Welcome!

Thank you for purchasing our *AZ-Delivery KY-001 DS18B20 Digital Temperature Sensor Module.* On the following pages, you will be introduced to how to use and set up this handy device.

Have fun!

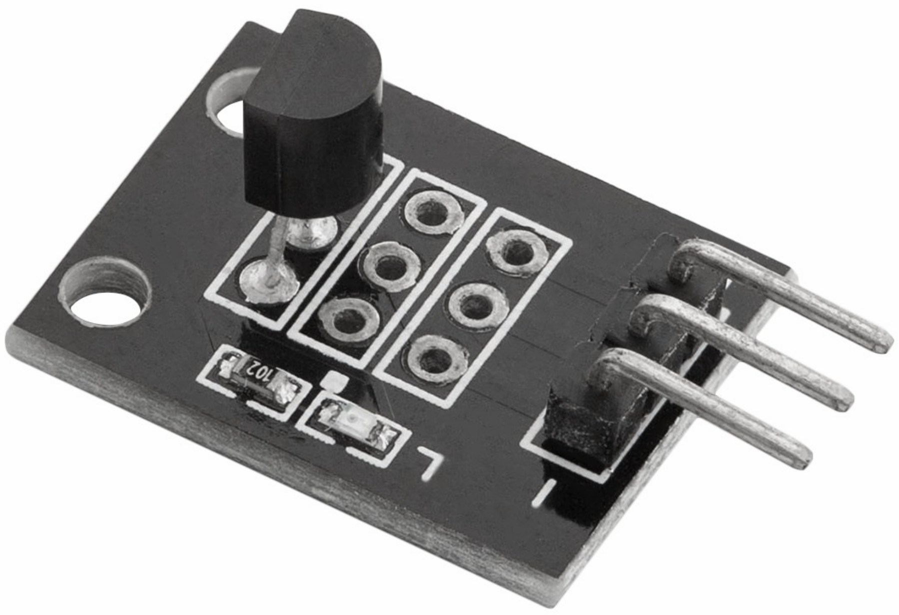


Table of Contents

Introduction

The DS18B20 is a digital temperature sensor that provides *9* to *12* bits digital temperature measurements and has an alarm function with nonvolatile user programmable upper and lower trigger points. The sensor communicates over the `One-Wire` bus that requires only one data pin, power supply pin and ground pin for communication with a microcontroller.

Each sensor has a unique *64*-bit serial address, which allows multiple sensors to function on the same One-Wire bus. Thus, it is simple to use one microcontroller to control many sensors distributed over a large area. Applications that can benefit from this feature include temperature monitoring systems inside buildings, equipment, or machinery, process monitoring and control systems.

Specifications

The KY-001 module has a digital temperature sensor DS18B20, an LED and a resistor. The external power supply voltage range is from *3.3V* to *5.5V DC*. The sensor does not require standby power, which means that when temperature data is not read, the sensor does not use power at all.

Measurement temperature range is from *-55°C* to *+125°C* (*67°F* to *257°F*), with an accuracy of ±*0.5°C* (*9* bit); ±*0.25°C* (*10* bit); ±*0.125°C* (*11* bit); and ±*0.0625°C* (*12* bit) resolution. Dimensions of the module: *15 x 19mm*.

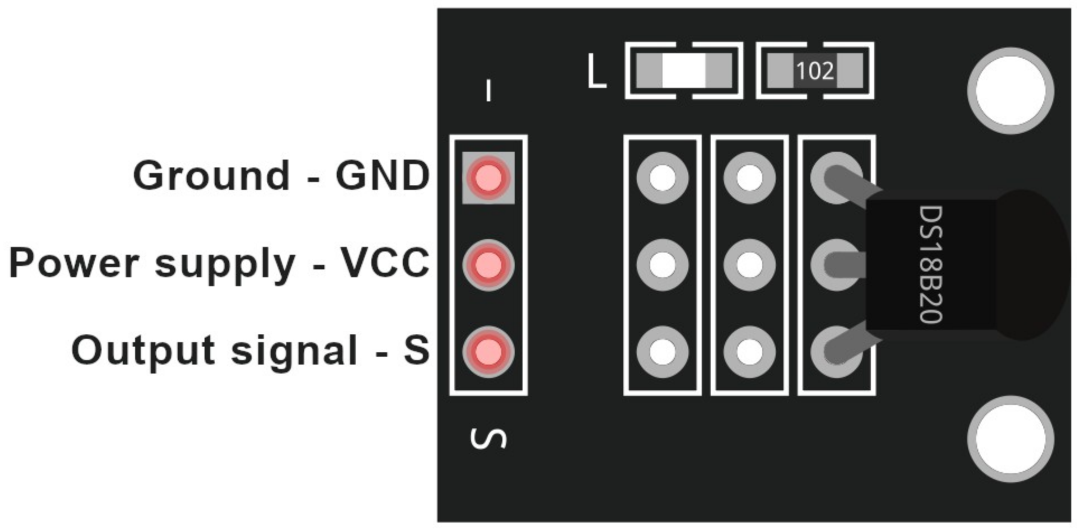
NOTE: The relevant technical data on the One-Wire interface does not mention the maximum number of sensors that can be linked on the same interface, but in practical application, this number is not as high, and you should pay attention.

NOTE: There is a cable length limitation that should be taken into consideration when using long distance communications. You should pay attention to cable distributed capacitance and resistance.

NOTE: The DS18B20 and ordinary transistors look similar, so to avoid damage, be careful not to regard it as a transistor!

The pinout

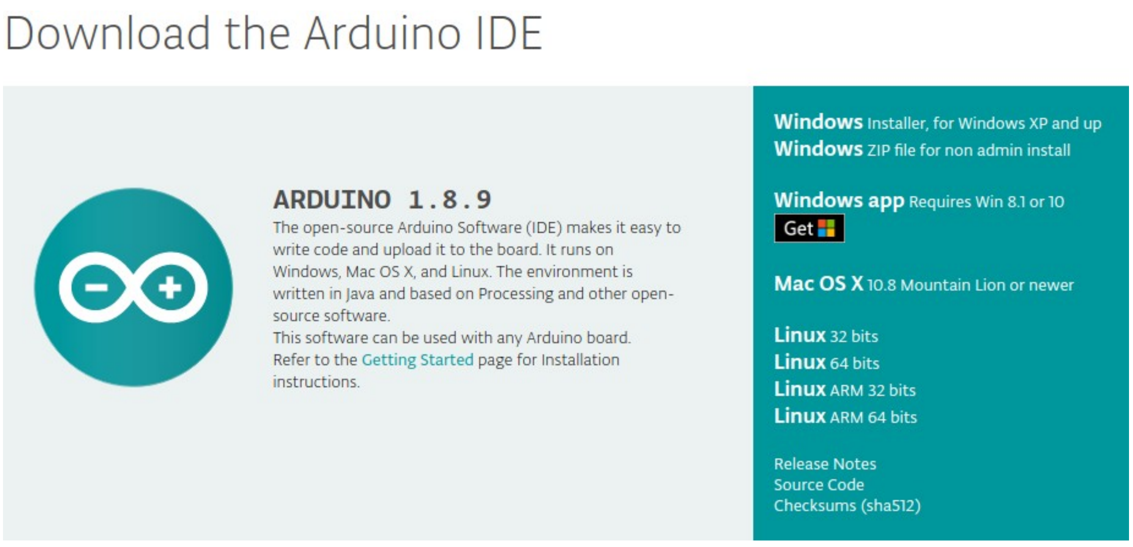
The KY-001 DS18B20 digital temperature sensor module has three pins. The pinout diagram is shown on the following image:



NOTE: If you experience any communication issues, try adding a *4.7k*Ω pull-up resistor on the *S* pin.

