

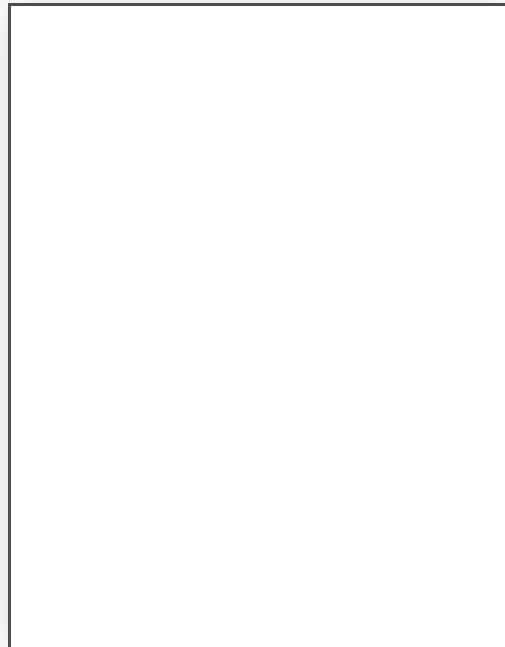
Measuring CO₂ with MQ135

By PS Lab Admin / March 4, 2021

Introduction

The MQ135 is a cheap gas sensor that is primarily intended for detecting the presence of flammable gases. It is marketed as a generalized “air quality” sensor, rather than precision device for measuring the concentration of any specific gas. Nevertheless, according to the data sheet the MQ135 is capable of measuring the concentrations of several gases, one of which is CO₂. This makes it an attractive low-cost alternative to more specialized (and more expensive) CO₂-specific sensors.

Much has been written about using the MQ135 as a CO₂ sensor, mostly focusing on using it together with Arduino. This article will, firstly, discuss the working principles of the MQ135 sensor and how to read the data sheet, and secondly present how to use it with the PSLab.



A MQ135 sensor on a small PCB.

Theory of operation

The MQ135 consists of a surface covered in a thin layer of SnO₂, and a heater resistor which serves to raise the temperature of the SnO₂ surface to several hundred degrees Celsius. SnO₂ is an n-type semiconductor, in which donor electrons are excited to the conduction band at elevated temperatures. However, SnO₂ when exposed to air readily adsorbs oxygen onto its surface. The adsorption reaction consumes electrons from the conduction band to form negatively charged oxygen species. Therefore, SnO₂ has low

conductivity in clean air. [Shimizu]

In the presence of a flammable gas, the gas will also adsorb onto the sensor surface, where it consumes the adsorbed oxygen species to form CO_2 and H_2O . This reaction releases the donor electrons back into the conduction band, thereby raising the conductivity of the sensor. By quantifying the conductivity response to the presence of various gases, a thin SnO_2 surface can be used to determine the concentration of the gas. [Shimizu]

CO_2 , notably, is not a flammable gas. Here, the sensing mechanism is different; instead of reacting with adsorbed oxygen directly, CO_2 sensing relies on water vapor first reacting with adsorbed oxygen to form adsorbed hydroxide (OH^-). CO_2 then in turn reacts with the adsorbed hydroxide, forming carbonate (CO_3^{2-}). This process ultimately also returns electrons to the SnO_2 conduction band, and therefore results in increased conductivity. [Wang et al.]

SnO_2 gas sensors can also detect the presence of reducing gases such as NO_x . Here, the sensing mechanism is the opposite; the reducing gas adsorbs onto the sensor surface, consuming additional electrons from the conduction band, and thereby lowers the conductivity further. The MQ135 data sheet does not specify response curves for any reducing gases, so while the sensor should be sensitive to such gases, the user would have to experimentally determine the response curves themselves. [Shimizu]

MQ135 gas response curves

The MQ135 data sheet provides a graph showing how the sensor resistance changes with the concentration of several gases. Figure 1 shows a reproduction of the data sheet graph, where the points have been manually extracted from the data sheet graph using WebPlotDigitizer.

