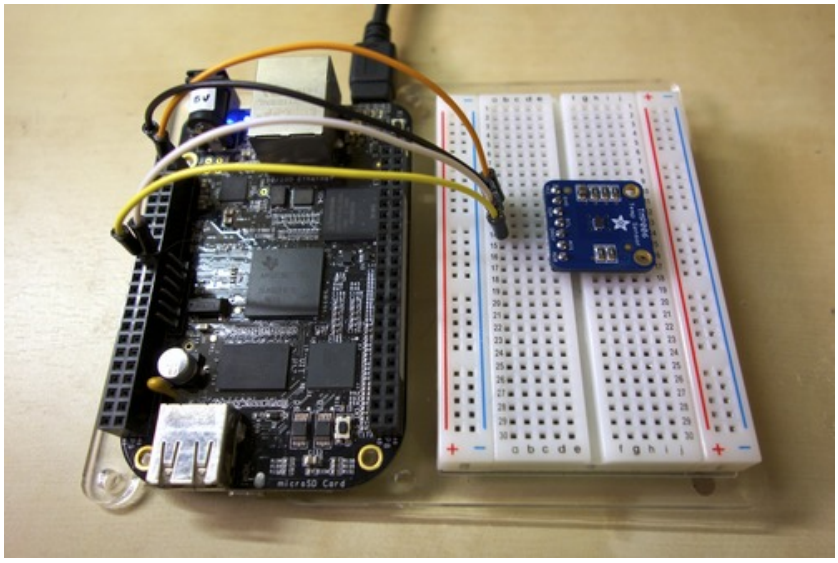
adafruit learning system

TMP006 Temperature Sensor Python Library

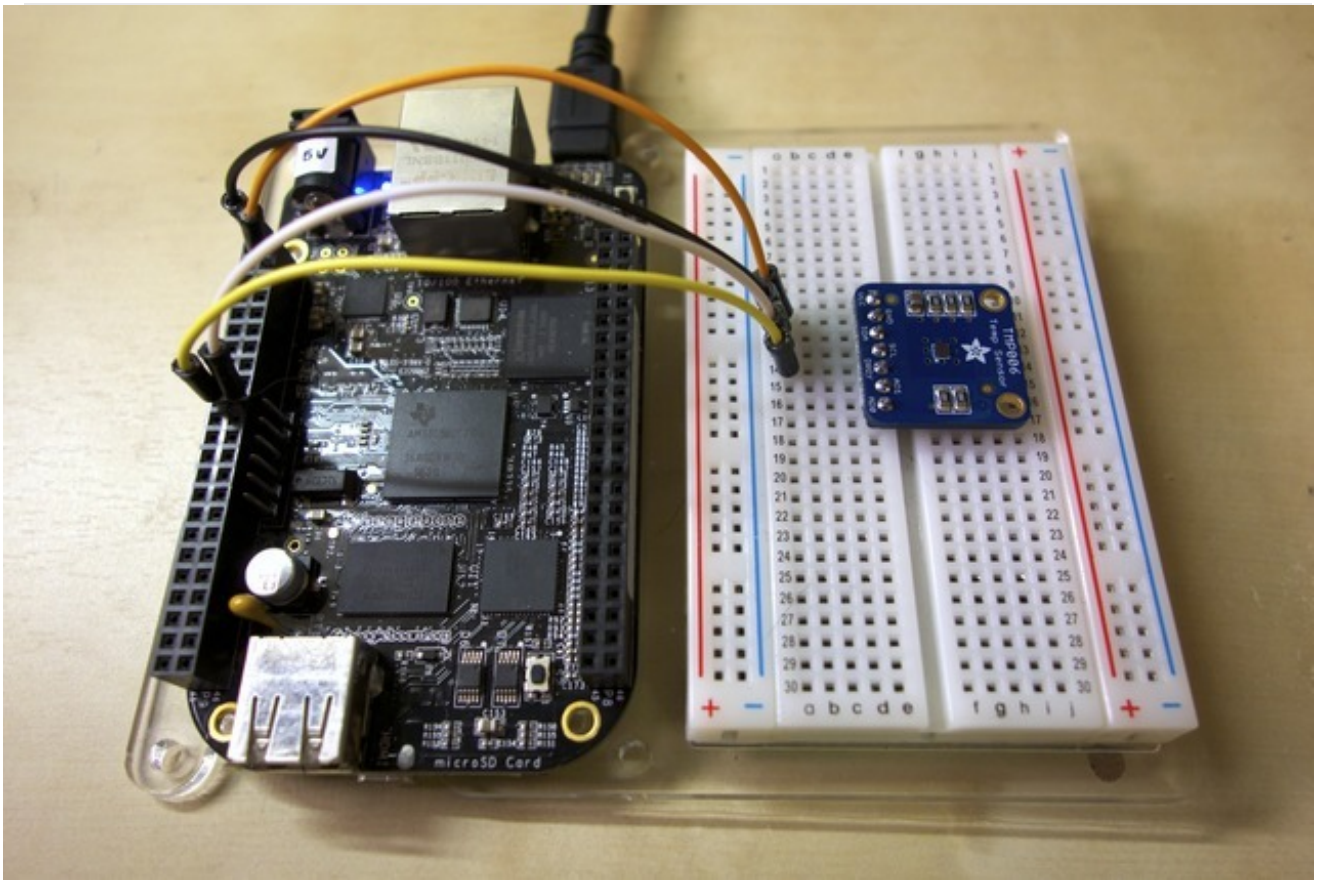
Created by Tony DiCola



Last updated on 2014-09-19 06:00:13 PM EDT

Guide Contents

Overview



Are you looking for an easy way to measure the temperature of something without having to attach a sensor directly to it? Consider using a non-contact temperature sensor like the TMP006 (http://adafru.it/1296)! This sensor reads the infra-red radiation, or heat, emitted from an object and can be easily read over an I2C connection. With the TMP006 Python library (http://adafru.it/dZV) you can now use the TMP006 non-contact temperature sensor with your Raspberry Pi or BeagleBone Black project!

