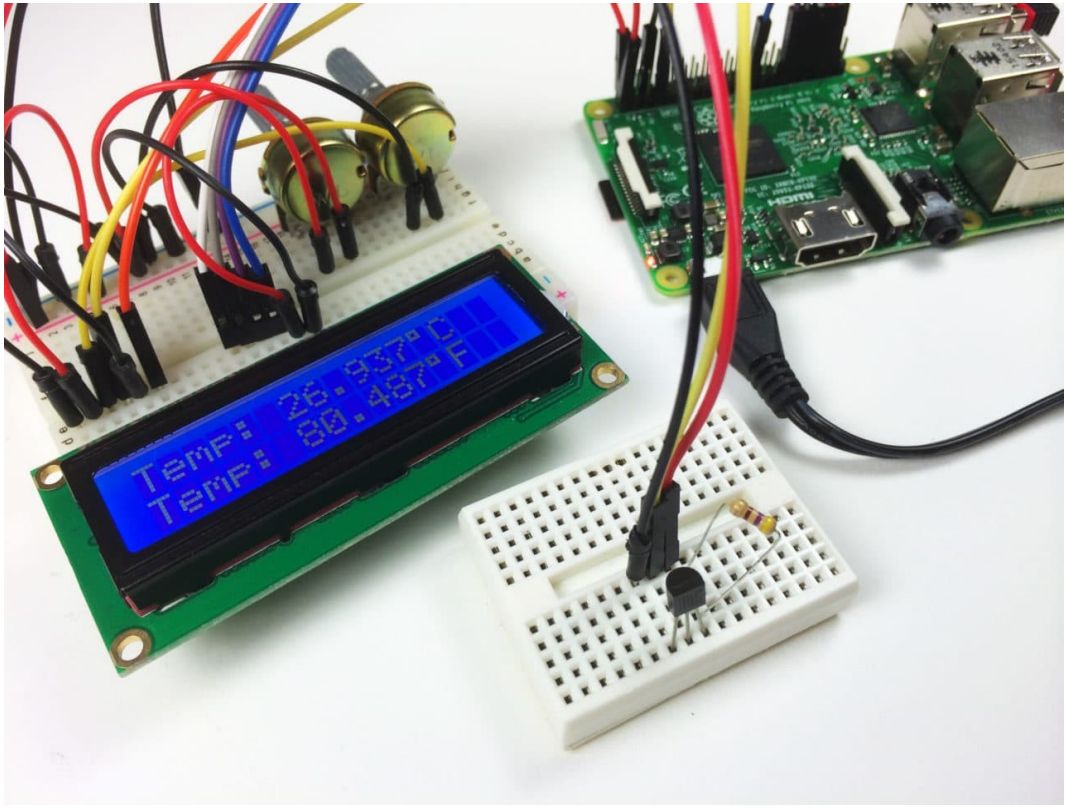
"Integration Of Temprature and Pressure sensor to Raspberry Pi"

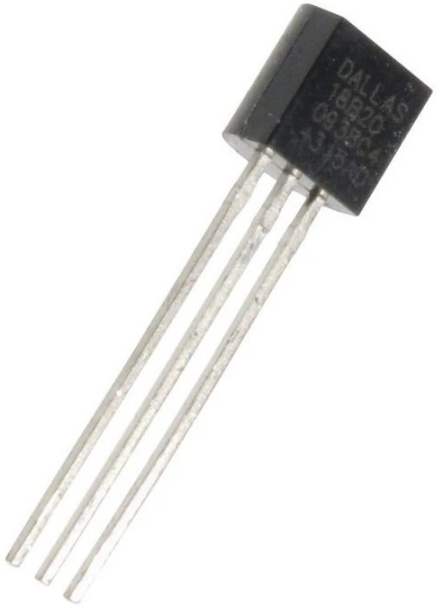
~Shubham Jain
SM20MTECH12007

Integration of Temprature Sensor with Raspberry Pi

- Introduction: DS18B20 Digital Temperature Sensor
- Hardware Prepration
- Software Prepration
- Python Code
- Code Explained
- Reference



ABOUT THE DS18B20 Sensor



The DS18B20 communicates with the “One-Wire” communication protocol, a proprietary serial communication protocol that uses only one wire to transmit the temperature readings to the microcontroller.

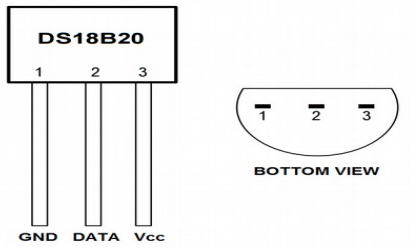
The DS18B20 can be operated in what is known as parasite power mode. Normally the DS18B20 needs three wires for operation: the Vcc, ground, and data wires. In parasite mode, only the ground and data lines are used, and power is supplied through the data line.