Note that the y-axis is R_s/R_0 , which is the ratio between the sensor resistance at any given moment (R_s) and the sensor resistance at a specific set of conditions (R_0). This is because the sensor resistance depends on the exact amount of active material on the sensor surface, which varies between individual sensors. Therefore, the resistance response must be normalized against a predefined state in order for results from different sensors to be comparable. For the MQ135, this predefined state is 100 ppm ammonia vapor in otherwise clean air at 20°C and 65% relative humidity. The sensor resistance at this state must be known in order for the curves in Figure 1 to apply. This may seem like a problem, since most people will not be able to reproduce those conditions. Fortunately, there are other ways to derive R_0 . More on that later.

The responses show clearly linear behavior in loglog-scale, which means that the

relationship between resistance and concentration can be expressed as:

$$R_S / R_0 = a * c^b \quad (1)$$

where a and b are constants, and c is gas concentration in ppm.

The curves in the Figure 1 have been fitted using exponential regression, specifically `scipy.optimize.curve_fit`.

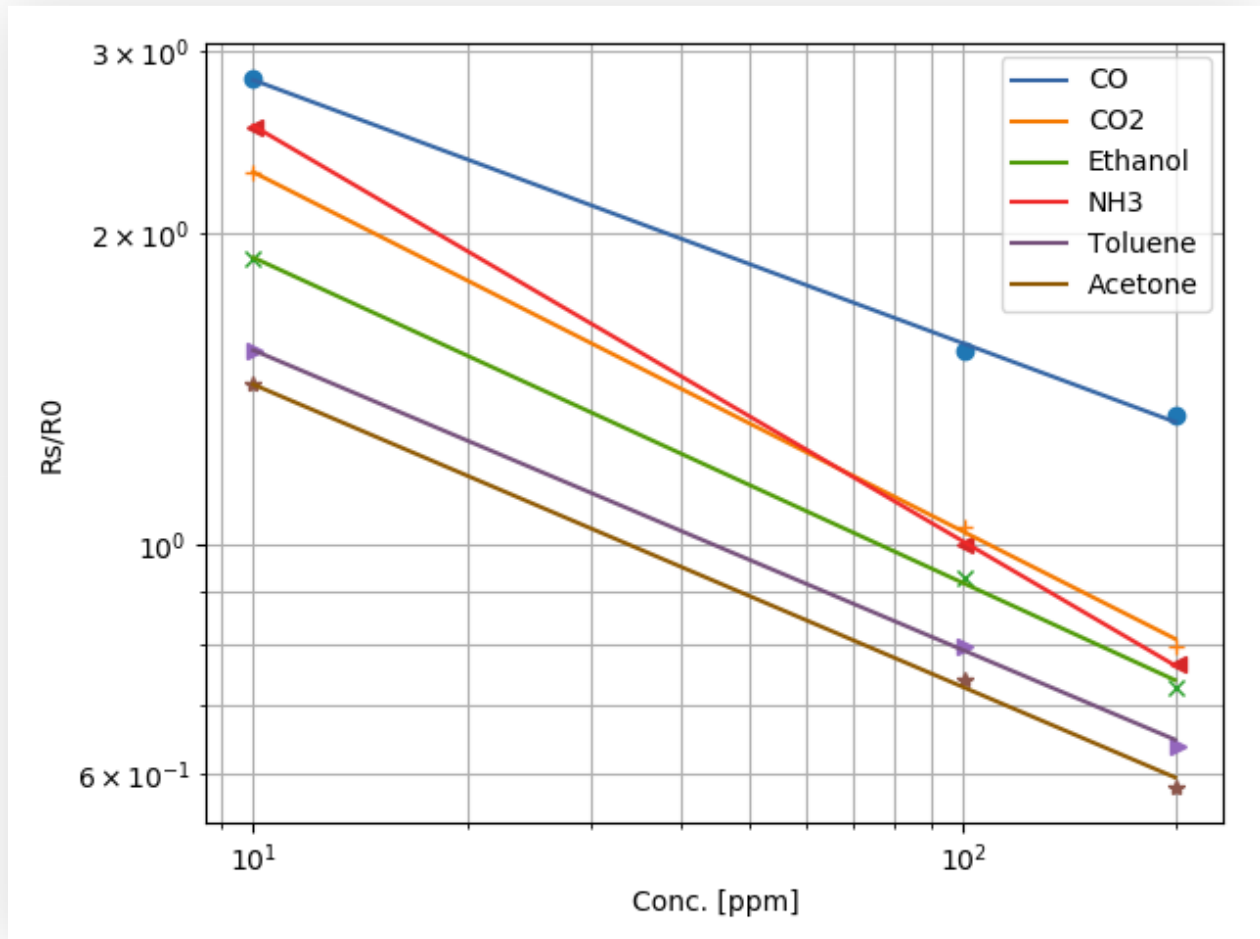


Figure 2: Data sheet gas response values (points) and fitted curves (lines).

