

## 23. Wind Speed, Pressure, & Humidity Sensors

### Introduction:

Three more things that can be useful to know are wind speed, air pressure, and humidity. Wind speed is important for airplane flight controls, for evaluating a site for wind turbines, or for evaluating if conditions are ripe for fungus growth. Air pressure is again needed for aircraft flight controls, for predicting weather, and can also tell you wind speed with a pitot tube (coming shortly). Humidity gives you an idea about how comfortable a given temperature is. Some equipment and plants also required the humidity be kept within a certain range.

In this lecture, we'll look at

- Measuring wind speed with a fan, a hot-wire anemometer, and a pitot tube.
- Measuring air pressure using a BMP280 digital sensor, and
- Measuring humidity using a BMP280, DHT11, and DHT22 digital sensors.

There are many other ways to measure each of these - but this gives you an idea of some of the methods available to you.

### Wind Speed: Fan

One way to measure wind speed is to use a fan. From duality:

- The faster you spin a fan, the faster with breeze you produce,
- The faster the breeze driving a fan, the faster the fan will spin

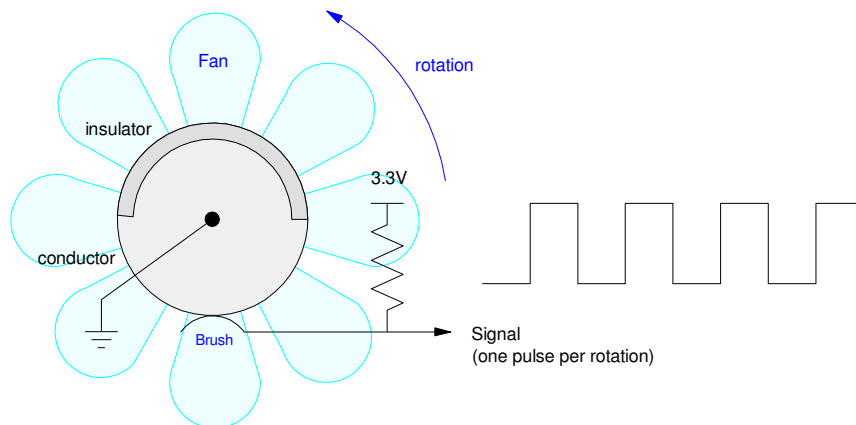
Essentially, remove the motor from a fan and you have a wind speed sensor.

Sensors of this type can be non-directional (the direction of the wind doesn't matter) or directional (the sensor must be perpendicular to the wind). In either case, the result is the same: if you can measure the speed of a fan, you can measure the wind's speed.



Two types of wind speed sensors: non-directional (left) and directional (right)

A simple way to convert rotational speed to a digital signal a microcontroller can read is to output a square wave with one or more pulses per rotation. By measuring the frequency (or period) of the square wave, you have a measurement of fan speed (and hence wind speed).



By outputting N pulses per rotation, the speed of the fan (and the speed of the wind) can be measured

In code, you can use methods from lecture #1 to measure

- The time of the positive pulse on pin GP0 (x):
- The time of the negative pulse on GP0 (y), or
- The period (z)

```
from machine import Pin, time_pulse_us

Fan = Pin(0, Pin.IN, Pin.PULL_UP)

while(1):
    x = time_pulse_us(0, 1, 100_000)
    y = time_pulse_us(0, 0, 100_000)
    z = x + y
    print(x, y, x+y)
```

```
10183 10517 20700
10382 10466 20848
10186 10492 20678
:
```

## Wind Speed: Hot-Wire Anemometer

With a thermistor, you can measure air speed. The idea is as follows...

Assume you have a thermistor measuring air temperature.

- The measured reading is actually the temperature of the thermistor, not the air
- When you apply current to a thermistor, it warms up due to  $I^2R$  heating
- The amount it warms up is related to the dissipation factor: 3.5mW/K in this case

For the previous example, the self heating at 25C will be:

$$I = \frac{3.3V}{R+R_0} = \frac{3.3V}{2000\Omega} = 1.65mA$$

$$P = I^2R = (1.65mA)^2 \cdot 1000\Omega$$

$$P = 2.7225mW$$

The self-heating will then be

$$T = \frac{2.7225mW}{3.5mW/K} = 0.778K$$

i.e. the reading you get will be high by 0.778 degrees C (or K)

This is when there is no breeze. As the air flow across the thermistor increases, the temperature cools down to ambient (air) temperature. So, measuring the self heating is a measure of air speed.

In order to measure the self heating, two thermistors and two voltage dividers are used:

- One has a low resistance in order to increase the self-heating
- The other has a large resistance in order to decrease the self-heating.

For example, assume you have two thermistors:

- R0 is 100 Ohms at 25C (meaning more current and more self heating), and
- R1 is 100k Ohms at 25C (meaning 1000x less current and self heating)
- each with a B-value of 3930:

$$R_0 = 100 \cdot \exp\left(\frac{3930}{T+273} - \frac{3930}{298}\right) \Omega$$

$$R_1 = 100k \cdot \exp\left(\frac{3930}{T+273} - \frac{3930}{298}\right) \Omega$$