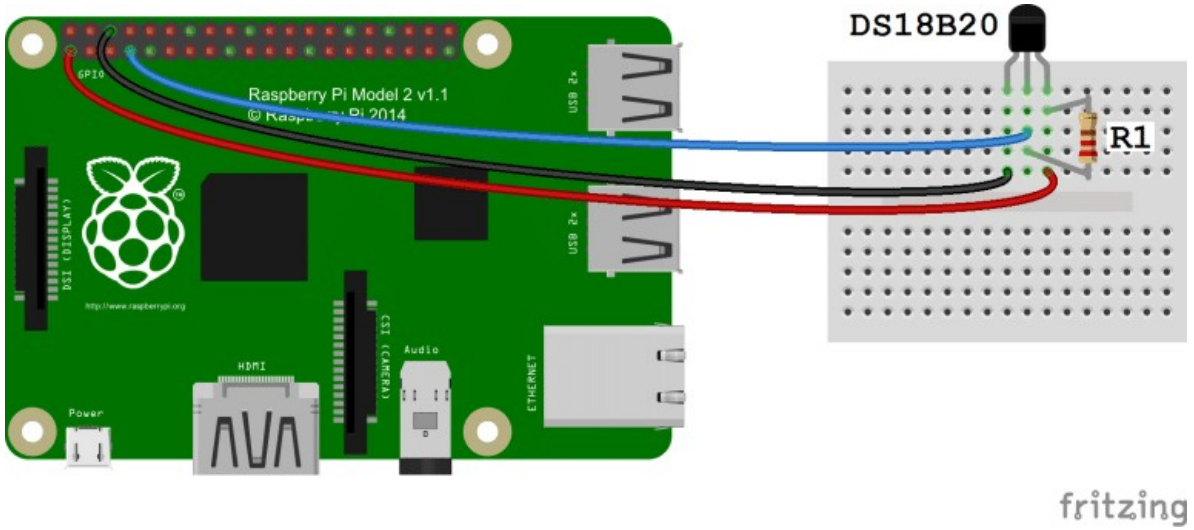
The DS18B20 also has an alarm function that can be configured to output a signal when the temperature crosses a high or low threshold that’s set by the user.

A 64 bit ROM stores the device’s unique serial code. This 64 bit address allows a microcontroller to receive temperature data from a virtually unlimited number of sensors at the same pin. The address tells the microcontroller which sensor a particular temperature value is coming from.

Hooking up Sensor with Raspberry Pi

- Connect GPIO GND [Pin 6] on the Pi to the negative rail on the breadboard and connect GPIO 3.3V [Pin 1] on the Pi to the Positive rail on the breadboard.
- Plug the DS18B20+ into your breadboard, ensuring that all three pins are in different rows. Familiarise yourself with the pin layout, as it's quite easy to hook it up backwards!
- Connect DS18B20+ GND [Pin 1] to the negative rail of the breadboard.
- Connect DS18B20+ VDD [Pin 3] to the positive rail of the breadboard
- Place your 4.7k ohm resistor between DS18B20+ DQ [Pin 2] and a free row on your breadboard.
- Connect that free end of the 4.7k? resistor to the positive rail of the breadboard.
- Finally, connect DS18B20+ DQ [Pin 2] to GPIO 4 [Pin 7] with a jumper wire.