Equation 1 gives the sensor resistance as a function of gas concentration, but we are interested in the opposite relation, i.e. gas concentration as a function of sensor resistance. A little elementary algebra gets us there:

$$c = (1 / a)^{1/b} * (R_S / R_0)^{1/b} = A * (R_S / R_0)^B \quad (2)$$

With that, we can correlate sensor resistance with gas concentration for any of the gases in Figure 1. Now, we need a way to determine the sensor resistance.

MQ135 electrical characteristics

As previously mentioned, the MQ135 behaves as a variable resistor when exposed to different gas concentrations. In order to measure gas concentration using equation 2, we need a way to measure the sensor resistance.

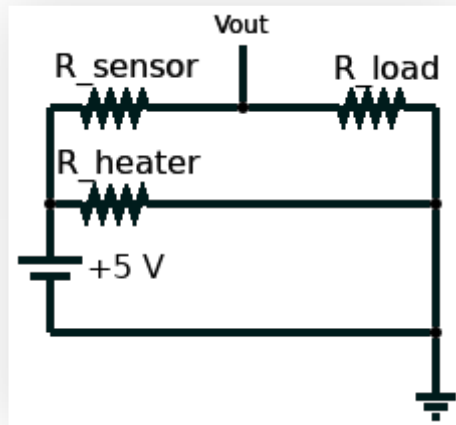


Figure 3: Circuit diagram of a typical sensing circuit using a MQ135. Note the voltage divider in the upper path.

By connecting a resistor in series with the sensor and measuring the voltage between them we can create a voltage divider, as in figure 3. The expression for the output voltage from a voltage divider is:

$$V_{out} = R_L / (R_S + R_L) * V_{in} \quad (3)$$

Rearrange to isolate the sensor resistance:

$$R_S = R_L * (V_{in} / V_{out} - 1) \quad (4)$$

Understanding R₀

While equation 4 provides a way to measure the sensor resistance, the sensor resistance alone is not enough to determine the gas concentration. This is because every individual sensor has a unique resistance response, based on the amount and distribution of active material on the sensor surface. Therefore, in order to compare the response of one sensor with other sensors, the response must be normalized. R_0 is the normalization factor, and is defined as the sensor resistance at a known set of conditions, i.e. a certain temperature, relative humidity, and most importantly a certain gas concentration. The ratio of the sensor resistance and R_0 can be compared between sensors.

The MQ135 data sheet defines R_0 as the sensor resistance at 100 ppm ammonia at 20 °C

and 65% relative humidity. Of course, it is possible to redefine R_0 at some other set of conditions, but if R_0 is redefined, the curves in Figure 2 no longer apply! The only situation where it makes sense to redefine R_0 is if you have the means to create new response curves for the gas you wish to measure. Otherwise, keeping R_0 defined as above is the best available option.

That means we need a way to calculate R_0 , since most people will not be able to recreate the conditions at which it is defined. By rearranging equation 2, we get:

$$R_0 = R_S * (1 / A * c)^{-1/B} \quad (5)$$

Since we can measure the sensor resistance, that means the only unknown term in the RHS of equation 5 is the concentration. In other words, if we can measure the sensor resistance in any known gas concentration, we can calculate R_0 .

As it happens, we do know the concentration of one of the gases in figure 2: Carbon dioxide. The concentration of CO_2 in outside air is around 400 ppm (and rising, check co2.earth for up to date values if you are reading this some years after 2021).

Accounting for temperature and humidity

The sensor response also depends on ambient temperature and humidity. The data sheet specifies the effect on sensor response at four different temperatures and two different humidity levels for ammonia vapor. Figure 4 shows a reproduction of the data sheet graph, plus fitted curves.

