

PROGETTO 2 REGISTRATORE DI TEMPERATURA

Il progetto digitale qui proposto è un "Registratore di Temperatura". La motivazione per farlo è in alcuni progetti misti analogico-digitali già attivi a LNL (sismografi, registratore di punto di rugiada) si sono osservate derive forse legate alle variazioni di temperatura giornaliere e stagionali. Quindi, meglio verificare!

Per realizzare questo progetto si userà un sensore di temperatura (DS18B20 Temperature Sensor) in cui l'elettronica di digitalizzazione e trasmissione dati (ADC, gestore di bus seriale) è già integrata nel modulo con il sensore.

I problemi da risolvere sono legati alla lettura del dato e alla registrazione dei file con le misure e l'informazione aggiuntiva di data-ora.

Prosecuzione: si tratta di graficare i dati, per una più immediata ed agevole lettura e renderli disponibili su un sito web.

Con gli studenti è importante arrivare a registrare i dati; grafici e sito web possono essere lasciati alla loro fantasia e stile.

Condividere l'apprendimento e i progetti è sempre molto importante, prima di tutto con gli studenti. Potrebbe anche essere utile creare un blog per descrivere anche questo progetto.

Qui di seguito sono riportate 2 tracce per la gestione iniziale del DS18B20, la prima utilizza il driver "nativo" e poi legge e seleziona il dato reso disponibile nel file-system.

La seconda traccia usa pure il driver nativo, ma anche una libreria python "ad hoc" che consente di semplificare un po' il programma di lettura.

Suggerimenti:

```
1. abilitare 1-Wire (da Preferences → Raspberry Pi Configuration → Interfaces)
( 1-bis sudo apt-get update; sudo apt-get upgrade; sudo apt-get autoclean )
2. sudo modprobe w1-therm
3. cd /sys/bus/w1/devices
4. ls 28-00000xxxxxxx
5. cd 28-0000054bfcc0
6. cat w1_slave
questo mostra la temperatura letta dal sensore:
t=28625, indica una temperatura di 28.625 gradi Celsius.
```

Connessioni:

Rosso: pin 4 (5 V)
Nero: pin 6 (ground)
Bianco: pin 7 (bus – GPIO-3)

1. Raspberry Pi DS18B20 Temperature Sensor Tutorial

The DS18B20 temperature sensor is perfect for projects like weather stations and home automation systems. Few sensors are this easy to set up on the Raspberry Pi. They're the same size as a transistor and use only one wire for the data signal. They're also extremely accurate and take measurements quickly. The only other component you need is a 4.7K Ohm or 10K Ohm resistor.

Digital Temperature Sensors vs. Analog Temperature Sensors

Digital temperature sensors like the DS18B20 differ from analog thermistors in several important ways. In thermistors, changes in temperature cause changes in the resistance of a ceramic or polymer semi-conducting material. Usually, the thermistor is set up in a voltage divider, and the voltage is measured between the thermistor and a known resistor. The voltage measurement is converted to resistance and then converted to a temperature value by the micro-controller.

Digital temperature sensors are typically silicon based integrated circuits. Most contain the temperature sensor, an analog to digital converter (ADC), memory to temporarily store the temperature readings, and an interface that allows communication between the sensor and a micro-controller. Unlike analog temperature sensors, calculations are performed by the sensor, and the output is an actual temperature value.

About the DS18B20

The DS18B20 communicates with the "One-Wire" communication protocol, a proprietary serial communication protocol that uses only one wire to transmit the temperature readings to the micro-controller.

The DS18B20 can be operated in what is known as *parasite power* mode. Normally the DS18B20 needs three wires for operation: the Vcc, ground, and data wires. In parasite mode, only the ground and data lines are used, and power is supplied through the data line.

The DS18B20 also has an alarm function that can be configured to output a signal when the temperature crosses a high or low threshold that's set by the user.