ENABLE THE ONE-WIRE INTERFACE

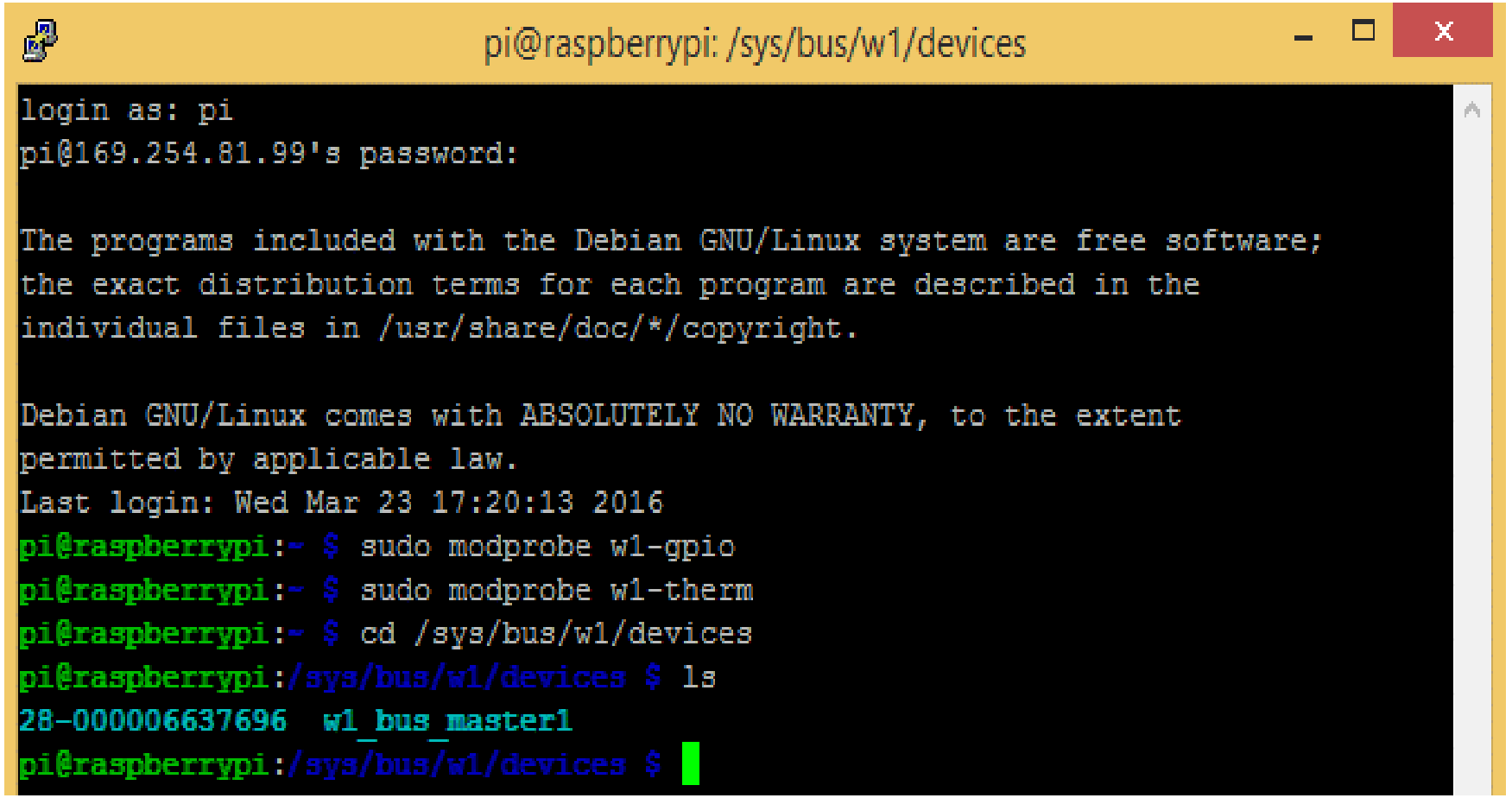
We’ll need to enable the One-Wire interface before the Pi can receive data from the sensor. Once you’ve connected the DS18B20, power up your Pi and log in, then follow these steps to enable the One-Wire interface:

At the command prompt, enter `sudo nano /boot/config.txt`, then add this to the bottom of the file:

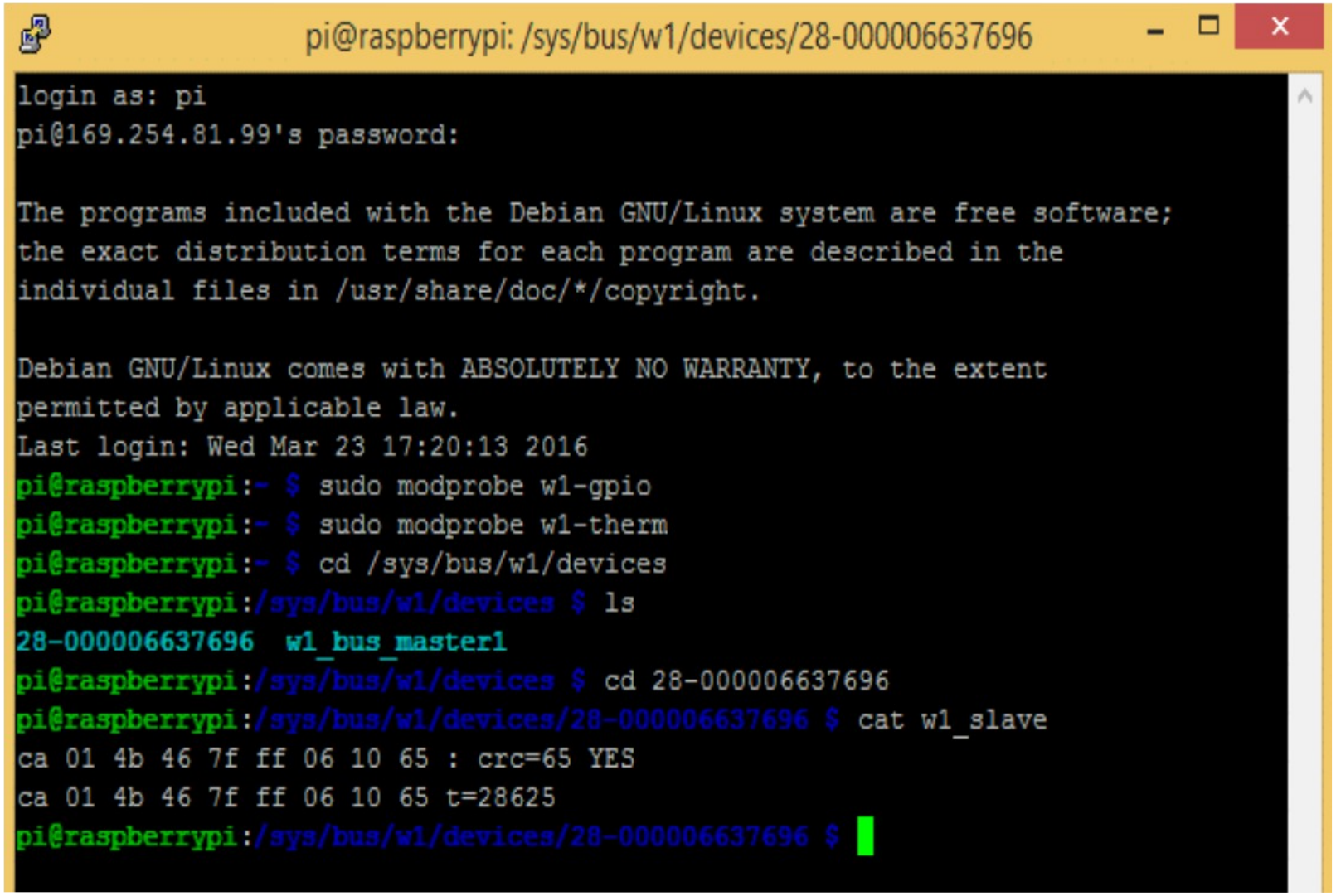
```
dtoverlay=w1-gpio
```

Exit Nano, and reboot the Pi with `sudo reboot`

Software Prepration



Log in to the Pi again, and at the command prompt enter sudo modprobe w1-gpio
Then enter sudo modprobe w1-therm
Change directories to the /sys/bus/w1/devices directory by entering cd /sys/bus/w1/devices
Now enter cd 28-XXXXXXXXXXXX (change the X's to your own address)
For example, cd 28-000006637696
Enter cat w1_slave which will show the raw temperature reading output by the sensor.


pi@raspberrypi: /sys/bus/w1/devices/28-000006637696
login as: pi
pi@169.254.81.99's password:
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 23 17:20:13 2016
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
28-000006637696 w1_bus_master1
pi@raspberrypi:/sys/bus/w1/devices $ cd 28-000006637696
pi@raspberrypi:/sys/bus/w1/devices/28-000006637696 $ cat w1_slave
ca 01 4b 46 7f ff 06 10 65 : crc=65 YES
ca 01 4b 46 7f ff 06 10 65 t=28625
pi@raspberrypi:/sys/bus/w1/devices/28-000006637696 $

Create python file and write the code given below

```
import os                                          //os module(allows to use functions)
import glob                                        //glob module for pathness
import time

os.system('modprobe w1-gpio')                      //to enable single wire commmunication
os.system('modprobe w1-therm')                  //to enable temprature sensor

base_dir = '/sys/bus/w1/devices/'                       //String Value
device_folder = glob.glob(base_dir + '28*')[0]     // to find file in base_dir
                                                   // starting with 28
device_file = device_folder + '/w1_slave'               // to store w1_slave address

def read_temp_raw():                                    // defining new method
    f = open(device_file, 'r')                     // set readOnly device_file
    lines = f.readlines()                               // store all lines as list in
                                                        variable
    f.close()                                      // close
    return lines                                        // method returns sored lines

def read_temp():                                        // method to output temprature
    lines = read_temp_raw()                        // store string returned from
                                                   // above method
    while lines[0].strip()[-3:] != 'YES':               // remove spaces in line 0,
                                                        //checks last 3 values in string
        time.sleep(0.2)                                 // time delay of 2 milisecond
        lines = read_temp_raw()                    // while loop exit at YES
    equals_pos = lines[1].find('t=')               //find t= in line 1
    if equals_pos != -1:                                // if no error, enter loop
        temp_string = lines[1][equals_pos+2:]      // store temp without decimal
        temp_c = float(temp_string) / 1000.0       // store temp in celcius
        temp_f = temp_c * 9.0 / 5.0 + 32.0              // store temp in farhenhiet
        return temp_c, temp_f                           // return temprature

while True:
print(read_temp())                                 // print temp after every 1s.
time.sleep(1)
```

