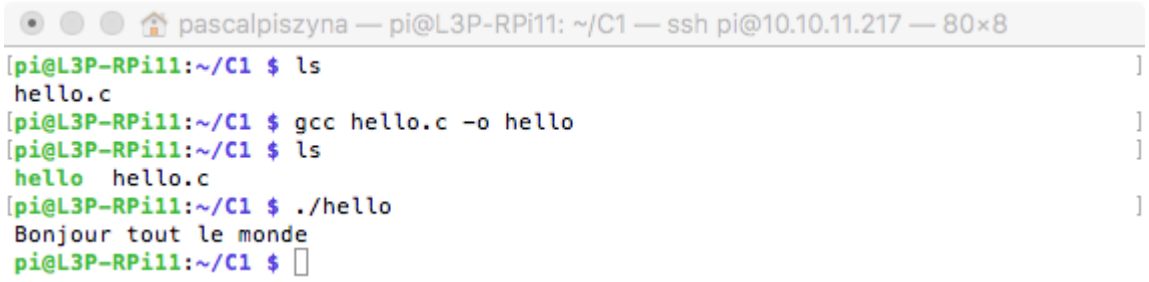
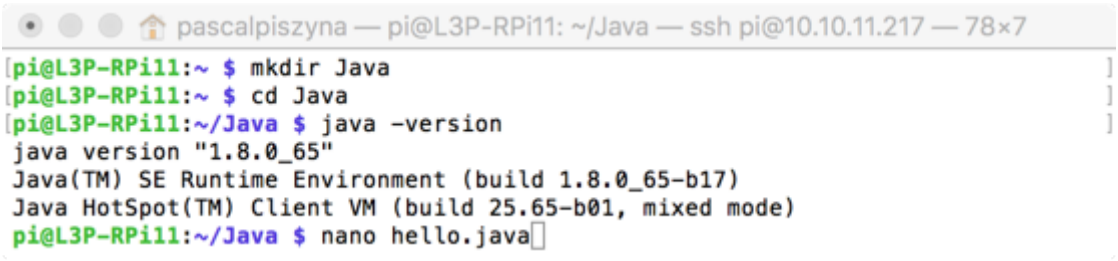
Raspberry Pi - Développement d'applications

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

   - **C/C++**



- **Java**


pascalpiszyna — pi@L3P-RPi11: ~/Java — ssh pi@10.10.11.217 — 78×7
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

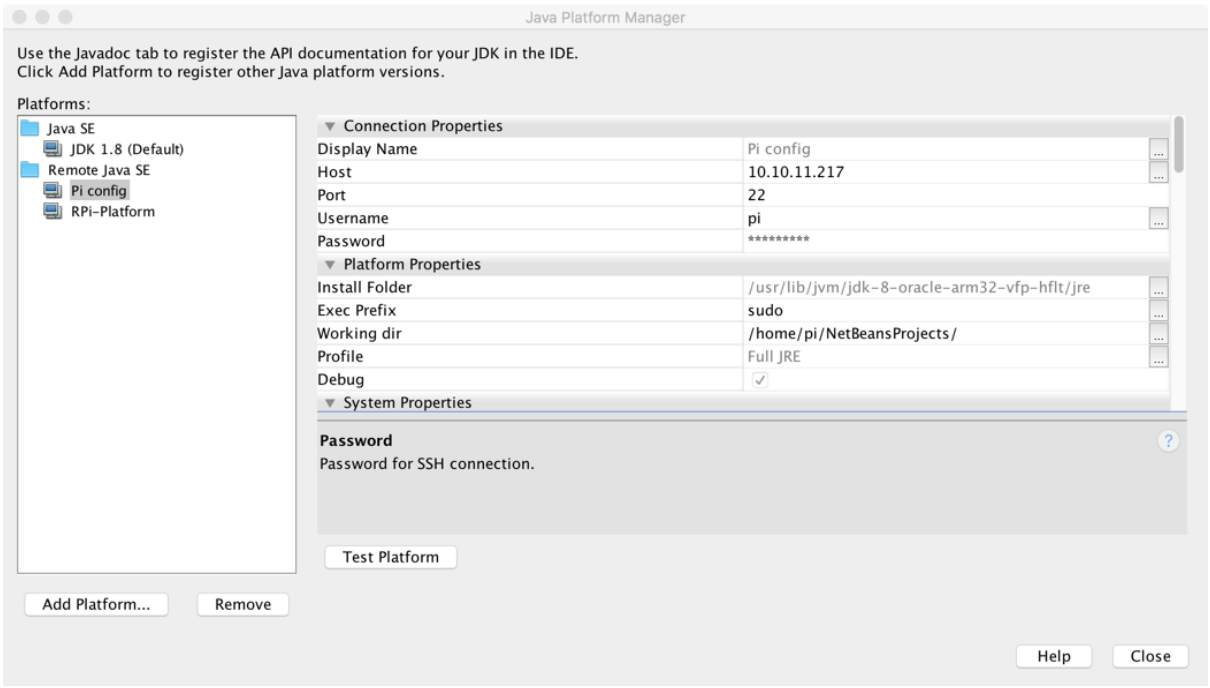

pascalpiszyna — pi@L3P-RPi11: ~/Java — ssh pi@10.10.11.217 — 78×10
GNU nano 2.2.6 File: Hello.java
public class Hello {
public static void main(String[] args) {
System.out.println("Bonjour tout le monde");
}
}
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell


pascalpiszyna — pi@L3P-RPi11: ~/Java — ssh pi@10.10.11.217 — 78×5
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**