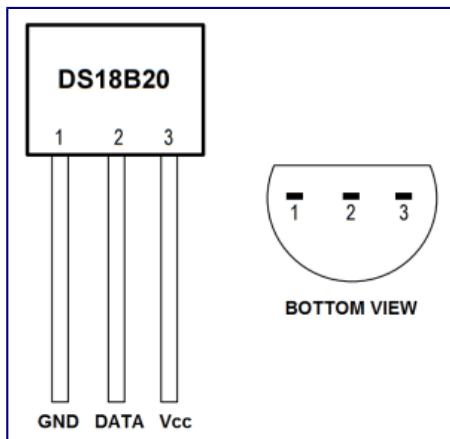
A 64 bit ROM stores the device's unique serial code. This 64 bit address allows a micro-controller to receive temperature data from a virtually unlimited number of sensors at the same pin. The address tells the micro-controller which sensor a particular temperature value is coming from.

Technical Specifications

- -55°C to 125°C range
- 3.0V to 5.0V operating voltage
- 750 ms sampling
- 0.5°C (9 bit); 0.25°C (10 bit); 0.125°C (11 bit); 0.0625°C (12 bit) resolution
- 64 bit unique address
- One-Wire communication protocol

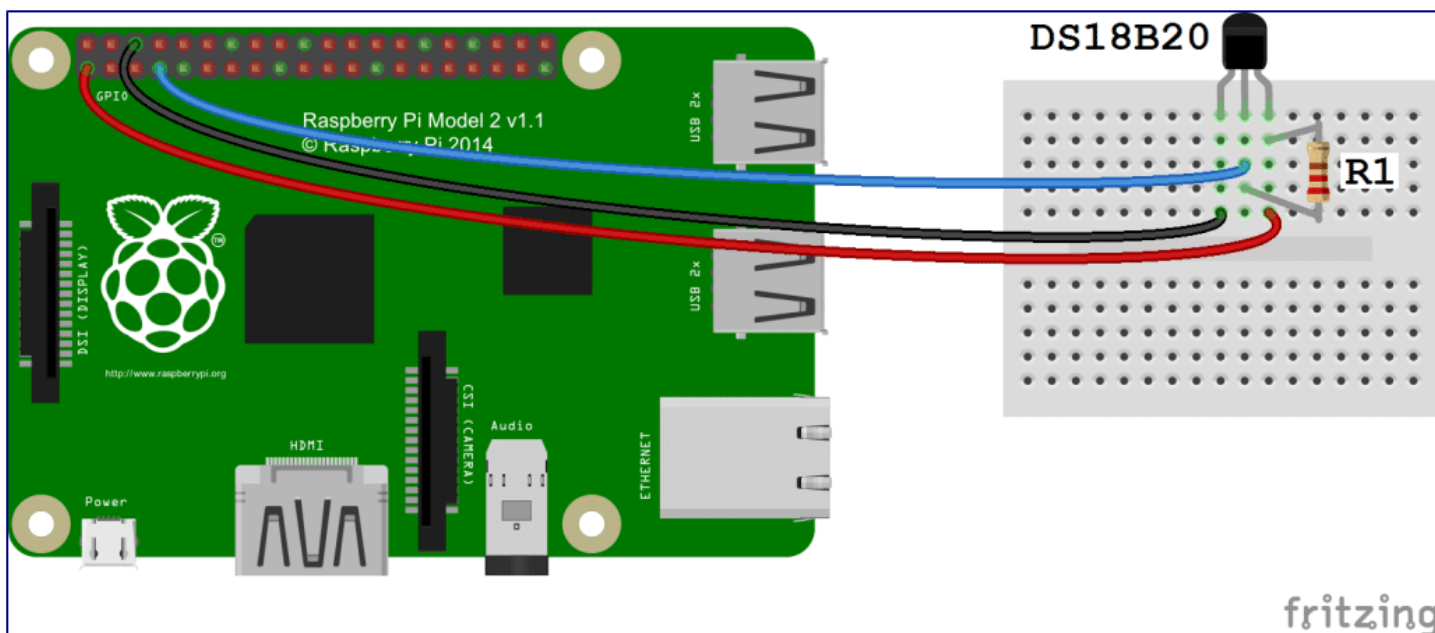
Connect the DS18B20 to the Raspberry Pi

The DS18B20 has three separate pins for ground, data, and Vcc:



Wiring for SSH Terminal Output

Follow this wiring diagram to output the temperature to an SSH terminal:



R1: 4.7K Ohm or 10K Ohm resistor

IMPORTANT: connect the device to the GPIO with the Raspberry OFF

black: ground – red: 3.3 V – white-blue: 4

Enable the One-Wire Interface

We'll need to enable the One-Wire interface before the Pi can receive data from the sensor. Once you've connected the DS18B20, power up your Pi and log in, then follow these steps to enable the One-Wire interface:

1. At the command prompt, enter: `sudo nano /boot/config.txt`, then add this to the bottom of the file (or use Preferences→Rasperry PI configuration)

```
dtoverlay=w1-gpio
```

2. Exit Nano, and reboot the Pi (`sudo reboot`)

3. Log in to the Pi again, and at the command prompt enter `sudo modprobe w1-gpio` (NON SERVE PIU', a aprtire da una certa release)

4. Then enter

```
sudo modprobe w1-therm
```

5. Change directories to the `/sys/bus/w1/devices` directory by entering:

```
cd /sys/bus/w1/devices
```

6. Now enter `ls` to list the devices:

28-000006637696 w1_bus_master1 is displayed in my case.

7. Now enter `cd 28-XXXXXXXXXXXX` (change the X's to your own address)

For example, in my case I would enter: `cd 28-000006637696`

8. Enter `cat w1_slave` which will show the raw temperature reading output by the sensor:

Here the temperature reading is `t=28625`, which means a temperature of 28.625 degrees Celsius.

9. Enter `cd` to return to the root directory

That's all that's required to set up the one wire interface. Now you can run one of the programs below to output the temperature to an SSH terminal.

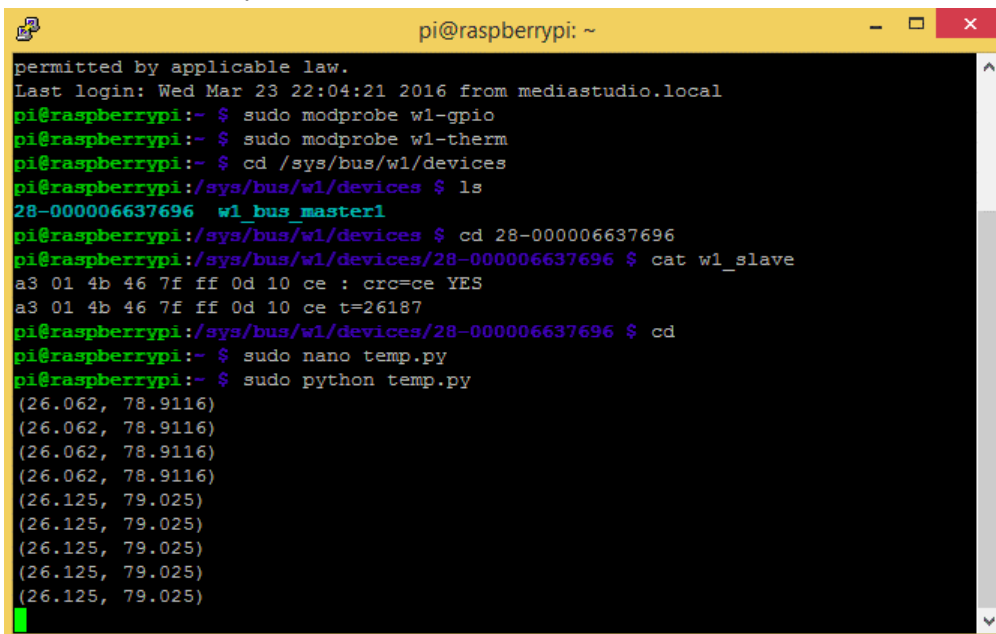
Programming the Temperature Sensor

The examples below are written in Python. If this is your first time running a Python program, check out our tutorial [How to Write and Run a Python Program on the Raspberry Pi](#) to see how to save and run Python files.

