GoPiGo3 Sensors Tutorial

Contents

Dexter Sensors Documentation

DI sensor documentation: https://di-sensors.readthedocs.io/en/master/

Dexter Temperature, Humidity, and Pressure Sensor Tutorial

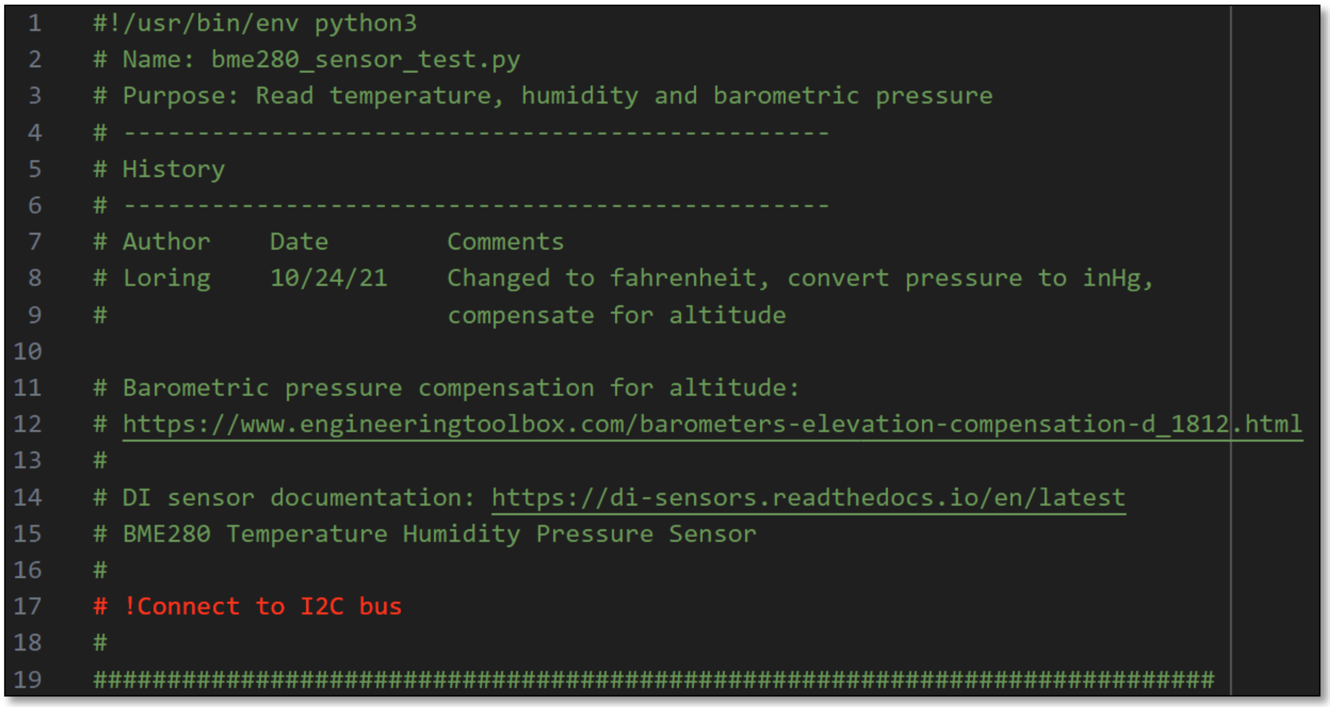
A tutorial for how to use the Dexter Temperature, Humidity Sensor (BME280).