How to set-up Arduino IDE

If the Arduino IDE is not installed, follow the *link* and download the installation file for the operating system of choice.


Download the Arduino IDE

ARDUINO 1.8.9
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Release Notes
Source Code
Checksums (sha512)

For `Windows` users, double click on the downloaded *`.exe`* file and follow the instructions in the installation window.

For *`Linux`* users, download a file with the extension *`.tar.xz`*, which has to be extracted. When it is extracted, go to the extracted directory and open the terminal in that directory. Two *`.sh`* scripts have to be executed, the first called *`arduino-linux-setup.sh`* and the second called *`install.sh`*.

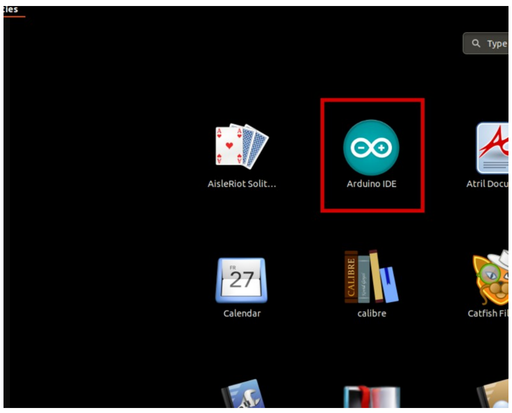
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

`sh arduino-linux-setup.sh user_name`

`user_name` - is the name of a superuser in the Linux operating system. A password for the superuser has to be entered when the command is started. Wait for a few minutes for the script to complete everything.

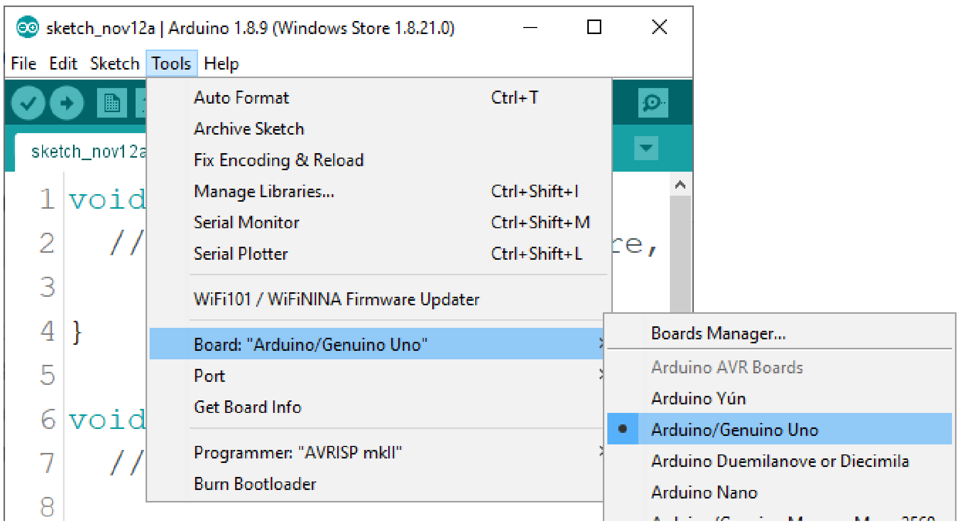
The second script called *`install.sh`* script has to be used after installation of the first script. Run the following command in the terminal (extracted directory): **`sh install.sh`**

After the installation of these scripts, go to the *`All Apps`*, where the *`Arduino IDE`* is installed.



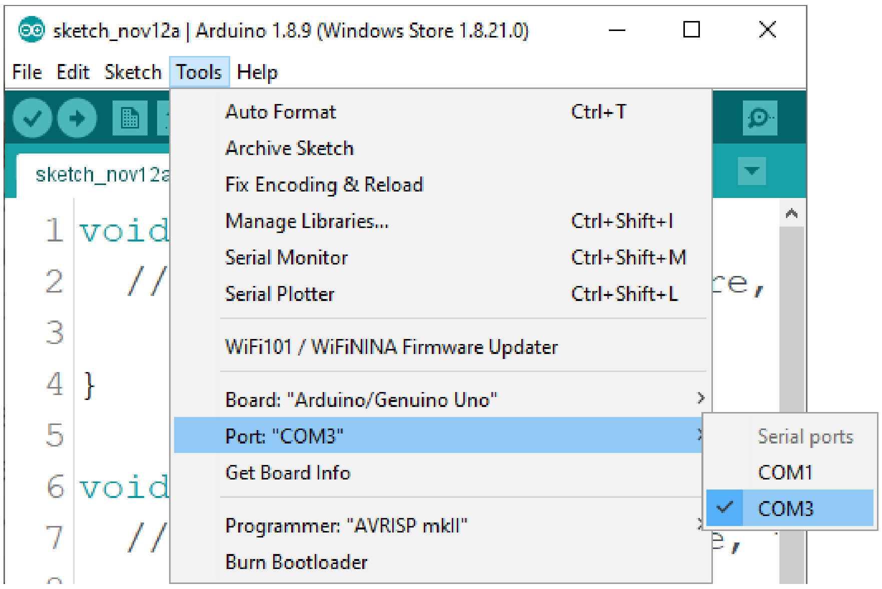
Almost all operating systems come with a text editor preinstalled (for example, *Windows* comes with *Notepad*, *Linux Ubuntu* comes with *Gedit*, *Linux Raspbian* comes with *Leafpad*, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

Next thing is to check if your PC can detect an Atmega328p board. Open freshly installed Arduino IDE, and go to:
Tools > Board > {your board name here}
{your board name here} should be the *Arduino/Genuino Uno*, as it can be seen on the following image:


sketch_nov12a | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Yún
Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano

The port to which the Atmega328p board is connected has to be selected. Go to: *Tools > Port > {port name goes here}*
and when the Atmega328p board is connected to the USB port, the port name can be seen in the drop-down menu on the previous image.

If the Arduino IDE is used on Windows, port names are as follows:


sketch_nov12a | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM3"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
COM3

For *`Linux`* users, for example port name is *`/dev/ttyUSBx`*, where *`x`* represents integer number between *0* and *9*.