Explanation

- #1 : The os Python module provides a big range of useful methods to manipulate files and directories.
- #2 : The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell.
- #3 : time module available in Python which provides functions for working with times
- #4 : To Load the drivers for single wire communication.
- #5 : To Load the drivers for Temprature sensors.
- #6 : Address `/sys/bus/w1/devices/` saved in bariable base_dir.
- #7 : Finds the file named with first digits as ‘28’ in address saved above in base_dir. And add that file in address i.e `/sys/bus/w1/devices/28.......` i in variable device_folder.
- #8 : File named `/w1_slave` is added to the already saved address in variable device_folder and now this address is stored in variable device_file
- #9 : New method to store contents of our w1_slave file defined.

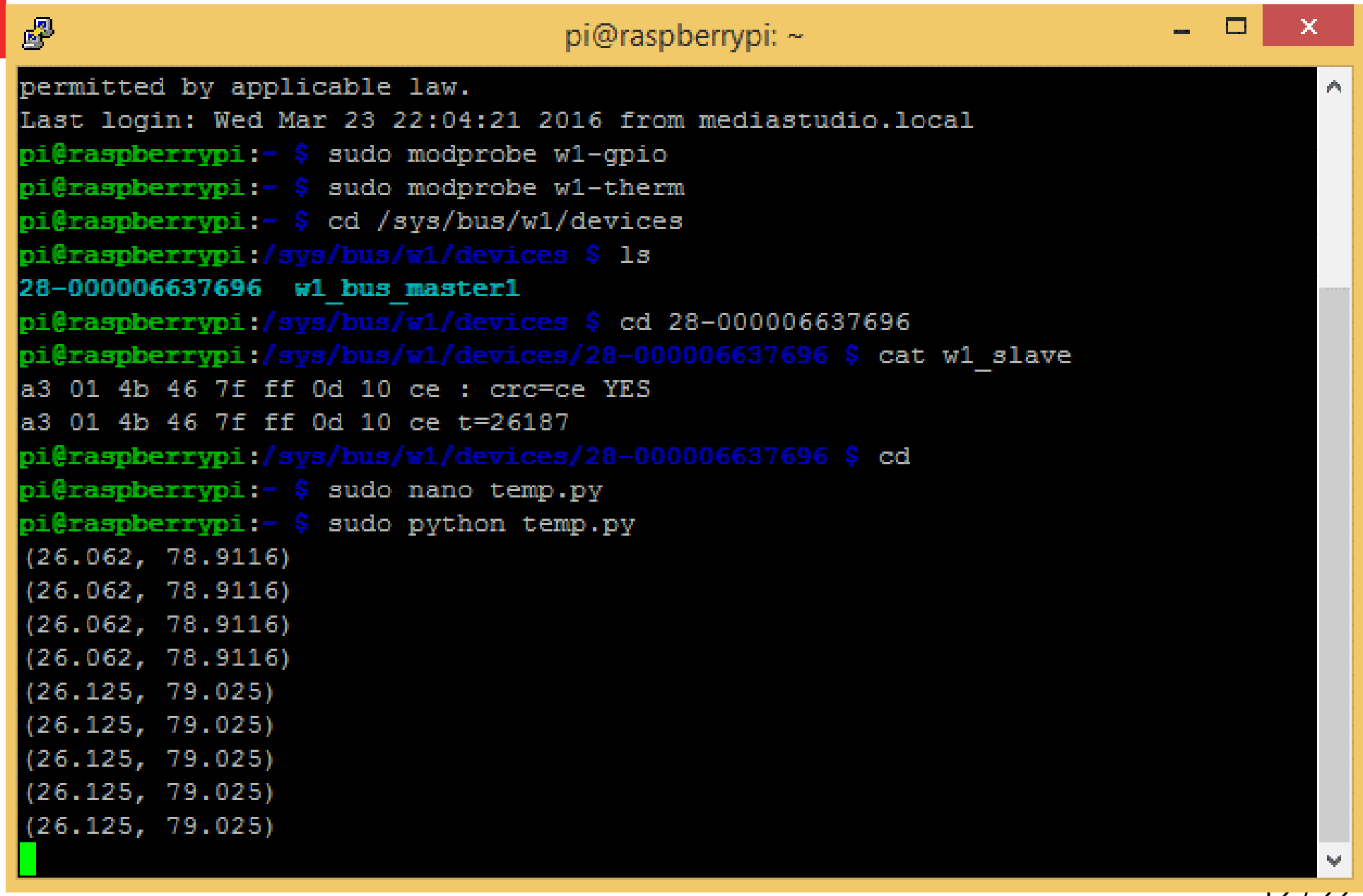
Explanation

- #10 : The w1slave file is opened in read mode with open() function and stored in f.
- #11 : The lines in w1slave are read with readline() function and these lines are then stored in variable lines of type List.
- #12 : opened w1slave file is closed.
- #13 : Our method `read_temp_raw()` returns the List of string stored in variable ‘lines’ on being called.
- #14 : new method named `read_temp()` defined.
- #15 : List of strings returned from `read_temp_raw()` are stored in new variable lines
- #16 : White spaces are stripped from String stored at 0^{th} index in the lines list. Last 3 characters after stripping are checked for sequence ‘YES’. If the sequence is matched, compiler exits while loop.
- #17 : Excecute if YES not found. `time.sleep(0.2)` introduces delay of 2 miliseconds.