Barometric pressure compensation for altitude:
https://www.engineeringtoolbox.com/barometers-elevation-compensation-d_1812.html

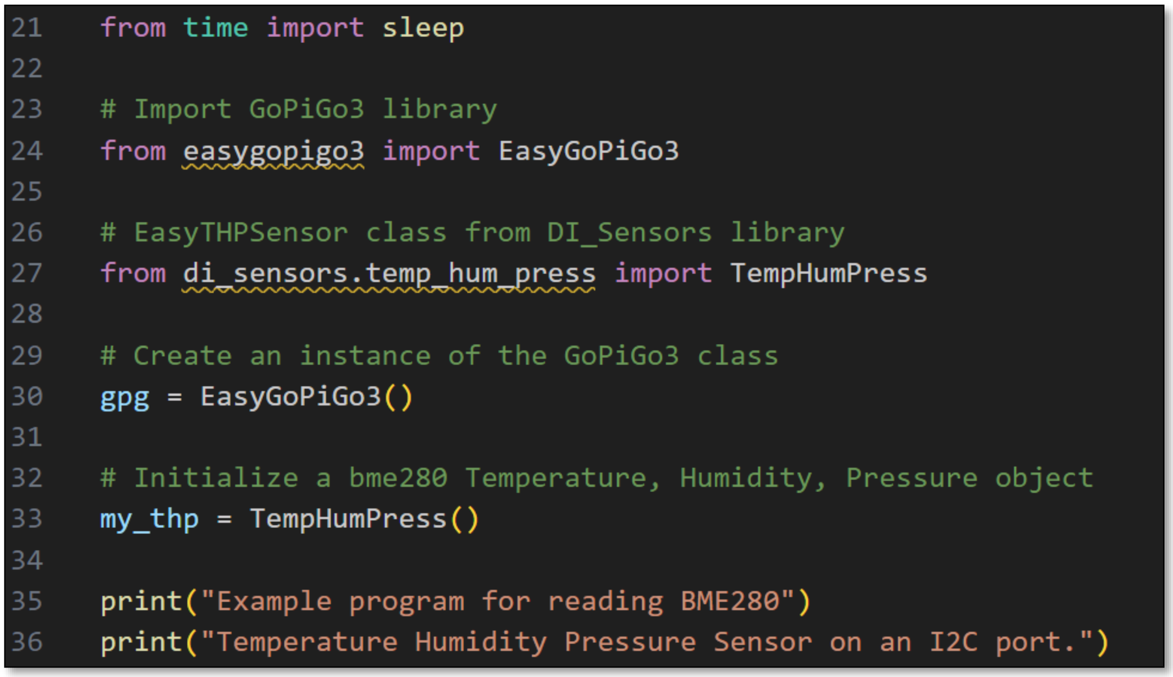
1. Shutdown the GoPiGo3. (Do not connect sensors when the GoPiGo3 has power.)
2. Plug the BME280 sensor into an I2C port.
3. Mount the sensor on a sensor mount.

