**根据产品的不同类型选择MS5837-30BA(300M)/MS5837-02BA(10M)**

**安装**

必须安装 python SMBus 库。

sudo apt-get install python-smbus

通过单击此网页中的下载按钮或使用 git 下载此存储库：

git 克隆 https://github.com/bluerobotics/ms5837-python

如果您想尝试该示例，请移至下载存储库的目录，然后运行python example.py​​. 要使用该库，请将ms5837.py文件复制到您的项目/程序目录并在您的程序中使用此导入语句：import ms5837.

**树莓派**

如果您使用的是 Raspberry Pi，则必须启用 i2c 接口。运行sudo raspi-config，并选择启用i2c接口interfacing options。

**用法**

import ms5837

ms5837 提供了一个通用的 MS5837 类，用于不同的模型

MS5837(model=ms5837.MODEL\_30BA, bus=1)

这些特定于模型的类继承自 MS5837，并且没有任何唯一成员

MS5837\_30BA(bus=1)

MS5837\_02BA(bus=1)

可以通过指定模型和总线来构造 MS5837 对象

sensor = ms5837.MS5837() # Use defaults (MS5837-30BA device on I2C bus 1)

sensor = ms5837.MS5837(ms5837.MODEL\_02BA, 0) # Specify MS5837-02BA device on I2C bus 0

或者通过创建特定于模型的对象

sensor = ms5837.MS5837\_30BA() # Use default I2C bus (1)

sensor = ms5837.MS5837\_30BA(0) # Specify I2C bus 0

**在里面（）**

初始化传感器。这需要在使用任何其他方法之前调用。

sensor.init()

如果传感器已成功初始化，则返回 true，否则返回 false。

**读取（过采样=OSR\_8192）**

读取传感器并更新压力和温度。将使用提供的过采样设置读取传感器。更大的过采样会提高分辨率，但需要更长的时间并增加电流消耗。

sensor.read(ms5837.OSR\_256)

有效的论点是：

ms5837.OSR\_256

ms5837.OSR\_512

ms5837.OSR\_1024

ms5837.OSR\_2048

ms5837.OSR\_4096

ms5837.OSR\_8192

如果读取成功则返回 True，否则返回 False。

**setFluidDensity(密度)**

设置用于深度测量的流体密度 (kg/m^3)。默认流体密度为 ms5837.DENISTY\_FRESHWATER。

sensor.setFluidDensity(1000) # Set fluid density to 1000 kg/m^3

sensor.setFluidDensity(ms5837.DENSITY\_SALTWATER) # Use predefined saltwater density

一些方便的常量是：

ms5837.DENSITY\_FRESHWATER = 997

ms5837.DENSITY\_SALTWATER = 1029

**压力（转换=UNITS\_mbar）**

获取最新的压力测量值。

sensor.pressure() # Get pressure in default units (millibar)

sensor.pressure(ms5837.UNITS\_atm) # Get pressure in atmospheres

sensor.pressure(ms5837.UNITS\_kPa) # Get pressure in kilopascal

一些方便的常量是：

ms5837.UNITS\_Pa = 100.0

ms5837.UNITS\_hPa = 1.0

ms5837.UNITS\_kPa = 0.1

ms5837.UNITS\_mbar = 1.0

ms5837.UNITS\_bar = 0.001

ms5837.UNITS\_atm = 0.000986923

ms5837.UNITS\_Torr = 0.750062

ms5837.UNITS\_psi = 0.014503773773022

返回以毫巴 \* 转换为单位的最新压力。调用 read() 进行更新。

**温度（转换=UNITS\_Centigrade）**

获取最新的温度测量值。

sensor.temperature() # Get temperature in default units (Centigrade)

sensor.temperature(ms5837.UNITS\_Farenheit) # Get temperature in Farenheit

有效的论点是：

ms5837.UNITS\_Centigrade

ms5837.UNITS\_Farenheit

ms5837.UNITS\_Kelvin

以请求的单位返回最近的温度，如果指定了无效单位，则返回以摄氏度为单位的温度。调用 read() 进行更新。

**深度（）**

获取最新的深度测量值（以米为单位）。

sensor.depth()

使用 setFluidDensity() 配置的流体密度 (kg/m^3) 返回以米为单位的最新深度。调用 read() 进行更新。

**高度（）**

获取相对于平均海平面压力的最新高度测量值（以米为单位）。

sensor.altitude()

使用 MSL 处的空气密度返回相对于 MSL 压力的最近高度（以米为单位）。调用 read() 进行更新。

英文原版

**ms5837-python**

A python module to interface with MS5837-30BA and MS5837-02BA waterproof pressure and temperature sensors. Tested on Raspberry Pi 3 with Raspbian.