Temperature Output to SSH Terminal

This is a basic Python program that will output the temperature readings in Celsius and Fahrenheit to your SSH terminal:

```
import os
import glob
import time
// os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')
base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'
def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines
def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c, temp_f
while True:
    print(read_temp())
    time.sleep(1)
```



```
pi@raspberrypi: ~
permitted by applicable law.
Last login: Wed Mar 23 22:04:21 2016 from mediastudio.local
pi@raspberrypi:~$ sudo modprobe w1-gpio
pi@raspberrypi:~$ sudo modprobe w1-therm
pi@raspberrypi:~$ cd /sys/bus/w1/devices
pi@raspberrypi:/sys/bus/w1/devices$ ls
28-000006637696  w1_bus_master1
pi@raspberrypi:/sys/bus/w1/devices$ cd 28-000006637696
pi@raspberrypi:/sys/bus/w1/devices/28-000006637696$ cat w1_slave
a3 01 4b 46 7f ff 0d 10 ce : crc=ce YES
a3 01 4b 46 7f ff 0d 10 ce t=26187
pi@raspberrypi:/sys/bus/w1/devices/28-000006637696$ cd
pi@raspberrypi:~$ sudo nano temp.py
pi@raspberrypi:~$ sudo python temp.py
(26.062, 78.9116)
(26.062, 78.9116)
(26.062, 78.9116)
(26.062, 78.9116)
(26.125, 79.025)
(26.125, 79.025)
(26.125, 79.025)
(26.125, 79.025)
(26.125, 79.025)
(26.125, 79.025)
```

2. DS18B20 Temperature Sensor With Python (Raspberry Pi)

What will you need?

- A Raspberry Pi (any model)
- A DS18B20 Temperature Sensor
- A 4.7K Ohm Resistor (Colour Code: Yellow Purple Red Gold)
- 3 jumper cables.
- An Internet connection for your Raspberry Pi

Getting Started

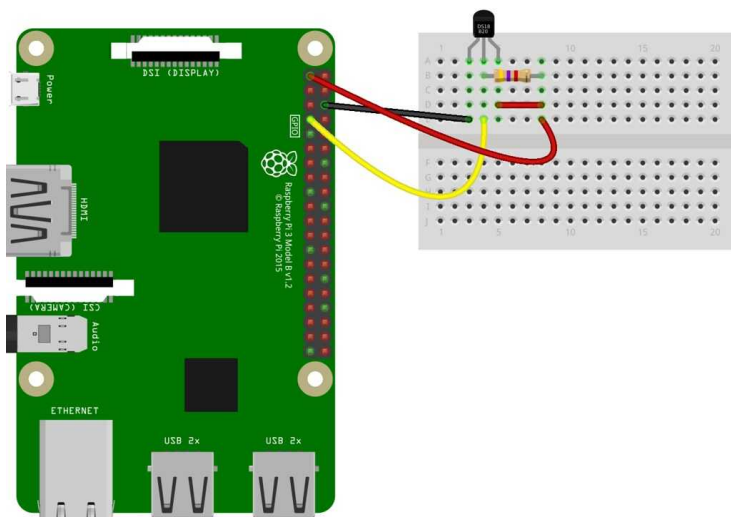
With your Raspberry Pi turned off, build the circuit according to the diagram below.

The DS18B20 is placed into the breadboard so that the **flat side faces you**.

- The black jumper cable goes from GND, which is the third pin down on the right column to the first pin of the DS18B20.
- The yellow jumper cable goes from the fourth pin down on the left column and is connected to the middle pin of the DS18B20.
- The red jumper cable goes from the top left pin of the Raspberry Pi to the far right pin of the DS18B20.

The Resistor connects the RIGHT pin to the MIDDLE pin. This is called a **pull up resistor** and is used to ensure that the middle pin is always on. In the diagram I had to use a spare red wire to show this connection. But in reality, using the resistor to make the connection, as per this photo is the best way.

black: ground – red: 3.3 V – white-blue-yellow : 4



fritzing

Now attach your keyboard, mouse, HDMI and power to your Raspberry Pi and boot to the desktop.

