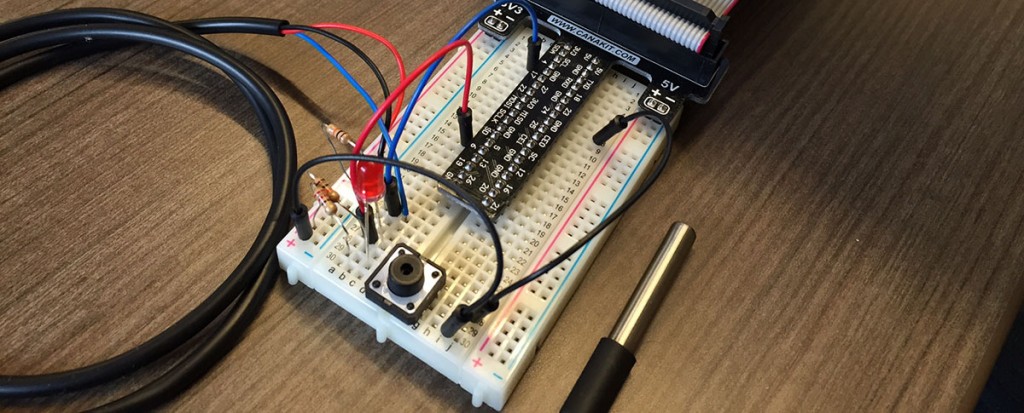
[Stream Temperature from your Raspberry Pi](http://blog.initialstate.com/stream-temperature-from-pi/)

[](http://blog.initialstate.com/wp-content/uploads/2014/12/PiTemp18B20.jpg)

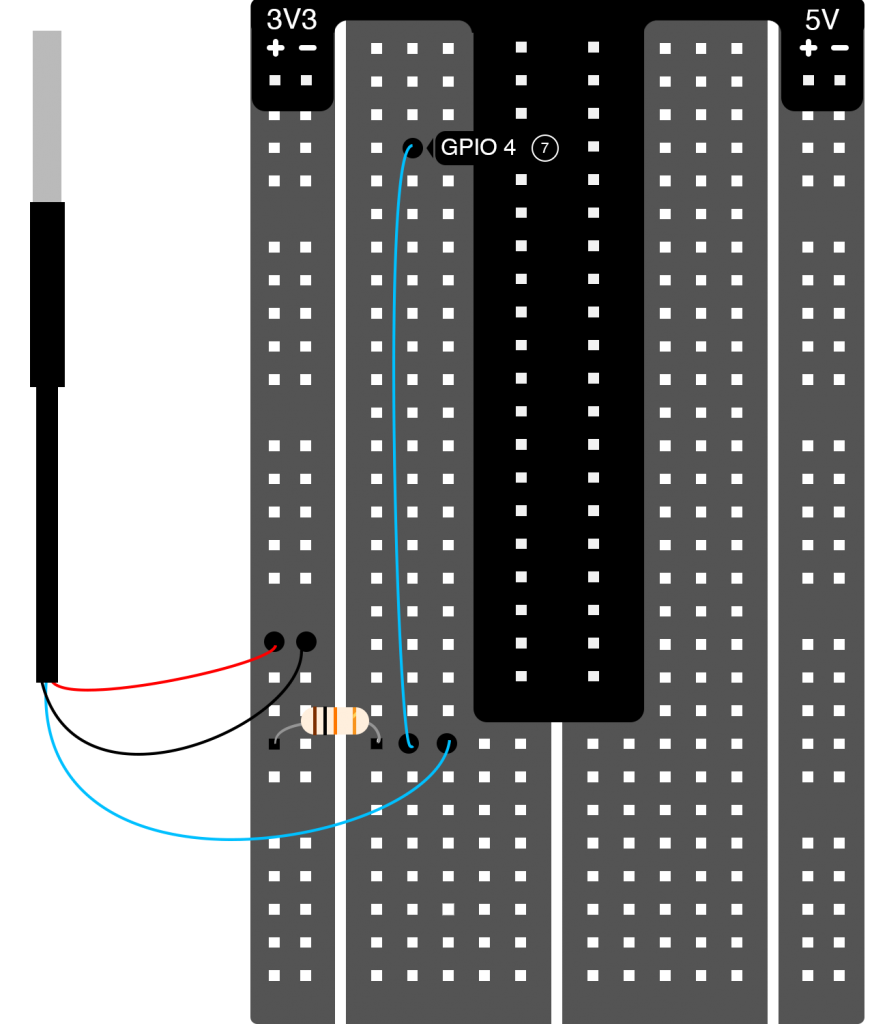
“Hello World!” – This is likely the output of the first program you ever wrote when learning how to code.  Setting up a device to stream temperature data is quickly becoming the de facto Internet of Things (IoT) “Hello World!” project.  If printing “Hello World!” the first time was a long, frustrating task, you might have never written another program.  Your first IoT project should only put a big ol’ smile on your face.  This fun, easy Raspberry Pi project will introduce you to the wonderful world of IoT data streaming.

