN_NUMBER, i);
32                Thread.sleep(25);
33            }
34            counter++;
35        }
36    }
37 }
```

On peut aussi lancer un exécutable Java (exemple pour un source *prog.java*) par la console :

```
sudo java -jar /home/pi/NetBeansProjects//prog/dist/prog.jar
```