Configuring the Raspberry Pi

We now need to take two steps to enable our DS18B20 for use.

Install the Python Library

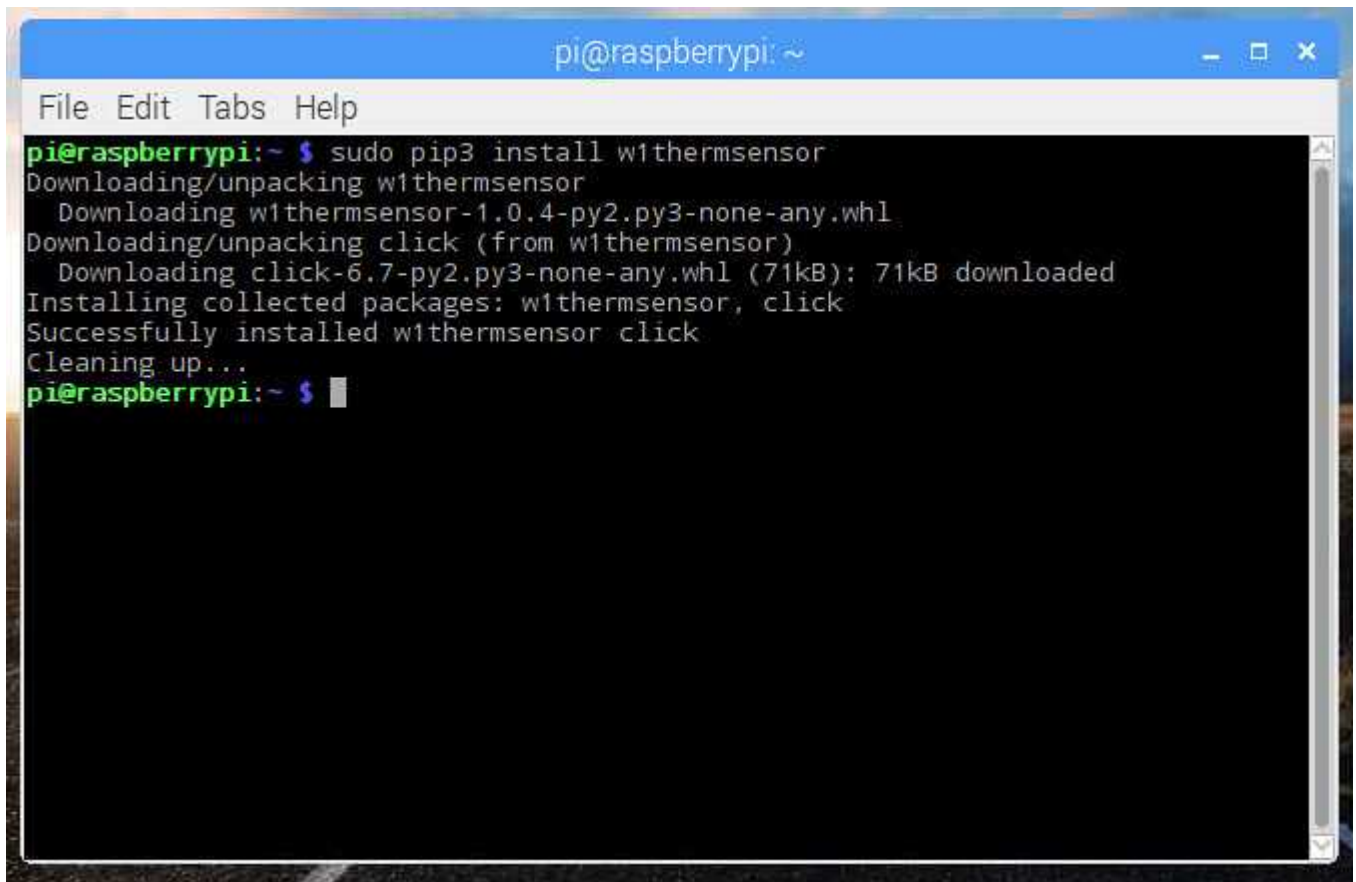
Firstly we need to install a Python library, pre-written code that enables the Python code that we shall later write to talk to the sensor. The Python library is called **w1thermsensor** and to install it

we need to use the Terminal. You can find the terminal icon in the top left of the screen. It looks like...