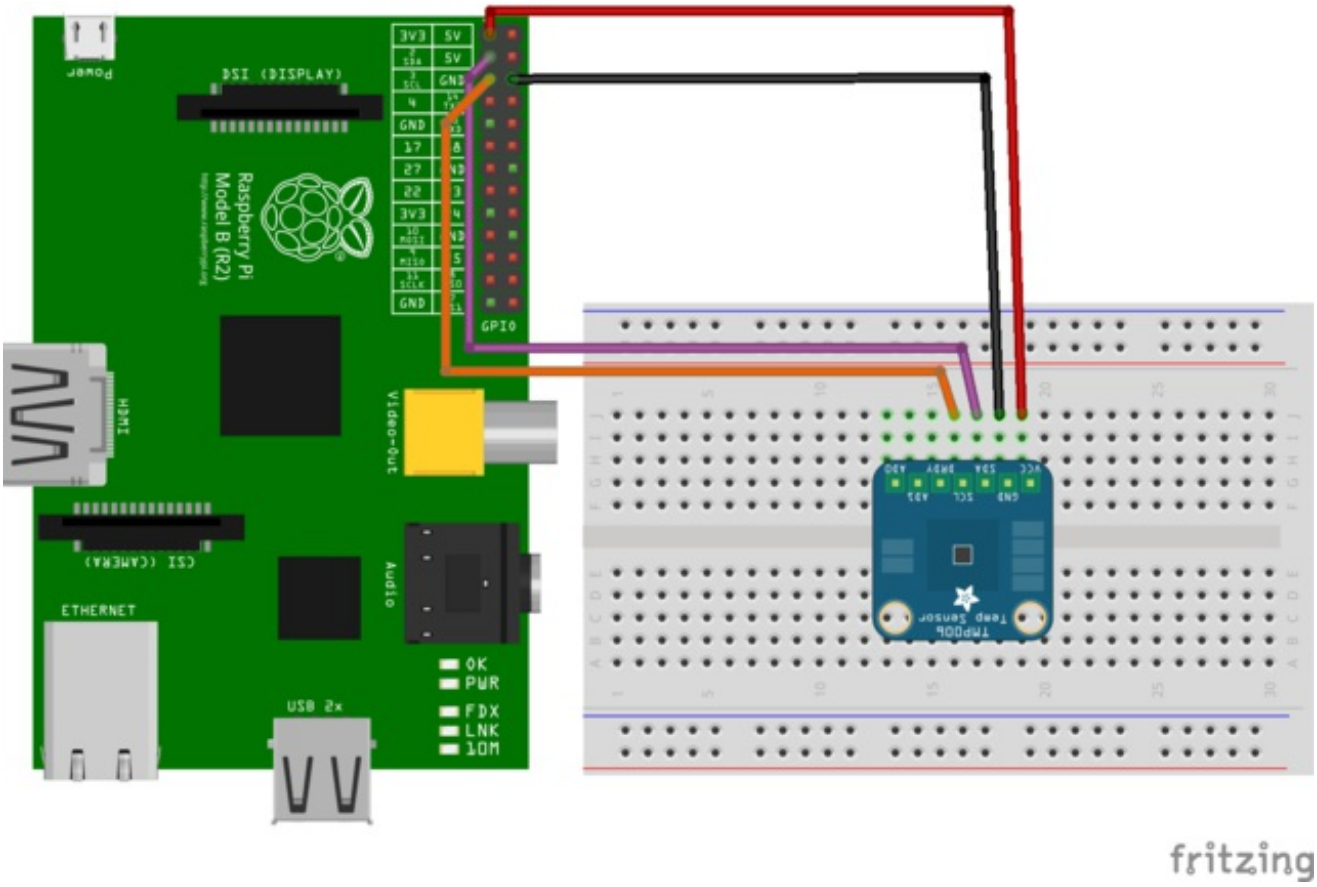
Before you get started make sure your Raspberry Pi is running the latest Raspbian (http://adafru.it/dpb) or Occidentalis (http://adafru.it/dZW) operating system, and your BeagleBone Black is running the latest official Debian operating system (http://adafru.it/dUl). It will also help to familiarize yourself with the TMP006 sensor by reading its Arduino guide (http://adafru.it/dZX).

Hardware

Wiring the TMP006 to a Raspberry Pi or BeagleBone Black is easy because the board only uses an I2C bus for communication.