**Installation**

The python SMBus library must be installed.

sudo apt-get install python-smbus

Download this repository by clicking on the download button in this webpage, or using git:

git clone https://github.com/bluerobotics/ms5837-python

If you would like to try the example, move to the directory where you downloaded the repository, and run python example.py. To use the library, copy the ms5837.py file to your project/program directory and use this import statement in your program: import ms5837.

**Raspberry Pi**

If you are using a Raspberry Pi, the i2c interface must be enabled. Run sudo raspi-config, and choose to enable the i2c interface in the interfacing options.

**Usage**

import ms5837

ms5837 provides a generic MS5837 class for use with different models

MS5837(model=ms5837.MODEL\_30BA, bus=1)

These model-specific classes inherit from MS5837 and don't have any unique members

MS5837\_30BA(bus=1)

MS5837\_02BA(bus=1)

An MS5837 object can be constructed by specifiying the model and the bus

sensor = ms5837.MS5837() # Use defaults (MS5837-30BA device on I2C bus 1)

sensor = ms5837.MS5837(ms5837.MODEL\_02BA, 0) # Specify MS5837-02BA device on I2C bus 0

Or by creating a model-specific object

sensor = ms5837.MS5837\_30BA() # Use default I2C bus (1)

sensor = ms5837.MS5837\_30BA(0) # Specify I2C bus 0

**init()**

Initialize the sensor. This needs to be called before using any other methods.

sensor.init()

Returns true if the sensor was successfully initialized, false otherwise.

**read(oversampling=OSR\_8192)**

Read the sensor and update the pressure and temperature. The sensor will be read with the supplied oversampling setting. Greater oversampling increases resolution, but takes longer and increases current consumption.

sensor.read(ms5837.OSR\_256)

Valid arguments are:

ms5837.OSR\_256

ms5837.OSR\_512

ms5837.OSR\_1024

ms5837.OSR\_2048

ms5837.OSR\_4096

ms5837.OSR\_8192

Returns True if read was successful, False otherwise.

**setFluidDensity(density)**

Sets the density in (kg/m^3) of the fluid for depth measurements. The default fluid density is ms5837.DENISTY\_FRESHWATER.

sensor.setFluidDensity(1000) # Set fluid density to 1000 kg/m^3

sensor.setFluidDensity(ms5837.DENSITY\_SALTWATER) # Use predefined saltwater density

Some convenient constants are:

ms5837.DENSITY\_FRESHWATER = 997

ms5837.DENSITY\_SALTWATER = 1029

**pressure(conversion=UNITS\_mbar)**

Get the most recent pressure measurement.

sensor.pressure() # Get pressure in default units (millibar)

sensor.pressure(ms5837.UNITS\_atm) # Get pressure in atmospheres

sensor.pressure(ms5837.UNITS\_kPa) # Get pressure in kilopascal

Some convenient constants are:

ms5837.UNITS\_Pa = 100.0

ms5837.UNITS\_hPa = 1.0

ms5837.UNITS\_kPa = 0.1

ms5837.UNITS\_mbar = 1.0

ms5837.UNITS\_bar = 0.001

ms5837.UNITS\_atm = 0.000986923

ms5837.UNITS\_Torr = 0.750062

ms5837.UNITS\_psi = 0.014503773773022

Returns the most recent pressure in millibar \* conversion. Call read() to update.

**temperature(conversion=UNITS\_Centigrade)**

Get the most recent temperature measurement.

sensor.temperature() # Get temperature in default units (Centigrade)

sensor.temperature(ms5837.UNITS\_Farenheit) # Get temperature in Farenheit

Valid arguments are:

ms5837.UNITS\_Centigrade

ms5837.UNITS\_Farenheit

ms5837.UNITS\_Kelvin

Returns the most recent temperature in the requested units, or temperature in degrees Centigrade if invalid units specified. Call read() to update.

**depth()**

Get the most recent depth measurement in meters.

sensor.depth()

Returns the most recent depth in meters using the fluid density (kg/m^3) configured by setFluidDensity(). Call read() to update.

**altitude()**

Get the most recent altitude measurement relative to Mean Sea Level pressure in meters.

sensor.altitude()

Returns the most recent altitude in meters relative to MSL pressure using the density of air at MSL. Call read() to update.