Explanation

- #18 : store returned list from read_temp_raw again to lines variable. The loop is rerun.
- #19 : Find ‘t=’ in 1st index position of list ‘Lines’ and store the index postion of Starting of ‘t=’ in variable `equals_pos`.
- #20 : If sensor is not installed properly we get -1 and if condition is not satisfied. And if we get a value, compiler will run the code for temprature calculation inside if condition.
- #21 : `temp_string = lines[1][equals_pos+2:]` +2 is done to move index postion 2 places forward (as ‘t=’ is stored in those two spaces).
- #22 : temprature in celcius calculation.
- #23 : temprature in fahrenheit calculation.
- #24 : method returns temp_c , temp_f.
- #25 : while loop for continious output.
- #26 : print temprature sensor readings.
- #27 : introducing time delay of 2 seconds between each reading.

Temprature Output



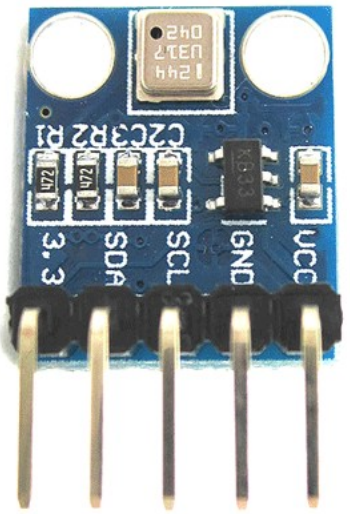
Pressure sensor

Integration of Temprature Sensor with Raspberry Pi

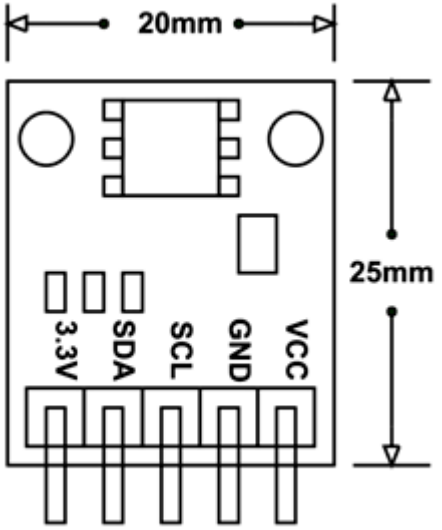
- Introduction: BMP180 digital pressure sensor
- Hardware Prepration
- Software Prepration
- Python Code
- Code Explained
- Reference

ABOUT THE BMP180 Digital Pressure Sensor

BMP180 is a high precision sensor designed for consumer applications. Barometric Pressure is nothing but weight of air applied on everything. The air has weight and wherever there is air its pressure is felt. BMP180 sensor senses that pressure and provides that information in digital output. Also the temperature affects the pressure and so we need temperature compensated pressure reading. To compensate, the BM180 also has good temperature sensor.