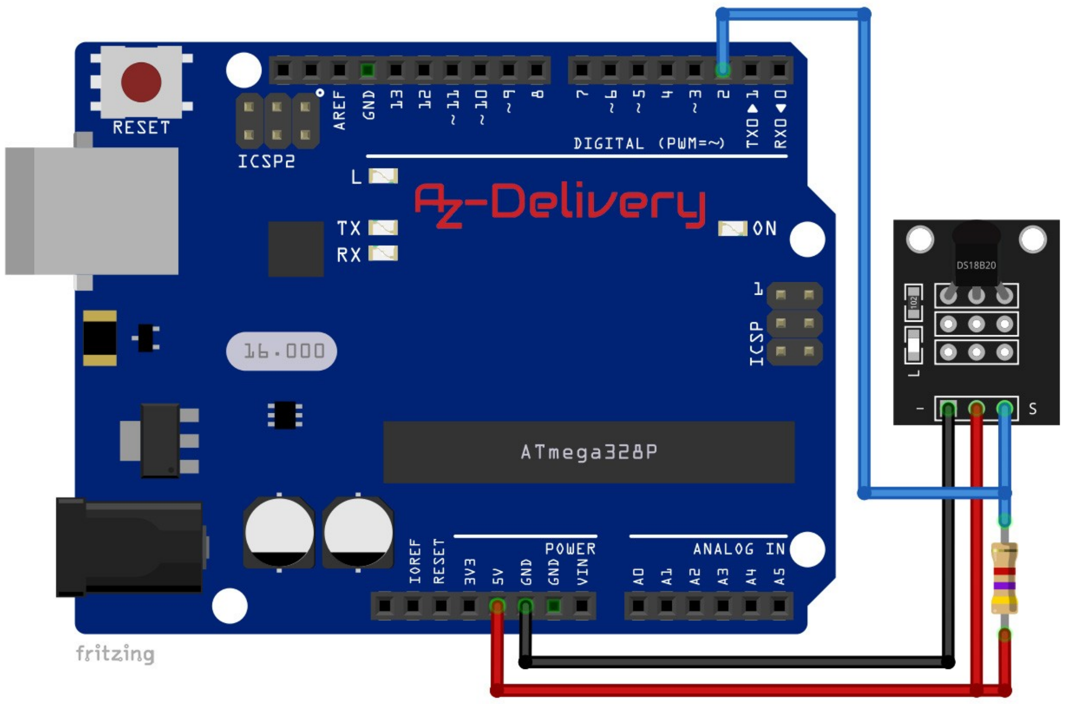
How to set-up the Raspberry Pi and Python

For the Raspberry Pi, first the operating system has to be installed, then everything has to be set-up so that it can be used in the `Headless` mode. The *Headless* mode enables remote connection to the Raspberry Pi, without the need for a *PC* screen Monitor, mouse or keyboard. The only things that are used in this mode are the Raspberry Pi itself, power supply and internet connection. All of this is explained minutely in the free eBook:
Raspberry Pi Quick Startup Guide

The *Raspbian* operating system comes with *Python* preinstalled.

Connecting the sensor with Atmega328p

Connect the KY-001 module with the Atmega328p as shown on the following connection diagram:



KY-001 pin	>	Mc pin	
OUT (S)	>	D2	Blue wire
+ (VCC)	>	5V	Red wire
- (GND)	>	GND	**Black wire**

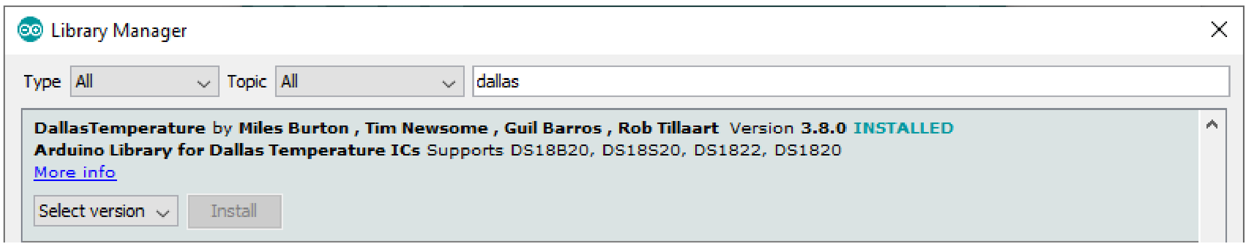
NOTE: Pull up *4.7kΩ* resistor is connected between *OUT* pin and *VCC*.

NOTE: There are a few versions of the KY-001 module, make sure to read the pin designations on the module before you connect it to the Atmega328p! We put in parentheses other names for pins that can be found on other versions of the module.

Library for Arduino IDE

To use the module with Atmega328p, first we have to download a library for it. Go to: *`Tools > Manage Libraries`*.

When a new window opens type *`Dallas`* in the search box and download the library *`DallasTemperature`* by *`Miles Burton, Tim Newsome Guil Barros and Rob Tillaart`*, as shown on the following image:



Library Manager

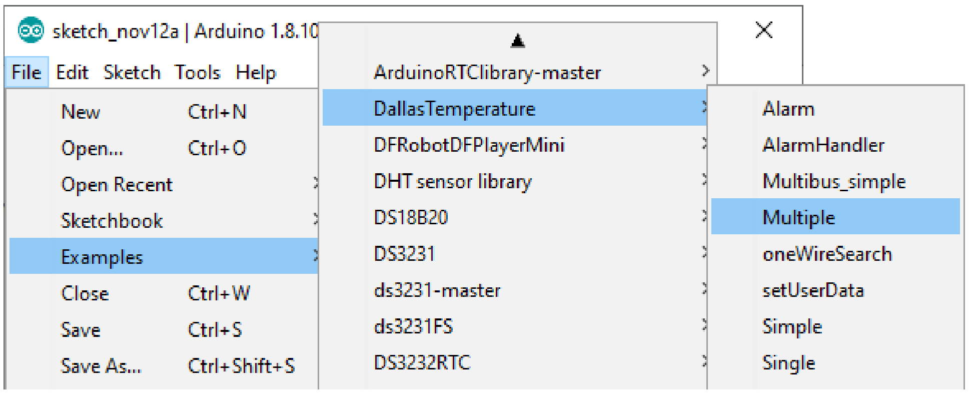
Type All Topic All dallas

DallasTemperature by **Miles Burton , Tim Newsome , Guil Barros , Rob Tillaart** Version **3.8.0** INSTALLED
Arduino Library for Dallas Temperature ICs Supports DS18B20, DS18S20, DS1822, DS1820
More info