bme280_test.py

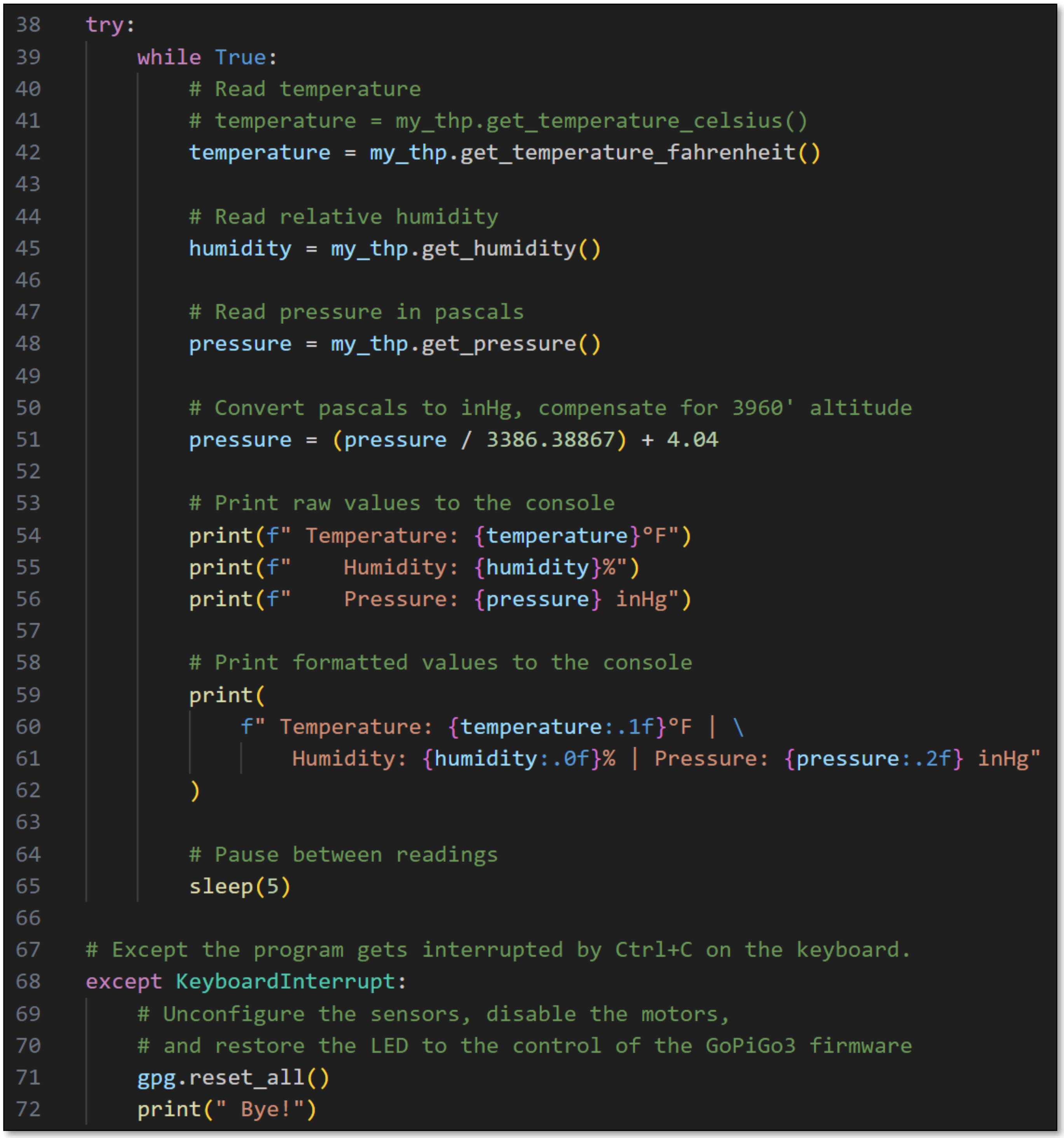
This program will read the Dexter Temperature, Humidity Sensor (BME280) every 5 seconds and display to the console.



```python
#!/usr/bin/env python3
# Name: bme280_sensor_test.py
# Purpose: Read temperature, humidity and barometric pressure
# ------------------------------------------------
# History
# ------------------------------------------------
# Author    Date        Comments
# Loring    10/24/21    Changed to fahrenheit, convert pressure to inHg,
#                       compensate for altitude

# Barometric pressure compensation for altitude:
# https://www.engineeringtoolbox.com/barometers-elevation-compensation-d_1812.html
#
# DI sensor documentation: https://di-sensors.readthedocs.io/en/latest
# BME280 Temperature Humidity Pressure Sensor
#
# !Connect to I2C bus
#
############################################################################
```



