# CamJam EduKit Sensors Worksheet Three

| | |
|---|---|
| **Project** | Temperature Sensor |
| **Description** | In this project, you will learn how to use a temperature sensor. *NOTE: This worksheet can use a Raspberry Pi Model A, B, A+, B+ or Pi2. The first 26 pins on the A+/B+/Pi2 (when looking at the Pi with the pins in the top left) are exactly the same as the Model A and Model B.* |

## Equipment Required

| | | |
|---|---|---|
| □ Raspberry Pi & SD card | □ Power supply | □ 1 x 4.7k Ω resistor |
| □ Keyboard & Mouse | □ 400 Point Breadboard | □ 3 x m/f jumper wires |
| □ Monitor & HDMI Cable | □ Temperature Sensor (DS18B20) | □ 3 x m/m jumper wire |

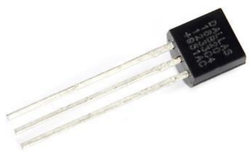
## The Parts

In this circuit you will be connecting a temperature sensor to the GPIO header of your Raspberry Pi and using Python to measure the temperature where you put the sensor. The sensor supplied in this kit is on the end of a long wire and is waterproof, which will allow you to easily measure the temperature of the room, outside a window, or even a cup of water. **Remember that electronics and water do not mix well, so keep the water away from the rest of the kit and the Raspberry Pi!**

Before you build the circuit, look at the additional parts you are going to use.

### The Temperature Sensor

The sensor supplied in the kit is a 'Dallas DS18B20' sealed into a metal tube and extended with wires, and looks like the photo on the left. The sensor inside the tube looks like the device on the right.



The sensor has a '1-wire serial' interface, which means that it sends digital messages through its output pin to the Raspberry Pi. The Pi reads these messages and puts them in a 'device file', which is like a text file. You can read this file just as you would any other text file, although you cannot edit it.

When the Raspberry Pi has received a good message from the sensor, two lines will appear in the 'device file'. The first one will end in 'YES', and the second one will end in a 't=xxxxx', where 'xxxxx' is the temperature in 1/1000th of a degree Celsius. For example:

```
a3 01 4b 46 7f ff 0e 10 d8 : crc=d8 YES
a3 01 4b 46 7f ff 0e 10 d8 t=32768
```

Which means that the temperature is 32.8°C.

## The Parts

The sensor has three wires (or legs); the black one is 'ground' (GND), the red one for the power supply (3.3v) and a white or yellow one is the output from the sensor. They may come unstripped or with very short wires. Strip off about 5mm of the coloured plastic to expose the wire, and twist those wires together. If you are confident with a soldering iron, you may want to coat the wires with solder to make them easier to insert onto the breadboard. Alternatively, you could solder small (10mm) lengths of solid wire onto the sensor's wires.

You may also need to strip off some of the black outer cover to lengthen the smaller wires.

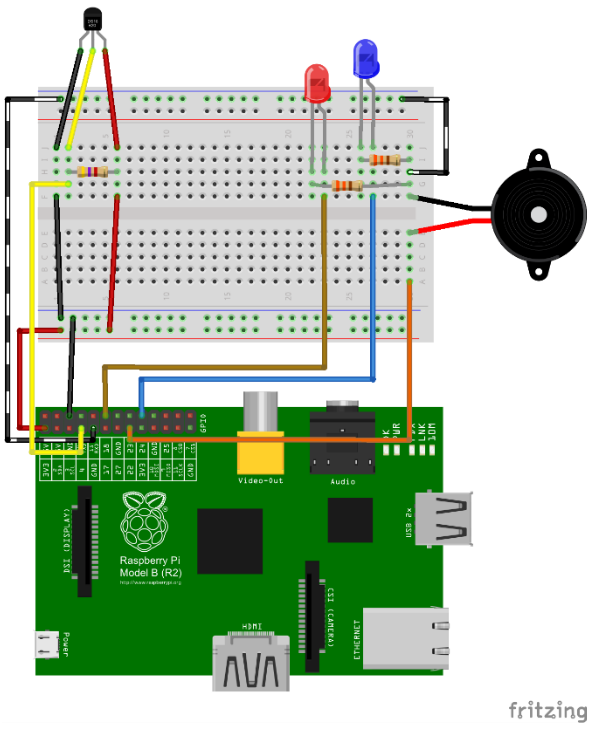
### The Resistor

The additional resistor used in this circuit is the 4.7kΩ (or 4700Ω) resistor. You can identify the 4.7kΩ resistor by the colour bands along the body. There will be either four or five colour bands on the resistor:

- If there are four bands, the colours will be Yellow, Purple, Red, and then Gold.
- If there are five bands, the colours will be Yellow, Purple, Black, Brown, Brown.

The resistor is used as a 'pull-up' for the data-line, and is required to keep the data transfer stable by supplying power to the signal circuit.

## Building the Circuit



**Before building this circuit, you must turn the Raspberry Pi off.**

You should leave the LED and buzzer circuit connected on the breadboard, and add this new circuit to the other end.

