Using DS18B20 Temperature Sensors with Picon Zero

Requirements:

- Picon Zero
- DS18B20 temperature sensor – either separate chip or in waterproof tube
- 10kΩ resistor – only for version 7 firmware and earlier
- Raspberry Pi Connected to Internet, keyboard and screen
- Raspberry Pi already setup following Worksheet 01

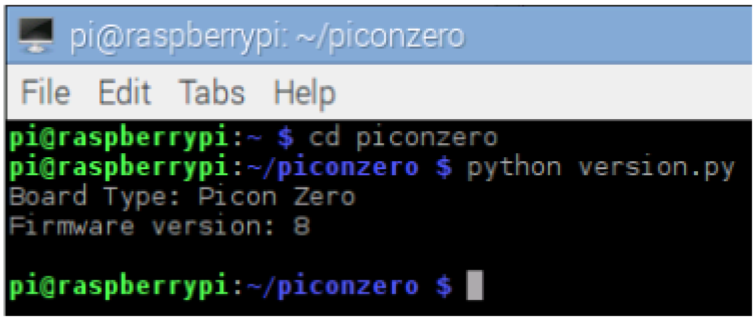
Check Your Version of Picon Zero:

The Picon Zero continues to evolve and new features are occasionally added to the firmware. One of the features not in the original version 06 release is the ability to add pullup resistors to digital inputs in the software. This was added in Firmware version 08 for both Digital Inputs and DS18B20 temperature sensors and makes the wiring much simpler.

To check which version of Picon Zero you have, change to the piconzero folder on your Raspberry Pi and enter the following command:

Python version.py

The board type (Picon Zero) and the Firmware version will be printed on the screen



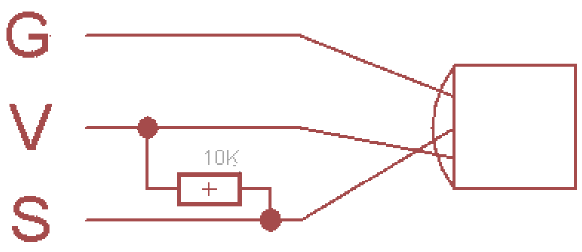
```
pi@raspberrypi:~ $ cd piconzero
pi@raspberrypi:~/piconzero $ python version.py
Board Type: Picon Zero
Firmware version: 8

pi@raspberrypi:~/piconzero $ 
```

If your version is 7 or earlier, you will need to add the pullup resistor in the following activity

Wire up the Sensor

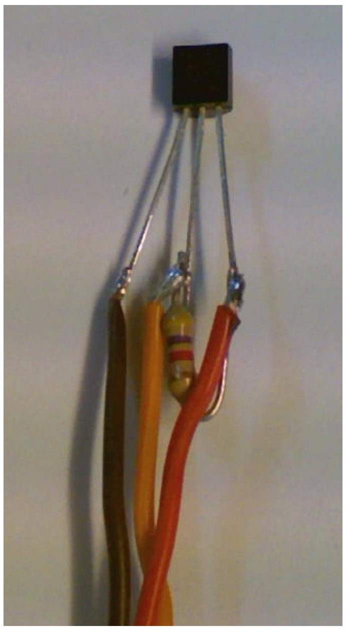
If you need the resistor (if your Picon Zero firmware is earlier than revision 8), wire it between the Signal pin and the Power pin as shown below.



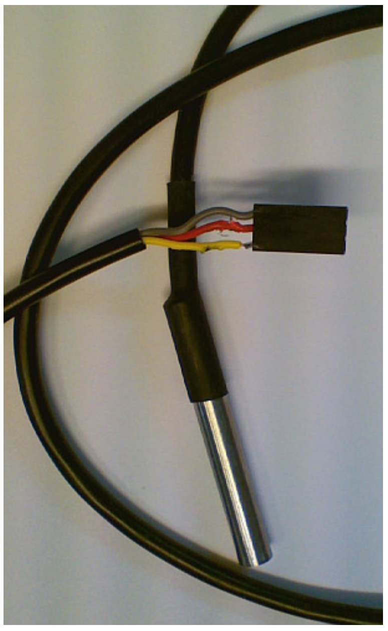
The easiest way to connect sensors, switches and other input devices to Picon Zero is to use a 3-pin "dupont" female header. You can make these up yourself, or easier to get a ready-made cable and attach your input device to the other end. These are the same cables sold as servo cables:

We stock these at 4tronix here:
http://4tronix.co.uk/store/index.php?rt=product/product&product_id=553

If using the waterproof sensors with cable, then it is best to connect directly to a 3-way female connector, although you can do it with the cable shown above if you don't have such things.



With Pullup resistor



Without Pullup resistor

What are Pullup Resistors For?

Inputs on microcontrollers are usually "floating". This means that they don't have a particular value of either On or Off – or High or Low.

If you touch a floating input, it is likely to change from Off to On, or vice versa. This is often used as a cheap sort of touch input method.

For real programming, we need a defined state for the pin, so we "pull it up" to 5V using a resistor, so that we know it is high. The resistor can generally be any value from 1K to 100K, but 10K is a normal value.

Programming the Test

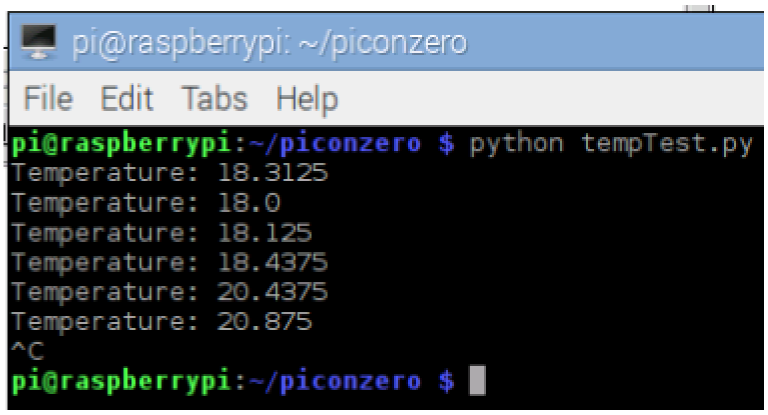
Set the input configuration to DS18B20 Input:

```
pz.setInputConfig (0, 2)
```

Then subsequent readInput( ) commands will read the temperature directly in sixteenths of a degree centigrade. So to get the number of degrees, we need to divide by 16 (or multiply by 0.625)

Here is a complete (but trivial) program to show the output (positive temperatures only)

```
import piconzero as pz, time
pz.init()
pz.setInputConfig(0, 2)   # Set input to DS18B20
while True:
    value = pz.readInput(0) # temp in centigrade
    print "Temperature:", value / 16
    time.sleep(1)
```



```
pi@raspberrypi:~/piconzero $ python tempTest.py
Temperature: 18.3125
Temperature: 18.0
Temperature: 18.125
Temperature: 18.4375
Temperature: 20.4375
Temperature: 20.875
^C
pi@raspberrypi:~/piconzero $
```

The DS18B20 can also read negative temperatures. If the value returned is greater than 32767, then actually it is a negative number, so we need to take away 65536 from the returned value, before dividing by 16. This is a consequence of using 160bit binary arithmetic. So we can correct the above program for negative temperatures, by adding the red lines below, after the readInput( )

```
import piconzero as pz, time
pz.init()
pz.setInputConfig(0, 2)   # Set input to DS18B20
while True:
    value = pz.readInput(0) # temp in centigrade
    if (value > 32767):  # correct for negative numbers
        value -= 65536
    print "Temperature:", value / 16
    time.sleep(1)
```