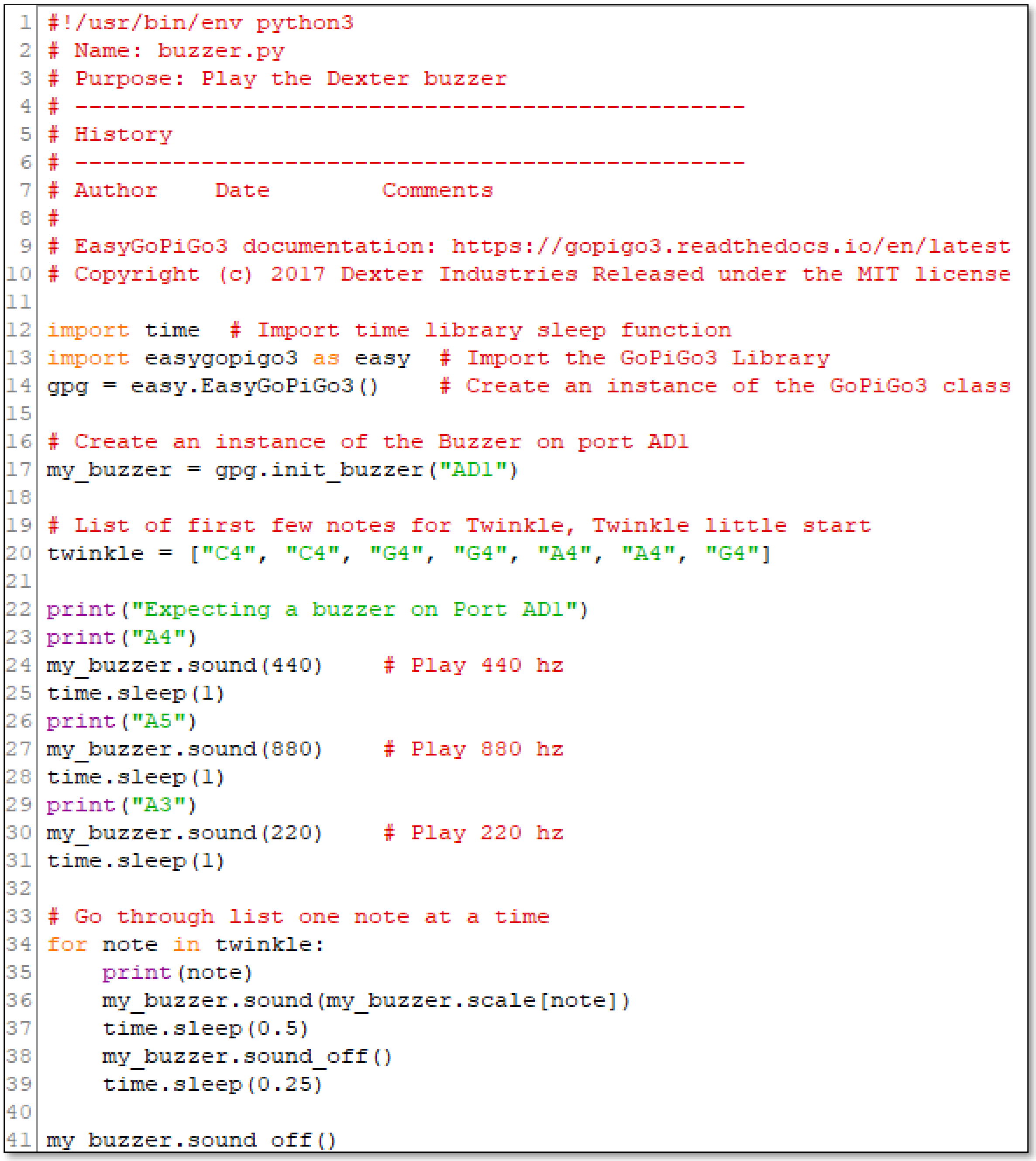
```python
from time import sleep

# Import GoPiGo3 library
from easygopigo3 import EasyGoPiGo3

# EasyTHPSensor class from DI_Sensors library
from di_sensors.temp_hum_press import TempHumPress

# Create an instance of the GoPiGo3 class
gpg = EasyGoPiGo3()

# Initialize a bme280 Temperature, Humidity, Pressure object
my_thp = TempHumPress()

print("Example program for reading BME280")
print("Temperature Humidity Pressure Sensor on an I2C port.")
```



```python
try:
    while True:
        # Read temperature
        # temperature = my_thp.get_temperature_celsius()
        temperature = my_thp.get_temperature_fahrenheit()

        # Read relative humidity
        humidity = my_thp.get_humidity()

        # Read pressure in pascals
        pressure = my_thp.get_pressure()

        # Convert pascals to inHg, compensate for 3960' altitude
        pressure = (pressure / 3386.38867) + 4.04

        # Print raw values to the console
        print(f" Temperature: {temperature}°F")
        print(f"    Humidity: {humidity}%")
        print(f"    Pressure: {pressure} inHg")

        # Print formatted values to the console
        print(
            f" Temperature: {temperature:.1f}°F | \
                Humidity: {humidity:.0f}% | Pressure: {pressure:.2f} inHg"
        )

        # Pause between readings
        sleep(5)

# Except the program gets interrupted by Ctrl+C on the keyboard.
except KeyboardInterrupt:
    # Unconfigure the sensors, disable the motors,
    # and restore the LED to the control of the GoPiGo3 firmware
    gpg.reset_all()
    print(" Bye!")
```