The circuit will be using another 'ground' (GND) pin to act like the 'negative' or 0 volt ends of a battery. One of the pins marked 3v3 will provide the power for the sensor. 3v3 means that it is a 3.3 volt power supply.

Use two female to male jumper wires to connect the GND and 3v3 GPIO pins to the bottom two rows of holes on the breadboard. Match up the colours marked on the breadboard - red and blue - with the jumper wires from the Pi – connect 3v3 to the red row, and GND to the blue row. These two 'rails' (as they are known) will provide the ground and power supply for the whole of the breadboard.

Connect the temperature sensor as shown, with a male/male jumper wire going to the bottom 'rail' attached to the Pi's ground (GND). Connect the red wire using a jumper to the 3v3 'rail' at the bottom. This supplies the temperature sensor with its power.

## Building the Circuit

If you have problems pressing the wire strands into the breadboard holes, use the leg of a resistor or jumper to guide them into place.

The yellow lead goes into a column with one end of the 4.7k Ω resistor and another jumper wire (shown in yellow) that goes to GPIO pin 4. The program will read the temperature from this pin.

The other end of the resistor should be inserted into another column of the breadboard, between the red lead of the temperature sensor and the jumper wire connected to the 3v3 'rail'.

## Code

Follow the instructions in Worksheet One to turn on your Pi.

Before the temperature sensor can be used, the Raspberry Pi has to be set up to recognise and use the temperature sensor. Run the following commands before running the code. These commands add additional modules to the operating system on your Raspberry Pi. You only need to do this once to check that the module can be loaded.

Open a Terminal Window:

- Enter the following command:
  ```
  sudo modprobe w1-gpio
  ```
- Followed by:
  ```
  sudo modprobe w1-therm
  ```
- Go to the 'devices' directory:
  ```
  cd /sys/bus/w1/devices
  ```
- View the devices, and check that you see "`28-00000626cc81`" (the number following "`28-`" may vary). If you see the number, then the changes have taken affect and the sensor should now work:
  ```
  ls -l
  ```

Example listing:

```
total 0
lrwxrwxrwx 1 root root 0 Jan 31 20:34 28-00000626cc81 ->
../../../devices/w1_bus_master1/28-000004d50803
lrwxrwxrwx 1 root root 0 Jan 31 20:34 w1 bus master1 ->
../../../devices/w1_bus_master1
```

The operating system will now periodically take readings from the temperature sensor, which you can view by looking at the contents of the 'device' as though it is a file. The contents of the 'file' will be text, which will need to be interpreted and converted into temperature values.

**NOTE:** If you do not see a listing similar to above, or the listing returns nothing, then is it is likely that you have the latest version of Raspbian (the operating system). This requires a little more set-up. Please return to Worksheet One and follow the instructions for updating Raspbian and configuring the 1-wire interface.

## Code

Now it is time to write the code:

1. Change directory to the directory we created in Worksheet One using:
```
cd ~/EduKitSensors/
```
2. Create a new text file "3-temperature.py" by typing the following:
```
nano 3-temperature.py
```

3. Type in the following code:
```
# Import Libraries
import os
import glob
import time

# Initialize the GPIO Pins
os.system('modprobe w1-gpio')  # Turns on the GPIO module
os.system('modprobe w1-therm') # Turns on the Temperature module

# Finds the correct device file that holds the temperature data
base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

# A function that reads the sensors data
def read_temp_raw():
  f = open(device_file, 'r') # Opens the temperature device file
  lines = f.readlines() # Returns the text
  f.close()
  return lines

# Convert the value of the sensor into a temperature
def read_temp():
  lines = read_temp_raw() # Read the temperature 'device file'

  # While the first line does not contain 'YES', wait for 0.2s
  # and then read the device file again.
  while lines[0].strip()[-3:] != 'YES':
    time.sleep(0.2)
    lines = read_temp_raw()

  # Look for the position of the '=' in the second line of the
  # device file.
  equals_pos = lines[1].find('t=')

  # If the '=' is found, convert the rest of the line after the
  # '=' into degrees Celsius, then degrees Fahrenheit
  if equals_pos != -1:
    temp_string = lines[1][equals_pos+2:]
```

## Code

```
    temp_c = float(temp_string) / 1000.0
    temp_f = temp_c * 9.0 / 5.0 + 32.0
    return temp_c, temp_f

# Print out the temperature until the program is stopped.
while True:
  print(read_temp())
  time.sleep(1)
```

Once complete use “Ctrl + x” then “y” then “enter” to save the file.

## Running the Code

To run this code type:

```
sudo python 3-temperature.py
```

The temperature readings will be printed out until the program is stopped (by pressing Ctrl+C).

## Challenge One

Measure the temperature of a glass of cold water by putting the silver end of the temperature probe into a glass of water.

Try this again with a glass of hot water, and watch the temperature change as it cools down (maybe by adding in cold water).

## Challenge Two

Alter the code to light the LEDs and sound the buzzer under the following conditions:

- Light the blue LED when the temperature is near 0°C - the sensor is accurate to 0.5°C, so light the blue LED when the temperature is at or below 0.5°C.
- Light the red LED when the temperature is above 50°C.
- Sound the buzzer when the temperature is above 75°C.