When the terminal opens, enter the following to install the library, just press **ENTER** to start.

```
sudo pip3 install w1thermsensor
```

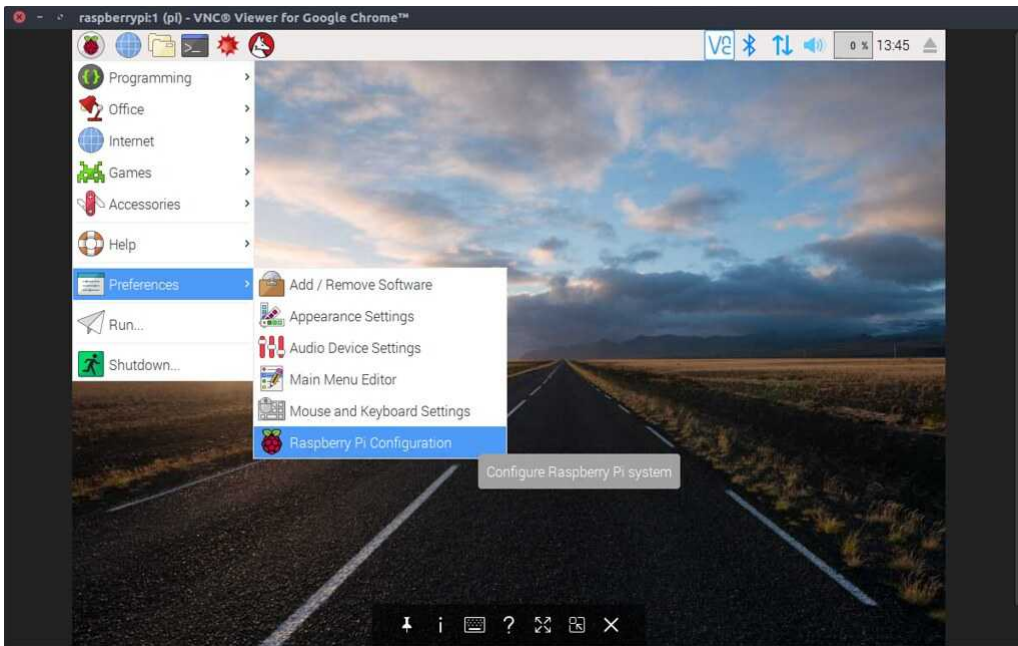
A screenshot of a terminal window on a Raspberry Pi. The window title is 'pi@raspberrypi: ~'. The terminal output shows the command 'sudo pip3 install w1thermsensor' being executed. The output includes: 'Downloading/unpacking w1thermsensor', 'Downloading w1thermsensor-1.0.4-py2.py3-none-any.whl', 'Downloading/unpacking click (from w1thermsensor)', 'Downloading click-6.7-py2.py3-none-any.whl (71kB): 71kB downloaded', 'Installing collected packages: w1thermsensor, click', 'Successfully installed w1thermsensor click', and 'Cleaning up...'. The prompt 'pi@raspberrypi:~ \$' is visible at the end of the output.

```
pi@raspberrypi:~ $ sudo pip3 install w1thermsensor
Downloading/unpacking w1thermsensor
  Downloading w1thermsensor-1.0.4-py2.py3-none-any.whl
Downloading/unpacking click (from w1thermsensor)
  Downloading click-6.7-py2.py3-none-any.whl (71kB): 71kB downloaded
Installing collected packages: w1thermsensor, click
Successfully installed w1thermsensor click
Cleaning up...
pi@raspberrypi:~ $
```