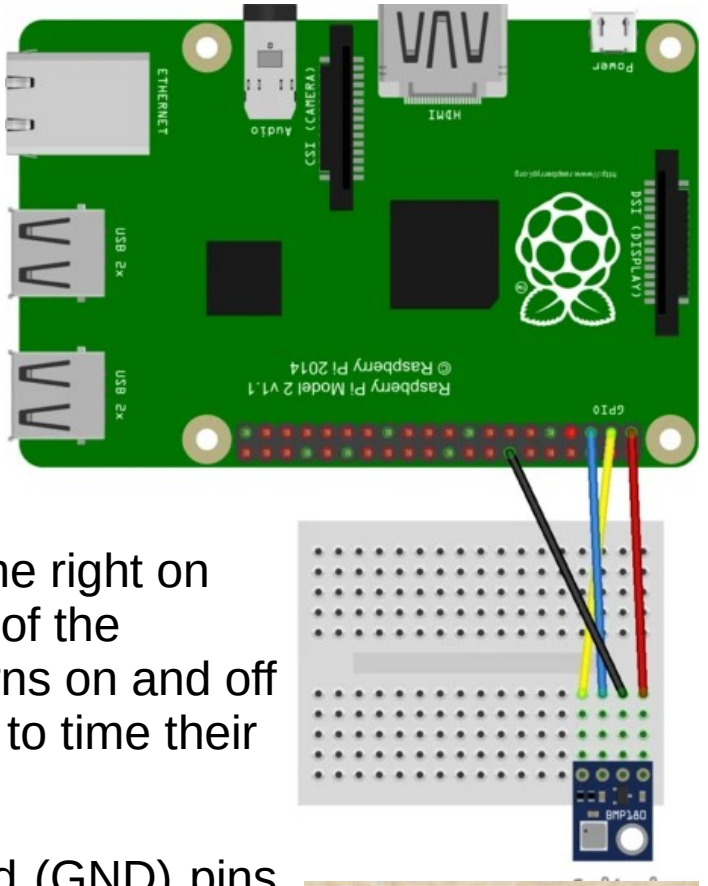


The BMP180 is an i2c board, which means that it uses the i2c protocol to communicate with your Raspberry Pi. The advantage of i2c is that it only uses two pins on the Raspberry Pi (plus power and ground) to communicate with a lot of different devices. One pin carries a clock signal, and the other carries the data.

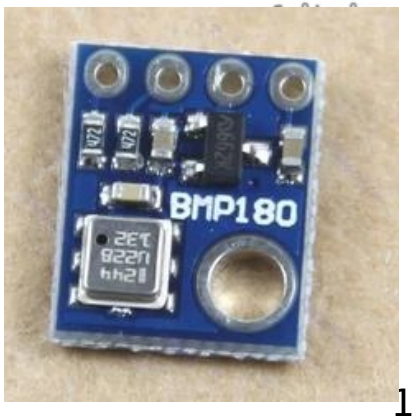


Hooking up Sensor with Raspberry Pi

- The red jumper goes from the 3v3 (3.3 volt power supply) pin of the Raspberry Pi to the VCC pin of the BMP180.
- The yellow jumper goes from the second pin from the right on the top row of the Raspberry Pi to the SDA pin of the BMP180. The BMP180 uses this wire to communicate with the Raspberry Pi.



- The blue jumper goes from the third pin from the right on the top row of the Raspberry Pi to the SCL pin of the BMP180. This pin supplies a clock signal (it turns on and off at regular intervals) that is used by i2c devices to time their communication to the Raspberry Pi.
- The black jumper goes from any of the ground (GND) pins of the Raspberry Pi to the GND pin on the BMP180.
- Before you turn anything on, double check your connections. It is unlikely that anything bad will happen, but it is best to check twice before applying power.