Select version Install

Now, go to: *`File > Examples > DallasTemperature > …`*

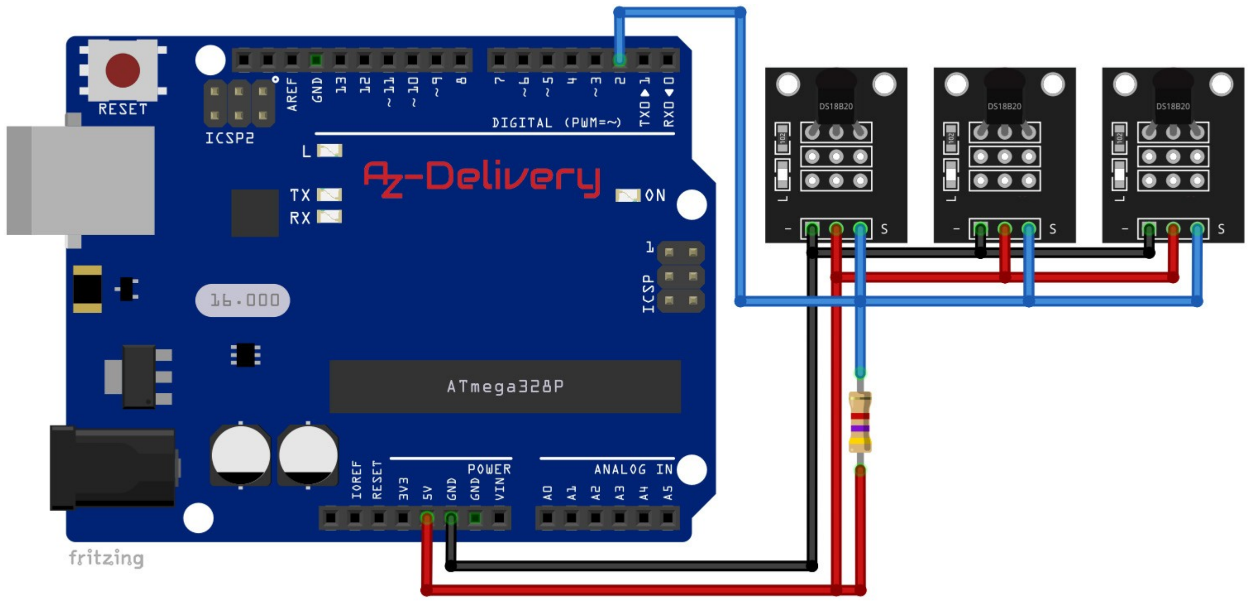
and where many sketch examples can be found. The sketch called *`Multiple`* is used and modified in this eBook, to read temperature data from three different KY-001 modules.



If only one DS18B20 sensor is used the sketch called *`Simple`* is good enough.

Connecting three KY-001 modules with Atmega328p

Connect three KY-001 modules with the Atmega328p as shown on the following connection diagram:



This connection diagram is the same as the last connection diagram, except two additional KY-001 modules that are connected to the first KY-001 module, in parallel.

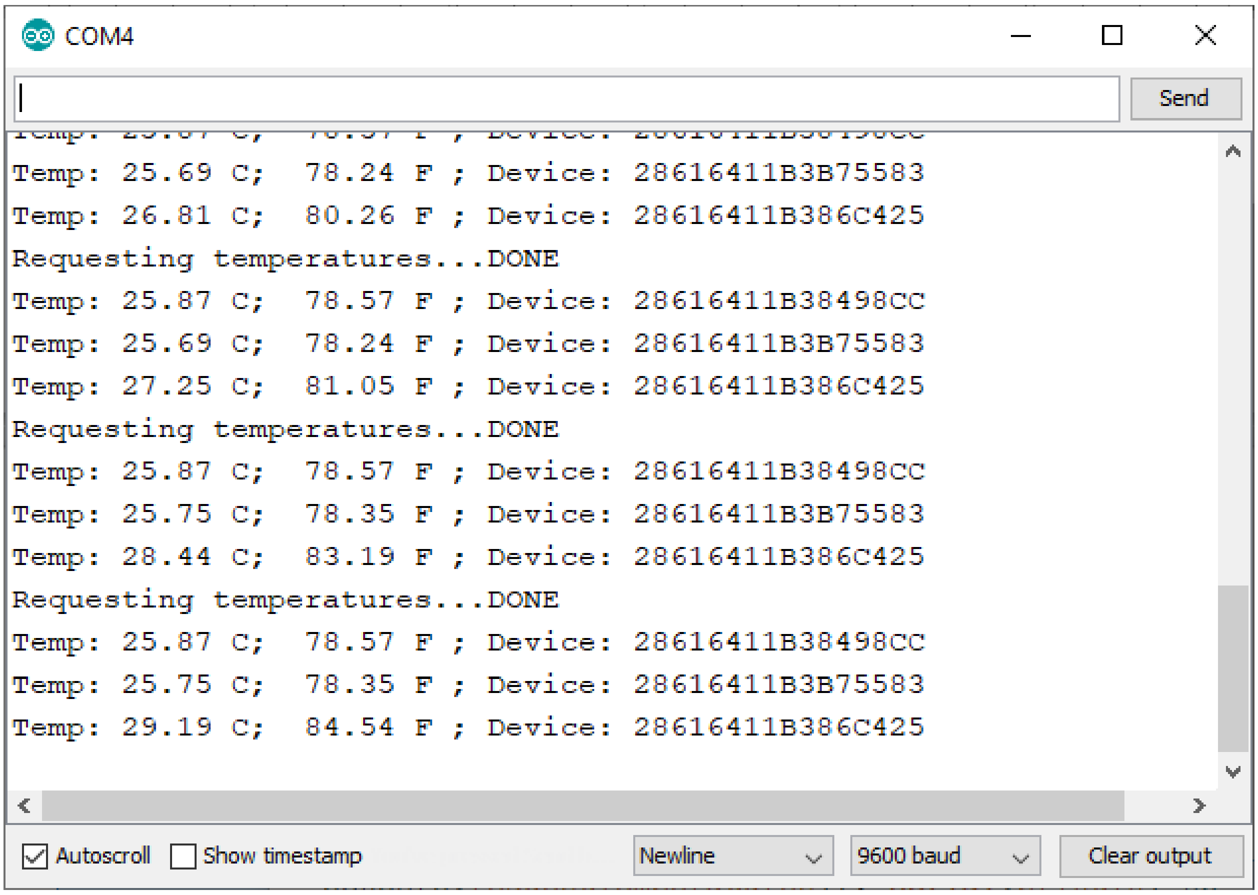
Sketch example for using three KY-001 modules

The following is the sketch example for three KY-001 modules on the same One-Wire interface:

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2 // Data wire is plugged into D2 pin
#define TEMPERATURE_PRECISION 12
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
DeviceAddress one, two, three;

void setup() {
  Serial.begin(9600);
  Serial.println("Dallas Temperature IC Control Library Demo");
  sensors.begin();
  Serial.print("Locating devices...");
  Serial.print("Found ");
  Serial.print(sensors.getDeviceCount(), DEC);
  Serial.println(" devices.");
  Serial.print("Parasite power is: ");
  if(sensors.isParasitePowerMode()) {
    Serial.println("ON");
  }
  else {
    Serial.println("OFF");
  }
  // one tab
  if(!sensors.getAddress(one, 0)) {
    Serial.println("Unable to find address for Device 0");
  }
  if(!sensors.getAddress(two, 2)) {
    Serial.println("Unable to find address for Device 2");
  }
  if(!sensors.getAddress(three, 1)) {
    Serial.println("Unable to find address for Device 1");
  }
  Serial.print("Device 0 Address: ");
  printAddress(one);
  Serial.println();
  Serial.print("Device 1 Address: ");
  printAddress(two);
  Serial.println();
  Serial.print("Device 2 Address: ");
  printAddress(three);
  Serial.println();
  sensors.setResolution(one, TEMPERATURE_PRECISION);
  sensors.setResolution(two, TEMPERATURE_PRECISION);
  sensors.setResolution(three, TEMPERATURE_PRECISION);
  Serial.print("Device 0 Resolution: ");
  Serial.print(sensors.getResolution(one), DEC);
  Serial.println();
  Serial.print("Device 1 Resolution: ");
  Serial.print(sensors.getResolution(two), DEC);
  Serial.println();
  Serial.print("Device 2 Resolution: ");
  Serial.print(sensors.getResolution(three), DEC);
  Serial.println();
}
```