Raspberry Pi

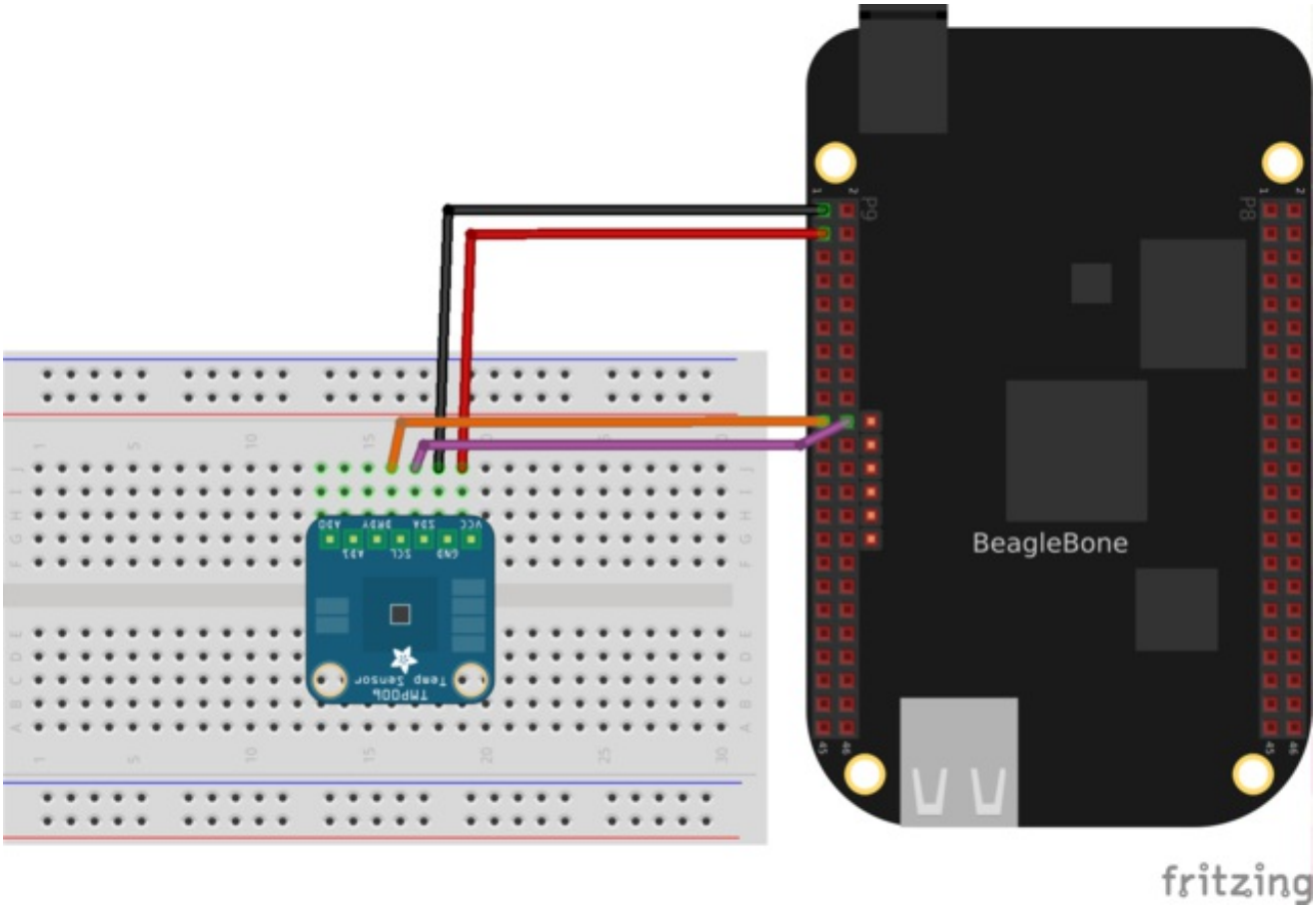
Connect the TMP006 to a Raspberry Pi as follows. Note that you must have I2C enabled on your Raspberry Pi (http://adafru.it/dZY).



- Connect **Pi 3.3V** power to **TMP006 VCC**.
- Connect **Pi GND** to **TMP006 GND**.
- Connect **Pi SDA** to **TMP006 SDA**.
- Connect **Pi SCL** to **TMP006 SCL**.

BeagleBone Black

Connect the TMP006 to a BeagleBone Black as follows. If you aren't familiar with how GPIO pins are numbered, check out this guide on BeagleBone Black GPIO (http://adafru.it/dZZ).



- Connect **BeagleBone Black P9_1 DGND** to **TMP006 GND**.
- Connect **BeagleBone Black P9_3 3.3V** power to **TMP006 VCC**.
- Connect **BeagleBone Black P9_19 SCL** to **TMP006 SCL**.
- Connect **BeagleBone Black P9_20 SDA** to **TMP006 SDA**.

Software

To install and use the TMP006 Python library (http://adafru.it/dZV) follow the steps below.

Before you get started make sure your board is connected to the internet through an ethernet or wireless connection so you can download dependencies.

You'll also want to be familiar with connecting to a Raspberry Pi (http://adafru.it/dZL) or BeagleBone Black (http://adafru.it/dUr) terminal with SSH.

Dependencies

First install dependencies by executing in a terminal:

```
sudo apt-get update
sudo apt-get install build-essential python-dev python-pip python-smbus git
```

You can ignore warnings about dependencies which are already installed.