Dexter Grove Buzzer

AD1 or **AD2** port



```python
#!/usr/bin/env python3
# Name: buzzer.py
# Purpose: Play the Dexter buzzer
# ------------------------------------------------
# History
# ------------------------------------------------
# Author    Date        Comments
#
# EasyGoPiGo3 documentation: https://gopigo3.readthedocs.io/en/latest
# Copyright (c) 2017 Dexter Industries Released under the MIT license

import time  # Import time library sleep function
import easygopigo3 as easy  # Import the GoPiGo3 Library
gpg = easy.EasyGoPiGo3()    # Create an instance of the GoPiGo3 class

# Create an instance of the Buzzer on port AD1
my_buzzer = gpg.init_buzzer("AD1")

# List of first few notes for Twinkle, Twinkle little start
twinkle = ["C4", "C4", "G4", "G4", "A4", "A4", "G4"]

print("Expecting a buzzer on Port AD1")
print("A4")
my_buzzer.sound(440)    # Play 440 hz
time.sleep(1)
print("A5")
my_buzzer.sound(880)    # Play 880 hz
time.sleep(1)
print("A3")
my_buzzer.sound(220)    # Play 220 hz
time.sleep(1)

# Go through list one note at a time
for note in twinkle:
    print(note)
    my_buzzer.sound(my_buzzer.scale[note])
    time.sleep(0.5)
    my_buzzer.sound_off()
    time.sleep(0.25)

my_buzzer.sound_off()
```

Dexter Light and Color Sensor

I2C port

Example code: sensor_light_color.py

Dexter Inertial Measurement Unit (IMU)

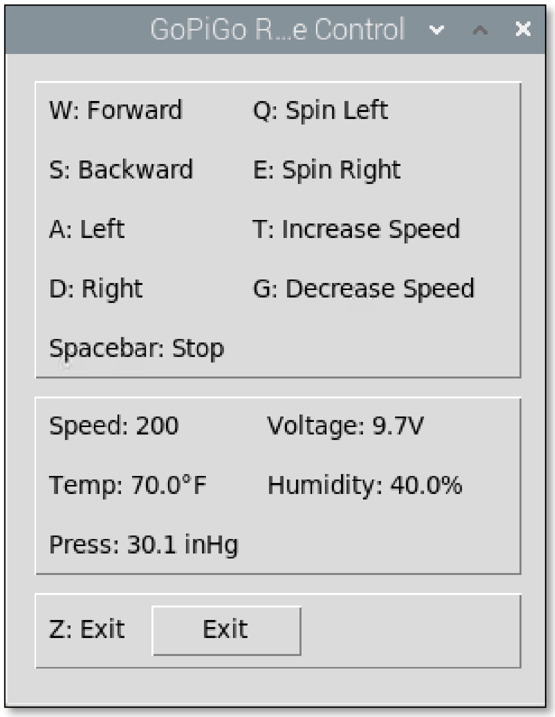
I2C port

Example code: sensor_imu.py

Sensors and Tkinter

The next step would be to send the data every 15 seconds or more to ThingSpeak. There is a tutorial to get started with that. All sensors can be setup to upload data to ThingSpeak.

The data can also be displayed in a GUI program. This is an example of a Tkinter remote control program that also displays real time data from a bme280 sensor.



W: Forward Q: Spin Left
S: Backward E: Spin Right
A: Left T: Increase Speed
D: Right G: Decrease Speed
Spacebar: Stop

Speed: 200 Voltage: 9.7V
Temp: 70.0°F Humidity: 40.0%
Press: 30.1 inHg

Z: Exit