Here are the supplies you will need:

* An internet connected Raspberry Pi or your single-board computer of choice.
* A DS18B20 temperature sensor (<https://www.adafruit.com/product/381>).
* A breadboard for wiring up a simple circuit (in the [Adafruit Pi starter kit](https://www.adafruit.com/products/2126)).
* A 4.7K to 10K resistor (also in the [Adafruit Pi starter kit](https://www.adafruit.com/products/2126)).

The DS18B20 temperature sensor works well with the Raspberry Pi because it has a digital output, and the Pi has no on-board analog to digital convertors (ADC).  Raspbian includes an interface to read the output of the sensor.  We just have to write a little code to grab and parse out the temperature.  Adafruit has a great tutorial [here](https://learn.adafruit.com/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing/hardware) (and [here for the PDF version](http://blog.initialstate.com/wp-content/uploads/2014/12/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing.pdf)) for using the DS18B20 that we are simply going to follow and modify to stream the temperature instead of just outputting it to the screen.

The hardware setup is simple.  DS18B20 red wire to 3.3V.  Black wire to GND.  Blue wire to a pull-up resistor and to GPIO pin 4 of your Pi.  If your DS18B20 has red/blue/yellow wires, hook the red to 3.3V, blue to GND, and yellow to the pull-up/GPIO pin 4.

[](http://blog.initialstate.com/wp-content/uploads/2014/12/Breadboard-Temperature-Sensor-DS18B20_2.png)

To start the temperature sensor read interface we need to run two commands.  Go to a command prompt on your Pi or SSH into your Pi.  Type the following commands:

[?](http://blog.initialstate.com/stream-temperature-from-pi/)

|  |
| --- |
| sudo modprobe w1-gpio  sudo modprobe w1-therm |

The output of your temperature sensor is now being written to a file on your Pi.  To find that file,

[?](http://blog.initialstate.com/stream-temperature-from-pi/)

|  |
| --- |
| cd /sys/bus/w1/devices |

In this directory, there will be a sub-directory that starts with “28-“.  What comes after the “28-” is the serial number of your sensor.  cd into that directory.  Inside this directory, a file named w1\_slave contains the output of your sensor.  (\*Update 2/24/15: If there is no sub-directory that starts with a “28-“, you may have the issue listed [here](http://www.raspberrypi.org/forums/viewtopic.php?f=28&t=97314) and discussed [here](http://www.raspberrypi.org/forums/viewtopic.php?f=28&t=100525). If so, simply add the line “dtoverlay=w1-gpio,gpiopin=4” to your /boot/config.txt file and reboot.) The contents of this file will look something like this:

[?](http://blog.initialstate.com/stream-temperature-from-pi/)

|  |
| --- |
| a2 01 4b 46 7f ff 0e 10 d8 : crc=d8 YES  a2 01 4b 46 7f ff 0e 10 d8 t=26125 |

The number after “t=” is the number we want.  This is the temperature in 1/1000 degrees Celsius (in the example above, the temperature is 26.125 C).  We just need a simple program that reads this file and parses out that number.  Lucky for us, Adafruit already created a Python script for us that does just this and outputs the temperature to the screen.  Create a new file on your Pi (e.g. temperature.py) and copy-paste the following code in.

[?](http://blog.initialstate.com/stream-temperature-from-pi/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | 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(.5) |

Run this script (“sudo python temperature.py”) to see the output of your temperature sensor on the screen.