Raspberry Pi

On a Raspberry Pi execute the following to make sure the RPi.GPIO library is installed:

```
sudo pip install RPi.GPIO
```

BeagleBone Black

On a BeagleBone Black execute the following to make sure the Adafruit_BBIO library is installed:

```
sudo pip install Adafruit_BBIO
```

Library Install

Next download the TMP006 Python library to your home directory and install it by executing:

```
cd ~
git clone https://github.com/adafruit/Adafruit_Python_TMP.git
cd Adafruit_Python_TMP
sudo python setup.py install
```

That's all you need to do to install the Python library!

Usage

To learn how to use the TMP006 Python library you can run and review an example program included with the library. To run the example navigate to the **examples** directory and run the **simpletest.py** script by executing:

```
cd examples
sudo python simpletest.py
```

If everything goes well you should see the sensor's object and die temperature displayed every second:

```
Press Ctrl-C to quit.
Object temperature: 25.368*C / 77.662*F
  Die temperature: 23.094*C / 73.569*F
Object temperature: 25.368*C / 77.662*F
  Die temperature: 23.094*C / 73.569*F
Object temperature: 26.763*C / 80.174*F
  Die temperature: 23.094*C / 73.569*F
...
```

If you see an error make sure you're running the program as root with the sudo command, and that the dependencies & library were installed successfully.

To understand the usage, open **simpletest.py** in a text editor and follow along with the description of the code below.

```
import Adafruit_TMP.TMP006 as TMP006
```

First the TMP006 module is imported with a Python import statement.

```
# Default constructor will use the default I2C address (0x40) and pick a default I2C bus.
#
# For the Raspberry Pi this means you should hook up to the only exposed I2C bus
# from the main GPIO header and the library will figure out the bus number based
# on the Pi's revision.
#
# For the Beaglebone Black the library will assume bus 1 by default, which is
# exposed with SCL = P9_19 and SDA = P9_20.
sensor = TMP006.TMP006()
```

```
# Optionally you can override the address and/or bus number:
#sensor = TMP006.TMP006(address=0x42, busnum=2)
```

Next an instance of the TMP006 class is created.

Notice that if nothing is passed in to the initializer function the library will pick a default I2C device address (0x40) and bus number. If you need to specify the I2C address or bus number you can do so by specifying them as optional parameters in the TMP006 initializer.

```
# Initialize communication with the sensor, using the default 16 samples per conversion.
# This is the best accuracy but a little slower at reacting to changes.
sensor.begin()

# Optionally initialize with a faster but less precise sample rate.  You can use
# any value from TMP006_CFG_1SAMPLE, TMP006_CFG_2SAMPLE, TMP006_CFG_4SAMPLE,
# TMP006_CFG_8SAMPLE, or TMP006_CFG_16SAMPLE for the sample rate.
#sensor.begin(samplerate=TMP006.CFG_1SAMPLE)
```

After creating the TMP006 class instance the **begin()** function is called to initialize communication with the device.

Notice that you can specify an optional **samplerate** parameter which specifies how many samples will be taken to form a temperature reading. The more samples used the more accurate your reading, however there will be a trade-off in the speed of measurements.

```
# Loop printing measurements every second.
print 'Press Ctrl-C to quit.'
while True:
 obj_temp = sensor.readObjTempC()
 die_temp = sensor.readDieTempC()
 print 'Object temperature: {0:0.3F}*C / {1:0.3F}*F'.format(obj_temp, c_to_f(obj_temp))
 print '    Die temperature: {0:0.3F}*C / {1:0.3F}*F'.format(die_temp, c_to_f(die_temp))
 time.sleep(1.0)
```

Finally the example enters a loop where it reads temperature measurements and prints them out every second. The important thing to see are the two temperature measurement functions, **readObjTempC()** and **readDieTempC()**.

The **readObjTempC()** function will read the object temperature and return its value in Celsius. Make sure to skim the TMP006 user guide (http://adafru.it/e00) to understand how the TMP006 chip measures object temperature.

The **readDieTempC()** function will read the TMP006 die temperature and return its value in Celsius. Note that this value is the temperature of the chip's die and not the temperature of the object.

That's all there is to use the TMP006 Python library! If you run into issues or wish to contribute, feel free to followup on the library's Github page (http://adafru.it/dZV).