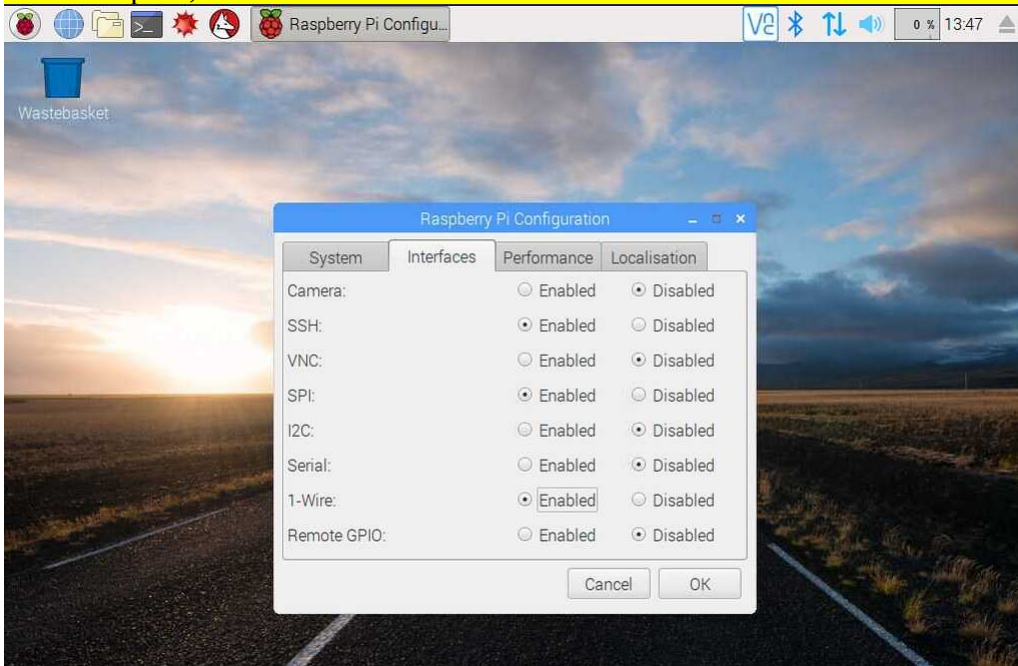
That's it, now you can close the Terminal window.

Enable the Interface

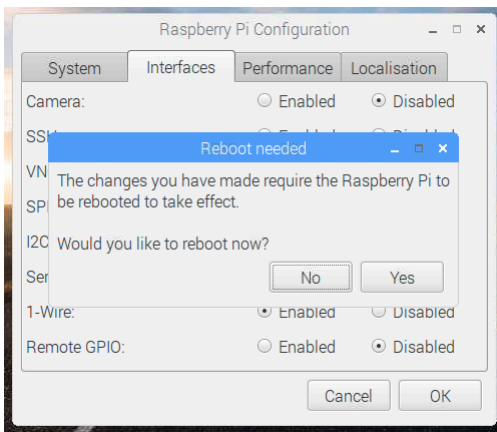
The DS18B20 uses a 1 wire serial interface, this is the middle pin of the sensor, that is connected to the Raspberry Pi via the yellow wire in the diagram. We need to tell our Raspberry Pi that we are using this pin and to do that we use the **Raspberry Pi Configuration** tool, found in the **Preferences** menu.



When it opens, click on the **Interfaces** tab and then click on **Enable** for the **1-Wire** interface.