This is the point where we deviate from the [Adafruit tutorial](https://learn.adafruit.com/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing/hardware).  Sending temperature to the screen is boring.  We want to stream this data to somewhere we can see not only the current temperature but a history of captured temperature data (plus, who wants to “read” a bunch of temperatures as text; we want a pretty data visualization).  First install the Initial State streamer to give our temperature measurements a destination to go to (go [here for instructions on setting up the Initial State streamer](http://blog.initialstate.com/new-initial-state-data-streamer/); it is super easy and takes less than two minutes).  We are going to modify the script above to stream the temperature to our Initial State account instead of outputting it to the screen.  Here are the modifications:

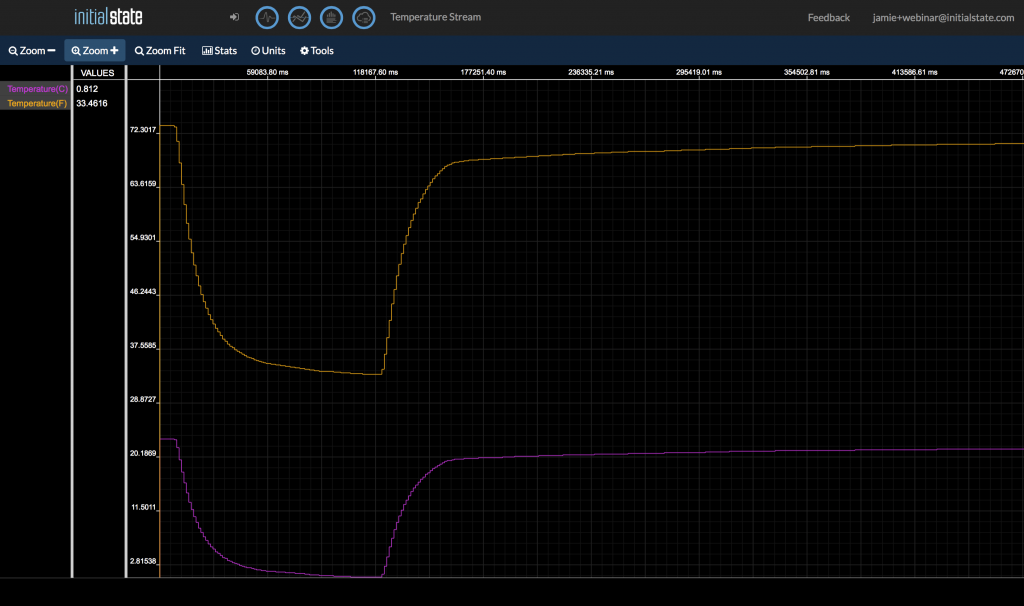
[?](http://blog.initialstate.com/stream-temperature-from-pi/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | import os  import glob  import time  from ISStreamer.Streamer import Streamer    streamer = Streamer(bucket\_name="Temperature Stream", access\_key="PUT YOUR ACCESS KEY HERE")    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          return temp\_c    while True:      temp\_c = read\_temp()      temp\_f = temp\_c \* 9.0 / 5.0 + 32.0      streamer.log("temperature(C)", temp\_c)      streamer.log("temperature(F)", temp\_f)      time.sleep(.5) |

You will need to copy+paste the access key associated with your Initial State account in between the quotes on line 6 where it says “PUT YOUR ACCESS KEY HERE”. You can find your access key under your account settings or on the landing page once you log into your account.

These are the modifications we just made:  
On line 4, we imported the Initial State data streamer module.  
On line 6, we created a new bucket called “Temperature Stream” and associated it with our account via our access key.  
On line 30, we modified the read\_temp() function to only return the temperature in Celsius.  
On line 34, we converted the output of read\_temp() to fahrenheit.  
On lines 35 and 36, we streamed the two temperatures.

Run the modified script (“sudo python temperature.py” or if you want to set this to run uninterrupted for a long time, you might want to use the nohup command – “nohup sudo python temperature.py > tmp.txt &”).  I put my temperature sensor on ice then warmed it back up again in my hand to test it out.  Go to your Initial State account and a new log will show up in your log shelf called “Temperature Stream”.  View this in Waves or Lines to see your data streaming in real-time.  Here is what my data looked like in Lines.

[](http://blog.initialstate.com/wp-content/uploads/2014/12/Screen-Shot-2014-12-19-at-2.28.34-PM.png)

Now that you can capture and stream temperature data, you have the tools to stream anything you can capture – sensor outputs, software variables, hardware events, [button presses](http://blog.initialstate.com/raspberry-pi-pressing-buttons-light-leds/), doors opening, your heart rate, gps data, [your hamster’s activity](http://blog.initialstate.com/tutorial-turn-raspberry-pi-hamster-fitness-tracker/), … anything. Get started streaming IoT data by signing up for a free Initial State account at <https://www.initialstate.com/app#/register/>.