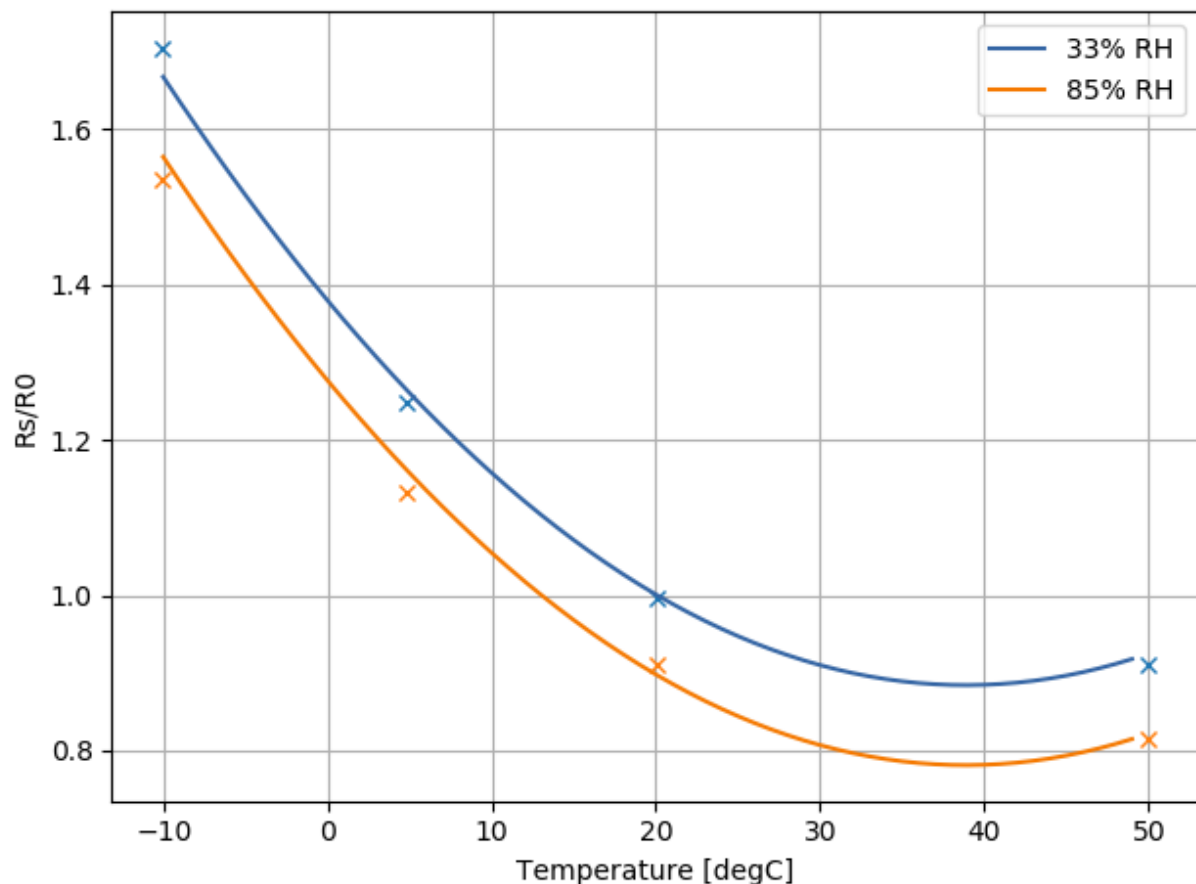


Figure 4: Data sheet temperature / humidity corrections (points) and fitter curves (lines).

From figure 4 we can see that the sensor resistance reaches a minimum at some temperature T_M and then starts to increase. The reason the resistance decreases as the temperature increases toward T_M is that higher surface temperature contributes to faster reaction rates between the gas and the adsorbed oxygen. The increase in sensor resistance at temperatures above T_M can be explained by faster re-adsorption of oxygen at higher temperatures, which means more oxygen will cover the sensor surface for the same flammable gas concentration. [Shimizu]

We can also see that sensor resistance seems to decrease with at higher relative humidity. The mechanism for this is less clear. On the one hand, water vapor can adsorb onto the sensor surface, thereby decreasing the oxygen coverage and increasing sensor conductivity. On the other hand, water vapor can heal defective adsorption sites, which increases the surface area available for oxygen adsorption and therefore decreases sensor conductivity. [Wicker et al.] Additionally, recall that the CO_2 sensing mechanism is highly dependent on the presence of water. Wang et al. reports a non-linear sensor response to CO_2 with respect to humidity, with a response peak at 34% RH. With that in mind, the data sheet humidity dependence graph (which only has two data points and therefore must be assumed to be linear) is of limited utility when measuring CO_2 .