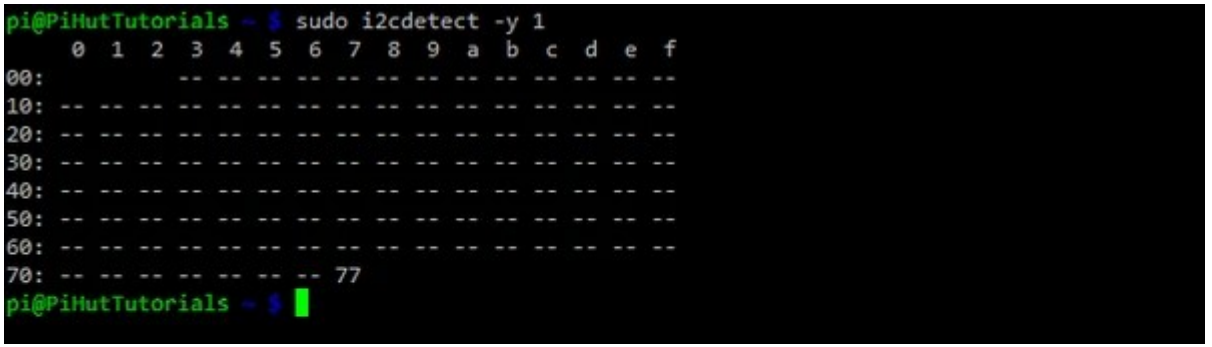
ENABLE THE I²C INTERFACE

To use the I²C interface, we should enable the I2C first. Please enter the following command:sudo nano /boot/config.txt, then add this to the bottom of the file:dtparam=i2c_arm=on

Press Ctrl+X ， and type “Y” to save the file you revised. reboot Pi.

check the connectivity with sensor by commandsudo i2cdetect -y 1

If the command worked, you should see the following:



This is showing that the BMP180 is on channel '77'.

Software Prepration

Now we need to install some Adafruit libraries so that Python knows how to read the output of the BMP180.

- Install the GIT application which will be able to retrieve code from the GitHub (a website that is used to store and version control code):

  ```
  sudo apt-get update
  ```

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

- Create and move to a directory that will contain your code.

  ```
  Mkdir ~/BMP180Code
  ```

  ```
  Cd ~/BMP180Code
  ```

- Download the GIT repository for the BMP180.

  ```
  git clone https://github.com/adafruit/Adafruit_Python_BMP.git
  ```

- The library now needs compiling so that you can use it

  ```
  cd Adafruit_Python_BMP
  ```

  ```
  sudo python setup.py install
  ```

Create python file and write the code given below

```
import time
import Adafruit_BMP.BMP085 as BMP085

bmp = BMP085(0x77, 1)

# To specify a different operating mode, uncomment one of the following:
# bmp = BMP085(0x77, 0)  # ULTRALOWPOWER Mode
# bmp = BMP085(0x77, 1)  # STANDARD Mode
# bmp = BMP085(0x77, 2)  # HIRES Mode
# bmp = BMP085(0x77, 3)  # ULTRAHIRES Mode

while True :
  temp = bmp.readTemperature()
  pressure = bmp.readPressure()
  altitude = bmp.readAltitude(101560)

print(f"Temperature: {temp} C")
print(f"Pressure: {pressure / 100.0} hPa")
print(f"Altitude: {altitude} m" )

  time.sleep(.1)
```

Explanation

- #1 : time module available in Python which provides functions for working with times
- #2 :Python library for accessing the BMP series pressure and temperature sensors like the BMP085/BMP180 on a R-Pi.
- #3 : Create an 'object' containing the BMP180 data. Initialise the BMP085 and use STANDARD mode (default value).
- #4 : different operating mode ULTRALOWPOWER Mode, STANDARD Mode, HIRES Mode, ULTRAHIRES Mode.
- #5 : Alway true while loop, for continious reading.
- #6 : Store raw temprature value in `temp` variable.
- #7 : Store raw pressure value in `pressure` variable.
- #8 : Store raw temprature value in `altitude` variable.
- #9 : Prints Temprature value degree celcius.
- #10 : Prints Pressure value in hPa.
- #11 : Prints Altitude in m.
- #12 : time delay of 1 second between consequtive readings.

Pressure Output

```
pi@raspberrypi:~/bmp180-python $ sudo  python  ./BMP180test.py
Temperature: 27.10 C
Pressure:    994.75 hPa
Altitude:     155.17

Temperature: 27.00 C
Pressure:    994.74 hPa
Altitude:     154.67

Temperature: 27.10 C
Pressure:    994.74 hPa
Altitude:     155.17

Temperature: 27.10 C
Pressure:    994.69 hPa
Altitude:     154.92
```

Refrences:

https://osoyoo.com/2017/07/06/bmp180_pressure-sensor/

https://thepihut.com/blogs/raspberry-pi-tutorials/18025084-sensors-pressure-temperature-and-altitude-with-the-bmp180

https://www.udemy.com/course/pi-ultimate-guide

https://wiki.seeedstudio.com/Grove-Barometer_Sensor-BMP180

www.google.com

https://thepihut.com/blogs/raspberry-pi-tutorials/ds18b20-one-wire-digital-temperature-sensor-and-the-raspberry-pi

https://www.waveshare.com/wiki/Raspberry_Pi_Tutorial_Series:_1-Wire_DS18B20_Sensor

www.raspberrypi.org

www.geeksforgeeks.org

www.stackoverflow.com

www.electronicwings.com