```
void printAddress(DeviceAddress deviceAddress) {
  for(uint8_t i = 0; i < 8; i++) {
    if(deviceAddress[i] < 16) Serial.print("0");
    Serial.print(deviceAddress[i], HEX);
  }
}
void printTemperature(DeviceAddress deviceAddress) {
  float tempC = sensors.getTempC(deviceAddress);
  Serial.print("Temp: ");
  Serial.print(tempC);
  Serial.print(" C;  ");
  Serial.print(DallasTemperature::toFahrenheit(tempC));
  Serial.print(" F");
}
void printResolution(DeviceAddress deviceAddress) {
  Serial.print("Resolution: ");
  Serial.println(sensors.getResolution(deviceAddress));
}
void printData(DeviceAddress deviceAddress) {
  printTemperature(deviceAddress);
  Serial.print(" ; Device: ");
  printAddress(deviceAddress); Serial.println();
}
void loop() {
  Serial.print("Requesting temperatures...");
  sensors.requestTemperatures();
  Serial.println("DONE");
  printData(one);
  printData(two);
  printData(three);
  delay(1000);
}
```

Upload the sketch to the Atmega328 and open Serial Monitor (*`Tools > Serial Monitor`*). The result should look like the following image:



```
Temp: 25.69 C;  78.24 F ; Device: 28616411B3B75583
Temp: 26.81 C;  80.26 F ; Device: 28616411B386C425
Requesting temperatures...DONE
Temp: 25.87 C;  78.57 F ; Device: 28616411B38498CC
Temp: 25.69 C;  78.24 F ; Device: 28616411B3B75583
Temp: 27.25 C;  81.05 F ; Device: 28616411B386C425
Requesting temperatures...DONE
Temp: 25.87 C;  78.57 F ; Device: 28616411B38498CC
Temp: 25.75 C;  78.35 F ; Device: 28616411B3B75583
Temp: 28.44 C;  83.19 F ; Device: 28616411B386C425
Requesting temperatures...DONE
Temp: 25.87 C;  78.57 F ; Device: 28616411B38498CC
Temp: 25.75 C;  78.35 F ; Device: 28616411B3B75583
Temp: 29.19 C;  84.54 F ; Device: 28616411B386C425
```

When temperature is read by microcontroller, the red LED blinks (the on-board LED of the KY-001 module).

The variable called *ONE_WIRE_BUS* is used to define on which digital pin of the Atmega328p represents the One-Wire bus. For the purpose of this eBook, the value of the *ONE_WIRE_BUS* variable is set to *D2*, but any other digital pin of the Atmega328p can be used, except the ones used in Serial Interface, *D0* and *D1* (it is a recommendation, it can be used but these pins should be disconnected when the sketches are uploading).

The variable *TEMPERATURE_PRECISION* is used to set precision for DS18B20 sensors. Number saved in this variable is a digital conversion number in bits and it can be in a range from *9* to *12*, any other number will result in an error. For the purpose of this eBook, this value is set to the maximum value (*12*).

The following line of code: `DeviceAddress one, two, three`
is used to create variables for sensor addresses, and in the example three are created.

The object called *oneWire* is created and used for One-Wire interface:
`OneWire oneWire(ONE_WIRE_BUS);`

Next, the object *oneWire* is used to define and create the *sensors* object:
`DallasTemperature sensors(&oneWire)`
This object is used for controlling all connected sensors.

To initialize the *sensors* object the following line of code is used:

```
sensors.begin()
```

With that line of the code *sensors* object detects all sensors connected on the One-Wire interface. It also detects all the addresses of sensors.

Next, all connected sensors are checked if it work properly, by using the following lines of code for every sensor that is connect to the One-Wire interface:

```
if(!sensors.getAddress(one, 0)) {
  Serial.println("Unable to find address: Device 0"); }
```

where *one* is the address of the first sensor. For the second sensor use *two* and for the third use the *three* variable.

To set-up analog to digital conversion precision of the specific sensor the following line of code is used:

```
sensors.setResolution(one, TEMPERATURE_PRECISION)
```

If the analog to digital conversion precision of the specific sensor needs to be read, the following line of code can be used:

```
sensors.getResolution(one)
```

The function returns a hexadecimal value, and to convert it to a decimal value use the following line of code:

```
Serial.print(sensors.getResolution(one), DEC);
```

In order to read the temperature data, first all data from all sensors have to be requested, by using the following line of code:

```
sensors.requestTemperatures();
```

Only after that line of code, the data of a particular sensor can be read, using the following line of code:

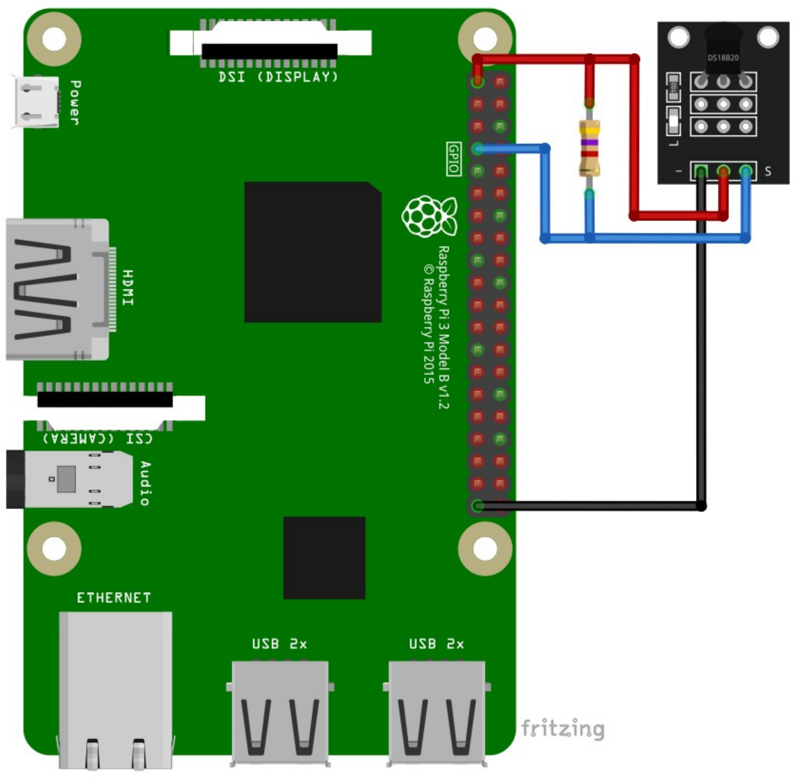
```
float tempC = sensors.getTempC(deviceAddress);
```

where the *`deviceAddress`* argument has to be passed to the function in order to read temperature data from a specific sensor. This data is temperature value in Celsius, and to convert it into Fahrenheit use the following line of code:

```
DallasTemperature::toFahrenheit(tempC)
```

Connecting the sensor with Raspberry Pi

Connect the KY-001 module with the Raspberry Pi as shown on the following connection diagram:



KY-001 pin	**>**	**Raspberry Pi pin**		
+ (VCC)	>	3V3	[pin 1]	**Red wire**
out (S)	>	GPIO4	[pin 7]	**Blue wire**
- (GND)	>	GND	[pin 39]	**Black wire**

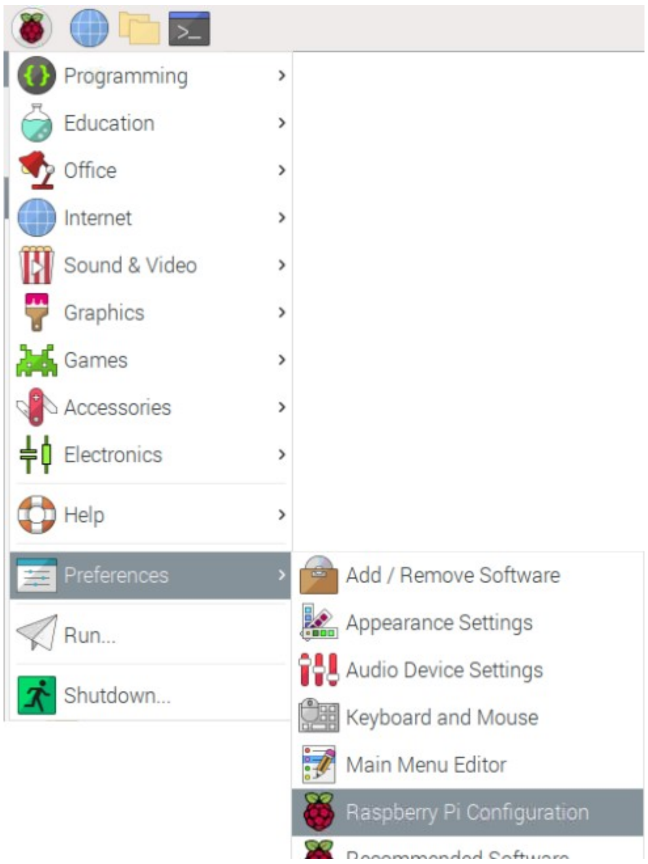
NOTE: Pull up *4.7kΩ* resistor between *OUT PIN* and *3V3* [pin 1] is **REQUIRED**!

Enabling the One-Wire interface

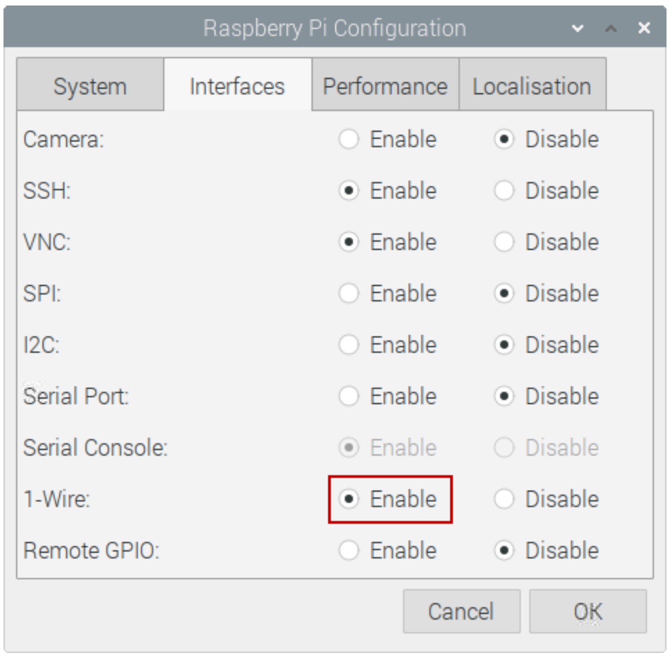
Before the DS18B20 sensor can be used on the Raspberry Pi, first the One-Wire interface has to be enabled in the Raspbian. By default, the hardware One-Wire interface is on pin GPIO4 (pin 7). In order to enable the One-Wire interface, go to:

`Applications Menu > Preferences > Raspberry Pi Configuration`

as shown on the following image:



When the new window appears, open `Interfaces` tab and search for radio buttons called *`1-Wire`* and select the *`Enable`* radio button, as shown on the following image:



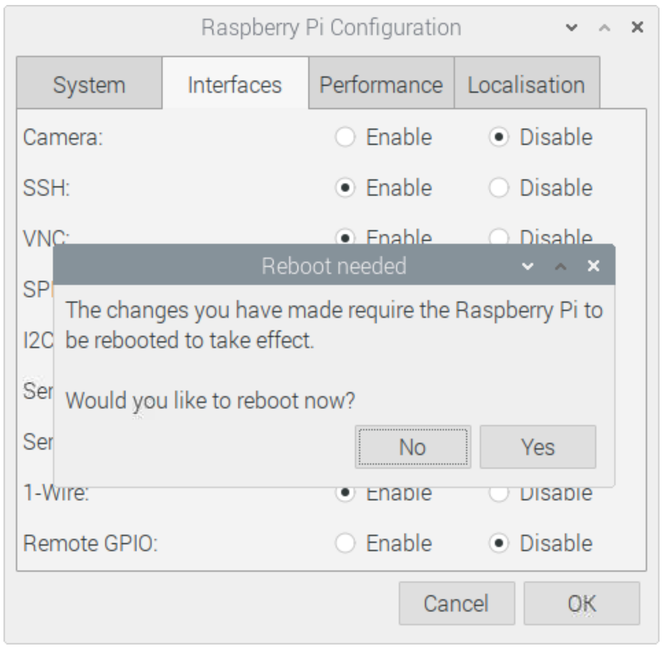
Raspberry Pi Configuration

System | Interfaces | Performance | Localisation

Camera:	○ Enable	● Disable
SSH:	● Enable	○ Disable
VNC:	● Enable	○ Disable
SPI:	○ Enable	● Disable
I2C:	○ Enable	● Disable
Serial Port:	○ Enable	● Disable
Serial Console:	● Enable	○ Disable
1-Wire:	● Enable	○ Disable
Remote GPIO:	○ Enable	● Disable

Cancel OK

Reboot the system afterwards.



Raspberry Pi Configuration

Reboot needed

The changes you have made require the Raspberry Pi to be rebooted to take effect.

Would you like to reboot now?

No Yes

When Raspbian is booted again, open the terminal, and run the following commands, one by one:

`sudo modprobe w1-gpio`

`sudo modprobe w1-therm`

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

and when you run the following command:

`ls`

the output in the terminal should be as follows:

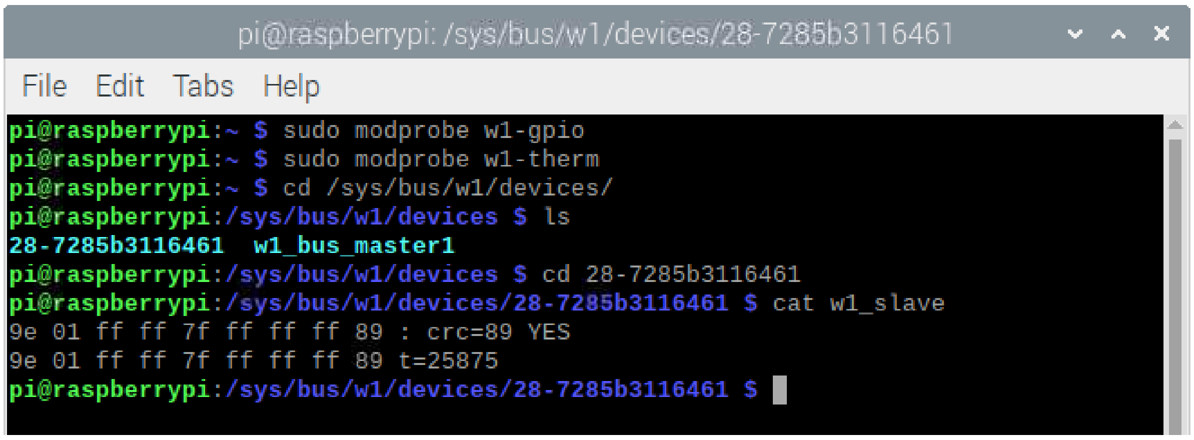
`28-7285b3116461` and **`w1_bus_master1`**

the first number *`28-7285b3116461`* is different for different sensors, because this is the serial address of the specific sensor, and each sensor has its own unique serial address. Now, to test if everything works, run these two commands:

`cd 28-7285b3116461` - a number or serial address from the last page

`cat w1_slave`

The result should look like the output on the following image:



```
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
28-7285b3116461  w1_bus_master1
pi@raspberrypi:/sys/bus/w1/devices $ cd 28-7285b3116461
pi@raspberrypi:/sys/bus/w1/devices/28-7285b3116461 $ cat w1_slave
9e 01 ff ff 7f ff ff ff 89 : crc=89 YES
9e 01 ff ff 7f ff ff ff 89 t=25875
pi@raspberrypi:/sys/bus/w1/devices/28-7285b3116461 $ 
```

`t=25875` - this is the temperature data in °C (Celsius) = 25.875°C.

Enabling multiple One-Wire interfaces

To enable the One-Wire interface, without a graphic user interface (GUI), before rebooting your Raspberry Pi, to the file located on:
/boot/config.txt
you need to add the following line:
```
dtoverlay=w1-gpio
```
or
```
dtoverlay=w1-gpio,gpiopin=x
```
where *x* is a custom pin(default is GPIO4 [pin 7], like it is mentioned in the previous chapter).

Newer kernels (*4.9.28* and later) allows using dynamic overlay loading, including creating multiple One-Wire interfaces to be used at the same time:
```
sudo dtoverlay w1-gpio gpiopin=4 pullup=0 # pin 7
sudo dtoverlay w1-gpio gpiopin=17 pullup=0 # pin 11
sudo dtoverlay w1-gpio gpiopin=27 pullup=0 # pin 13
```

Once any of the steps above have been performed, and discovery is complete the devices that the Raspberry Pi has discovered via all One-Wire interfaces can be listed by running the following command in the terminal:
```
ls /sys/bus/w1/devices/
```

NOTE: Using **`w1-gpio`** on the Raspberry Pi typically needs a *4.7kΩ* pull-up resistor connected between the GPIO pin and a *3.3V* supply.

Python script for reading multiple KY-001 modules

The code is splitted into two scripts, because of better readability. The following is a code for the class subscript:

```
import os
import glob
import time
class DS18B20:

   def __init__(self):
      os.system('modprobe w1-gpio')
      os.system('modprobe w1-therm')
      base_dir = '/sys/bus/w1/devices/'
      device_folder = glob.glob(base_dir + '28*')
      self._count_devices = len(device_folder)
      self._devices = list()
      i = 0
      while i < self._count_devices:
         self._devices.append(device_folder[i] + '/w1_slave')
         i += 1

   def device_names(self):
      names = list()
      for i in range(self._count_devices):
         names.append(self._devices[i])
         temp = names[i][20:35]
         names[i] = temp

      return names
```

```
# (one tab)
   def _read_temp(self, index):
      f = open(self._devices[index], 'r')
      lines = f.readlines()
      f.close()
      return lines

   def tempC(self, index = 0):
      lines = self._read_temp(index)
      retries = 5
      while (lines[0].strip()[-3:] != 'YES') and (retries > 0):
         time.sleep(0.1)
         lines = self._read_temp(index)
         retries -= 1

      if retries == 0:
         return 998

      equals_pos = lines[1].find('t=')
      if equals_pos != -1:
         temp = lines[1][equals_pos + 2:]
         return float(temp) / 1000
      else:
         return 999 # error

   def device_count(self):
      return self._count_devices
```

(The most of code in the script is modified from the script on the adafruit site)

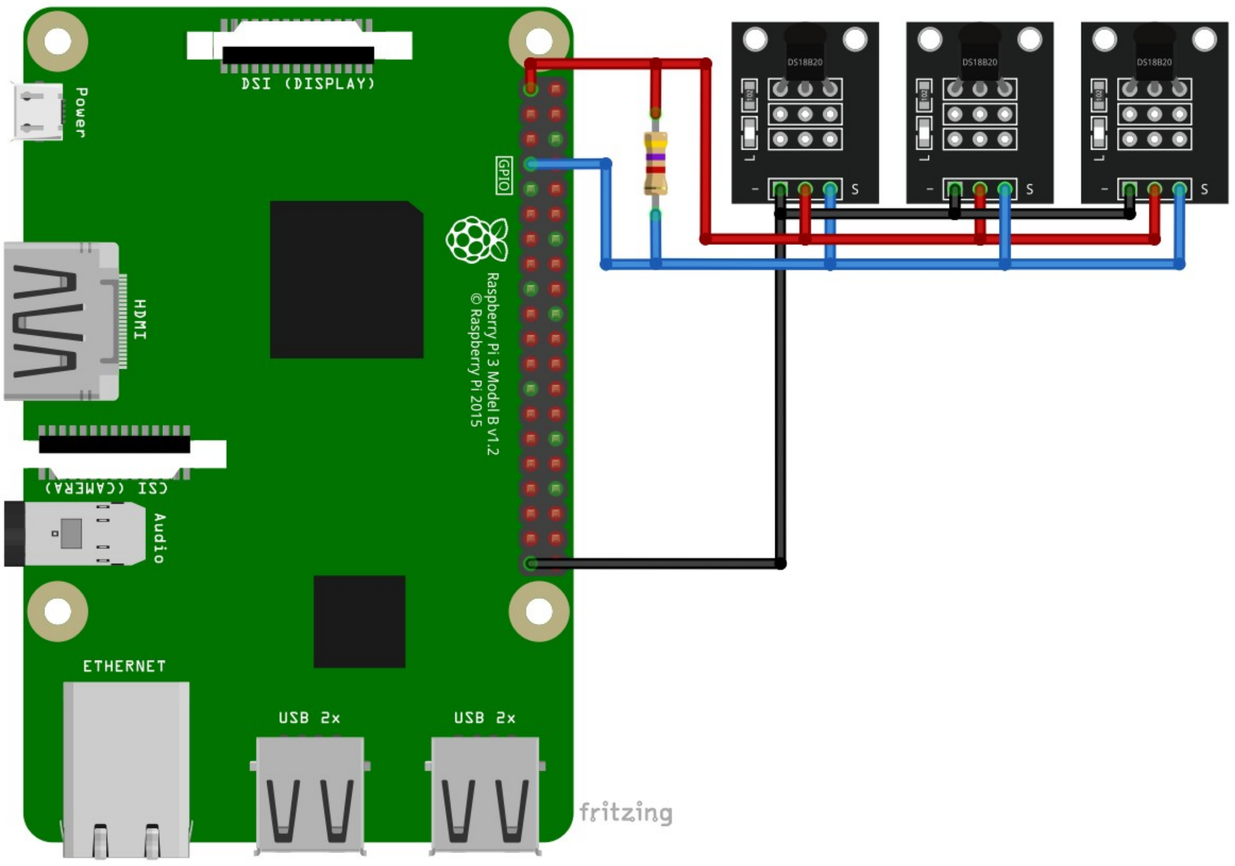
Save the script by the name *DS18B20classfile.py*.

The following is a code for the main script:

```
import time
from DS18B20classFile import DS18B20

degree_sign = u'\xb0' # degree sign
devices = DS18B20()
count = devices.device_count()
names = devices.device_names()

print('[Press CTRL + C to end the script!]')
try: # Main program loop
   while True:
      i = 0
      print('\nReading temperature, number of sensors: {}'
               .format(count))
      while i < count:
         container = devices.tempC(i)
         print('{}. Temp: {:.3f}{}C, {:.3f}{}F of the device {}'
               .format(i+1, container, degree_sign,
               container * 9.0 / 5.0 + 32.0, degree_sign,
               names[i]))
         i = i + 1

      time.sleep(1)

# Scavenging work after the end of the program
except KeyboardInterrupt:
   print('\nScript end!')
```

Save the script by the name `DS18B20multiple.py` in the same directory where the first script is saved.

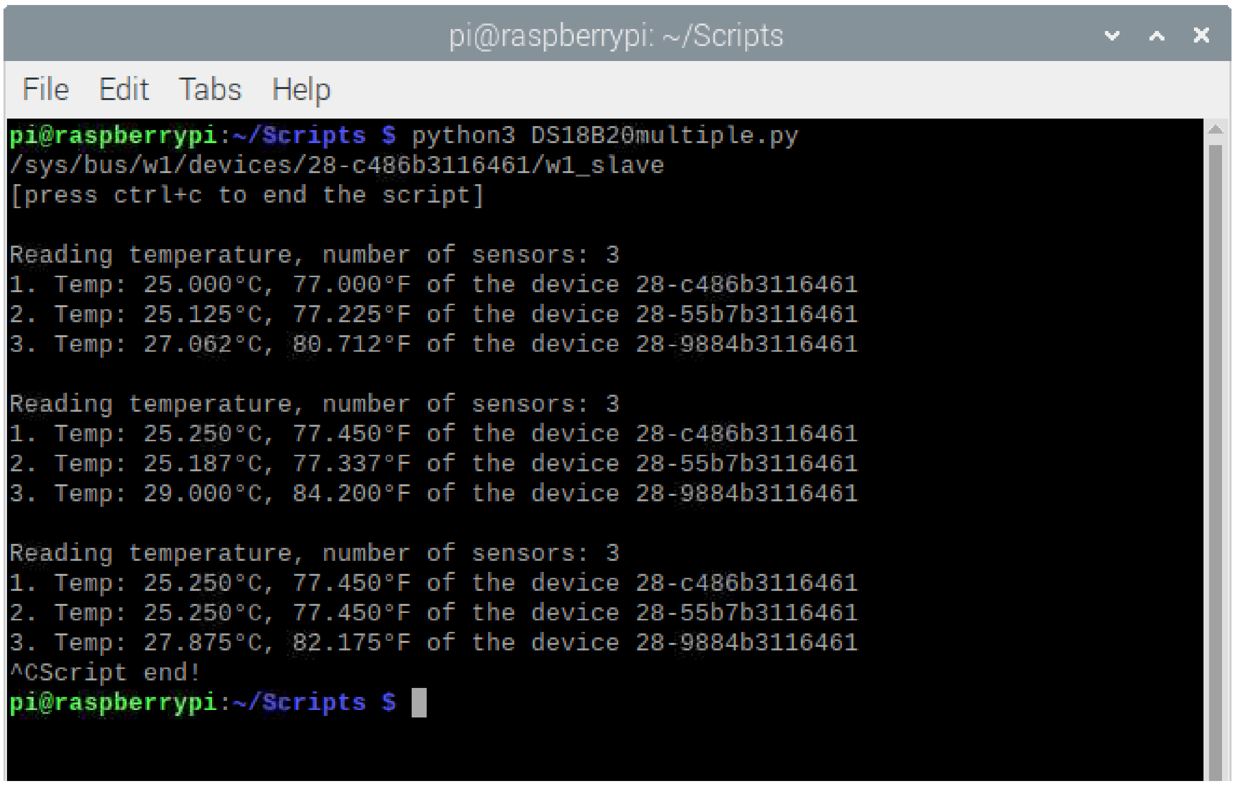
For example, three DS18B20 sensors are connected on the same One-Wire interface of the Raspberry Pi as shown on the following connection diagram:



To run the main script, open terminal in the directory where both scripts are saved, and run the following command:

```
python3 DS18B20multiple.py
```

The result should look like the output on the following image:



```
pi@raspberrypi:~/Scripts $ python3 DS18B20multiple.py
/sys/bus/w1/devices/28-c486b3116461/w1_slave
[press ctrl+c to end the script]

Reading temperature, number of sensors: 3
1. Temp: 25.000°C, 77.000°F of the device 28-c486b3116461
2. Temp: 25.125°C, 77.225°F of the device 28-55b7b3116461
3. Temp: 27.062°C, 80.712°F of the device 28-9884b3116461

Reading temperature, number of sensors: 3
1. Temp: 25.250°C, 77.450°F of the device 28-c486b3116461
2. Temp: 25.187°C, 77.337°F of the device 28-55b7b3116461
3. Temp: 29.000°C, 84.200°F of the device 28-9884b3116461

Reading temperature, number of sensors: 3
1. Temp: 25.250°C, 77.450°F of the device 28-c486b3116461
2. Temp: 25.250°C, 77.450°F of the device 28-55b7b3116461
3. Temp: 27.875°C, 82.175°F of the device 28-9884b3116461
^CScript end!
pi@raspberrypi:~/Scripts $ 
```

To stop the script press *CTRL + C* on the keyboard.

These scripts can be used for one or multiple KY-001 modules.

The class subscript will not be explained, and the main script is self explanatory.

Now it is the time to learn and make your own projects. You can do that with the help of many example scripts and other tutorials, which can be found on the internet.

If you are looking for the high quality microelectronics and accessories, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

https://az-delivery.de

Have Fun!

Impressum

https://az-delivery.de/pages/about-us