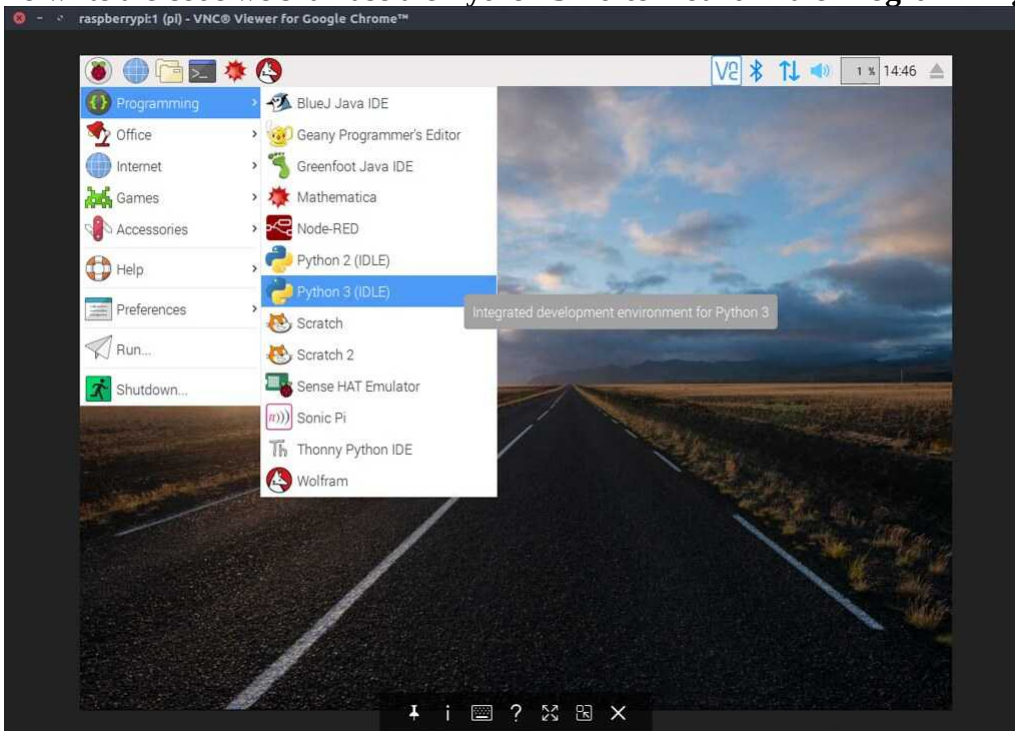
Now click on **Ok** and you will be asked to reboot, so go ahead and do that, and let the Raspberry Pi reboot to the desktop.



Writing the Python Code

Our project will gather the temperature from the DS18B20 sensor every one second and print it to the screen. The code will run forever.

To write the code we shall use the **Python 3 Editor** found in the **Programming** menu.



When the application opens, click on **File >> New** to create a new blank document. In this new window, click on **File >> Save** and call the project **temperature-sensor.py**. Remember to save your work often!

Importing the libraries

The first step in any Python project that uses libraries is to **import** the libraries that we wish to use. In this case we import **time** to control how often the sensor data is collected, and we import **w1thermsensor** to enable our project to talk to the sensor.

So let's import the libraries.

```
import time
from w1thermsensor import W1ThermSensor
```

Sensor

Our next line is to create an object to store a connection to the sensor. So rather than typing `W1ThermSensor()` everytime we want to use the sensor, we store the connection in an object called `sensor`.

```
sensor = W1ThermSensor()
```

Running in a loop

We'd like to get the temperature sensor data every second, and run forever. So let's use a **while True** loop to run the code inside of it forever.

```
while True:
```

Now the next lines of code are **indented**, this is how Python shows that this code belongs inside the loop that we have just created.

The first thing to do in our loop is to get the current temperature from the DS18B20 sensor, and then store it in a variable called **temperature**. Variables are boxes / containers into which we can store any data.

```
temperature = sensor.get_temperature()
```

Now that we have the data, let's print it to the screen using the `print()` function. But let's use the data in the form of a sentence that will tell us what the temperature is in Celsius. For this we use a little Python trick called **string formatting**, where we would like the temperature data to be printed in the sentence, we use an `%s` which will format the temperature data from a **float** (a number with a decimal place) to a **string** (text, characters that can be printed but not used in any mathematical equations)

```
print("The temperature is %s celsius" % temperature)
```

Our last line of Python code will tell the Raspberry Pi to wait for 1 second between taking a temperature reading.

```
time.sleep(1)
```

That's all of the code, so make sure that you save your work.

Complete Code Listing

Check that your code matches the code below before moving on.

```
import time
```

```
from w1thermsensor import W1ThermSensor
```

```
sensor = W1ThermSensor()
```

```
while True:
```

```
    temperature = sensor.get_temperature()
```

```
    print("The temperature is %s celsius" % temperature)
```

```
    time.sleep(1)
```

Run the Code!