With that, we have all the pieces we need to use the MQ135 to measure CO_2 concentration. Or any other of the gases in figure 2, for that matter. Enough theory, let's

take a look at how to do it in practice using the Pocket Science Lab!

Using the MQ135 with PSLab

If you have just the sensor itself, connect it as shown in figure 5, where the sensor is represented by a SparkFun gas sensor breakout board. CH1 is used to monitor the output voltage of the voltage divider between the sensor and the load resistor (which is 22K in this example circuit). Additionally, CH2 is used to monitor the voltage level of the PSLab's +5V line. It is important that the voltage on the +5V line does not drop below 4.9 V, as the MQ135 will not reach its operating temperature otherwise. The MQ135 heater resistor draws up to 200 mA of current, which may be more power than the USB port powering the PSLab can supply. The PSLab does not regulate the +5V line, so it is up to the USB port to deliver enough current.

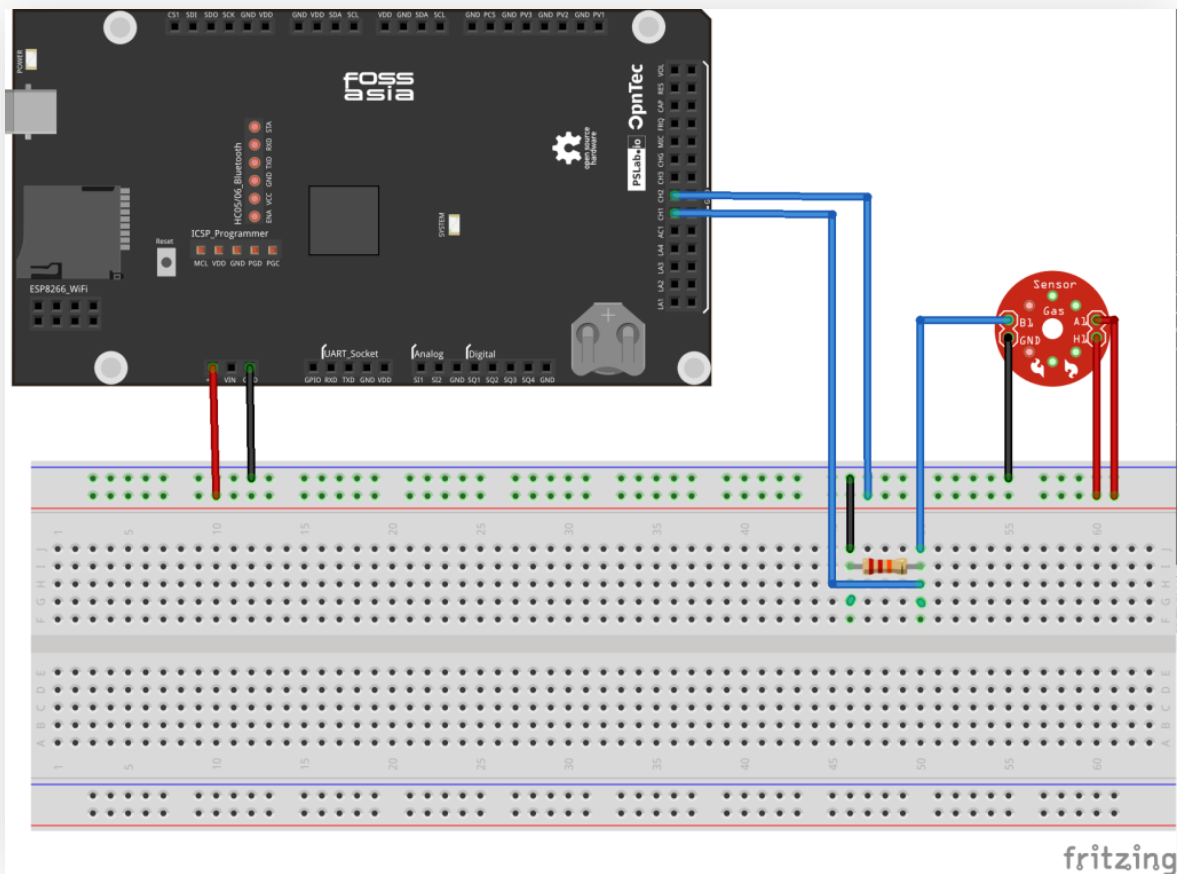


Figure 5: Guide to connect MQ135 to PSLab.