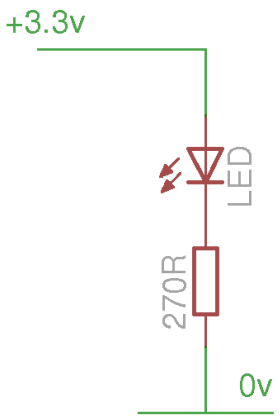
- **Javascript (node.js)**

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 :



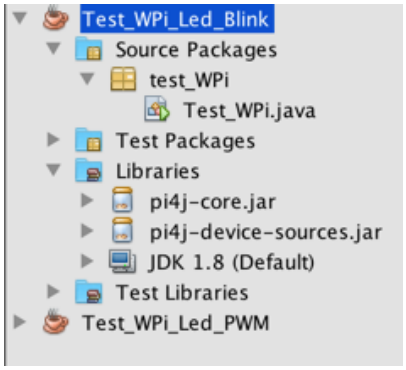
5. Tester les commandes spécifiques au GPIO :

```
pascalpiszyna — pi@L3P-RPi11: ~/Java — ssh pi@10.10.11.217 — 88×31
pi@L3P-RPi11:~/Java $ gpio readall
 +-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   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 2---+---+------+---------+-----+-----+
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 :

```java
package test_WPi;

import com.pi4j.wiringpi.Gpio;

public class Test_WPi {

    public static void main(String[] args) throws InterruptedException {
        System.out.println("<--WiringPi--> Led Example ... started.");
        // initialize wiringPi library
        Gpio.wiringPiSetup();
        Gpio.pinMode(1, Gpio.OUTPUT);
        for (int counter = 0; counter < 10; counter++) {
            // LED  ON
            Gpio.digitalWrite(1, Gpio.HIGH);
            Thread.sleep(1000);

            //  LED  OFF
            Gpio.digitalWrite(1, Gpio.LOW);
            Thread.sleep(1000);
        }
    }
}
```

- 2° méthode : avec la librairie Pi4J :

```java
package test_WPi;

import com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.io.gpio.GpioPinDigitalOutput;
import com.pi4j.io.gpio.PinState;
import com.pi4j.io.gpio.RaspiPin;

public class Test_WPi {

    public static void main(String[] args) throws InterruptedException {
        System.out.println("<--WiringPi--> Led Example ... started.");
        // initialize wiringPi library, this is needed for PWM
        final GpioController gpio = GpioFactory.getInstance();
        final GpioPinDigitalOutput pin = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01, "PinLED", PinState.LOW);

        for (int counter = 0; counter < 10; counter++) {
            // LED  ON
            pin.high();
            Thread.sleep(1000);

            //  LED  OFF
            pin.low();
            Thread.sleep(1000);

            // release the GPIO controller resources
            gpio.shutdown();
        }
    }
}
```

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

```java
package test_pi4j;

import com.pi4j.wiringpi.Gpio;
import com.pi4j.wiringpi.SoftPwm;

public class Test_Pi4J {

    private static final int PIN_NUMBER = 1;

    public static void main(String[] args) throws InterruptedException {
        System.out.println("<--Pi4J--> Led PWM Example ... started.");
        // initialize wiringPi library, this is needed for PWM
        Gpio.wiringPiSetup();
        // softPwmCreate(int pin, int value, int range)
        // the range is set like (min=0 ; max=100)
        SoftPwm.softPwmCreate(PIN_NUMBER, 0, 100);
        int counter = 0;
        while (counter < 4) {
            // fade LED to fully ON
            for (int i = 0; i <= 100; i++) {
                // softPwmWrite(int pin, int value)
                // This updates the PWM value on the given pin. The value is
                // checked to be in-range and pins
                // that haven't previously been initialized via softPwmCreate
                // will be silently ignored.
                SoftPwm.softPwmWrite(PIN_NUMBER, i);
                Thread.sleep(25);
            }
            // fade LED to fully OFF
            for (int i = 100; i >= 0; i--) {
                SoftPwm.softPwmWrite(PIN_NUMBER, i);
                Thread.sleep(25);
            }
            counter++;
        }
    }
}
```

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**