If you are using a SNS-MQ135 board, shown in figure 6, simply connect the AO pin to CH1, without the external load resistor. The SNS-MQ135 has its own load resistor, which unfortunately is only 1K. Such a small load resistor means the output voltage on AO will be very small when measuring CO₂, which limits

measurement accuracy. If possible, you should desolder the built-in SNS-MQ135 load resistor (outlined in red in figure 6), and use an external load resistor as in figure 5 instead. The caveat about the +5V level also applies to the SNS-MQ135.

The MQ135 must be allowed to heat to its operating temperature before it can be used. The data sheet recommends a heating period of at least 24h before use. Additionally, if the sensor hasn't been used for a long time (or ever, if it is new), it must be allowed to burn in for even longer (>72h) in order to burn off various substances which have reacted with the surface during that time.

After the MQ135 has been allowed to heat up and burn in properly, we can use pslab-python 2.1.0+ to measure the ambient CO₂ concentration.

Start by creating a multimeter to measure the voltage level on the +5V line:

```
>>> import pslab
>>> multimeter =
pslab.Multimeter()
>>>
multimeter.measure_voltage("CH2"
)
4.961014
```

If the voltage is at least 4.9 V, the next step is to determine the sensor's R0. It is a good idea to use an average value over a period of time (this only needs to be done once for each sensor):

```
>>> import time
>>> from pslab.external.gas_sensor import MQ135
>>> mq135 = MQ135(gas="CO2", r_load=22e3)
>>> r0 = []
>>> for i in range(3600):
>>>     r0.append(mq135.measure_r0(400))
>>>     time.sleep(1)
>>> mq135.r0 = sum(r0) / len(r0)
```

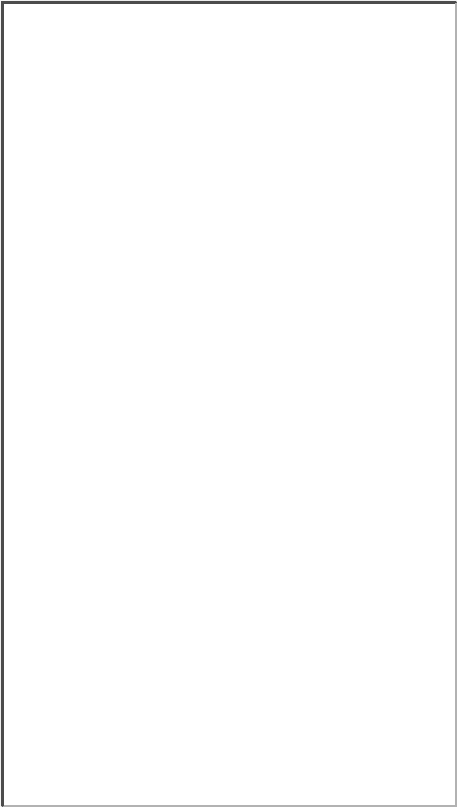


Figure 6: A MQ135 mounted on a PCB with a built-in load resistor (highlighted in red), as well as some other circuitry.

Finally:

```
>>> mq135.measure_concentration()  
418.067514310052683
```

Done!

References

Shimizu Y. (2014) SnO₂ Gas Sensor. In: Kreysa G., Ota K., Savinell R.F. (eds) Encyclopedia of Applied Electrochemistry. Springer, New York, NY. https://doi.org/10.1007/978-1-4419-6996-5_475

Wang et al. (2016) CO₂-sensing properties and mechanism of nano-SnO₂ thick-film sensor, Sensors and Actuators B: Chemical, Volume 227, Pages 73-84, ISSN 0925-4005, <https://doi.org/10.1016/j.snb.2015.12.025>

Wicker et al. (2017) Ambient Humidity Influence on CO Detection with SnO₂ Gas Sensing Materials. A Combined DRIFTS/DFT Investigation, The Journal of Physical Chemistry C, Volume 121, Number 45, Pages 25064-25073, <https://doi.org/10.1021/acs.jpcc.7b06253>

← PREVIOUS

pslab-python 2.1 Release Notes