Assume two voltage dividers with matching resistors (100 and 100k Ohms respectively). Without self-heating, the voltage difference should be zero. With self-heating, the 100 Ohm thermistor will dissipate 27.225W whereas the 100k resistor dissipated 27.225uW

$$P_0 = \frac{V^2}{R} = \frac{(1.65V)^2}{100} = 27.225mW$$

$$P_1 = \frac{V^2}{R} = \frac{(1.65V)^2}{100k} = 27.225\mu W$$

(note: at 25C the voltage divider outputs 1/2 of 3.3V or 1.65V). The self-heating will be:

$$\delta T_0 = \frac{27.225mW}{3.5mW/K} = 7.778K$$

$$\delta T_1 = \frac{27.225\mu W}{3.5mW/K} = 0.00778K$$

This self-heating means the 100 Ohm thermistor will be warmer than the 100k thermistor:

$$T_0 = 25C + \delta T_0 = 32.778C$$

$$T_1 = 25C + \delta T_1 = 25.00778C$$

resulting in the resistance being different:

$$R_0 = 71.501\Omega \quad (100 \text{ Ohms nominal})$$

$$R_1 = 99.965k\Omega \quad (100k \text{ Ohms nominal})$$

and the voltage being different:

$$V_0 = 1.3758V$$

$$V_1 = 1.6497V$$

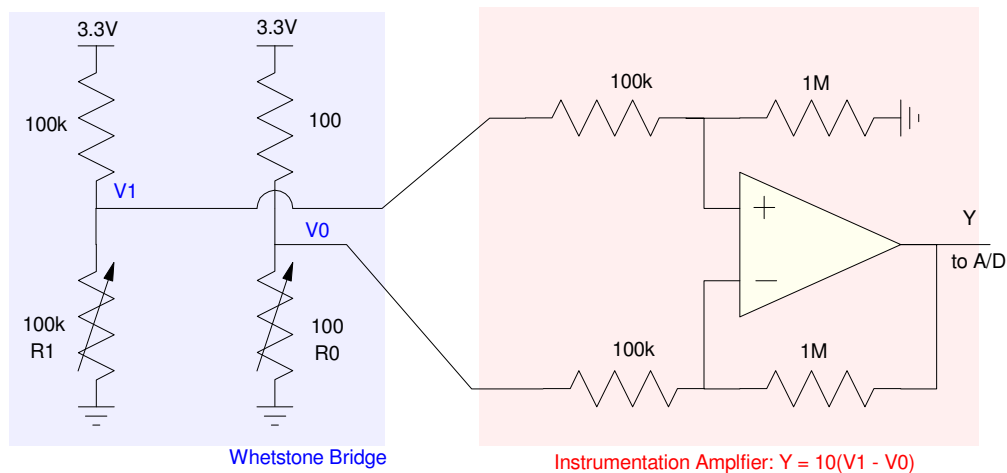
The voltage difference is a measure of air speed:

$$dV = V_1 - V_0 = 0.2739V$$

As the air speed increases

- you get more cooling, meaning
- the temperature (and voltage) difference decreases.

A voltage difference of 0.2739V is very small. This can be amplified with an instrumentation amplifier as follows:



Hot-Wire Anemometer: A Wheatstone bridge outputs a voltage difference.  
An instrumentation amplifier takes the difference and amplifies it

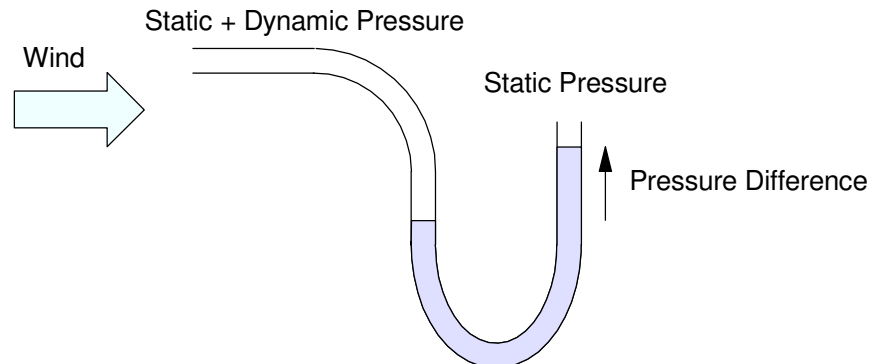
When the air speed is zero, the output should be 2.739V. As the air speed increases, this voltage decreases. With calibration, you then have a measure of air speed (i.e. a hot-wire anemometer).

## Wind Speed: Pitot Tube

A third way to measure wind speed is to measure

- The pressure of a tube going into the wind, and
- The pressure of a tube perpendicular to the wind

This is called a pitot tube and is a common way for aircraft to measure air speed.



Pitot Tube: Measure the pressure difference between a tube facing into the wind and one perpendicular to the wind

From Wikipedia, the total pressure (static plus dynamic) is

$$P_t = P_s + \left( \frac{\rho v^2}{2} \right)$$

where  $\rho$  is the fluid density. Solving for velocity in terms of the pressure difference:

$$v = \sqrt{\frac{2(P_t - P_s)}{\rho}}$$

or

$$v = \left( \sqrt{\frac{2}{\rho}} \right) (\Delta P)^{1/2}$$

By measuring the pressure difference, you have a measurement of speed.

To illustrate this, assume you want to measure the velocity of a car over the range of 0 - 30 m/s (0 - 67 mph). The density of air at 20C at 1 atmosphere is 1.204 kg / m<sup>3</sup> (Wikipedia). The pressure difference at 30 m/s is

$$\Delta P = \frac{\rho v^2}{2} = \frac{1}{2} \left( 1.204 \frac{\text{kg}}{\text{m}^3} \right) \left( 30 \frac{\text{m}}{\text{s}} \right)^2$$

$$\Delta P = 541.8 \frac{\text{kg}}{\text{m} \cdot \text{s}^2}$$

Adjusting the units:

$$\Delta P = 541.8 \left( \frac{\text{kg} \cdot \text{m}}{\text{s}^2} \right) \left( \frac{1}{\text{m}^2} \right)$$

$$\Delta P = 541.8 \left( \frac{N}{m^2} \right) = 541.8 \text{ Pa}$$

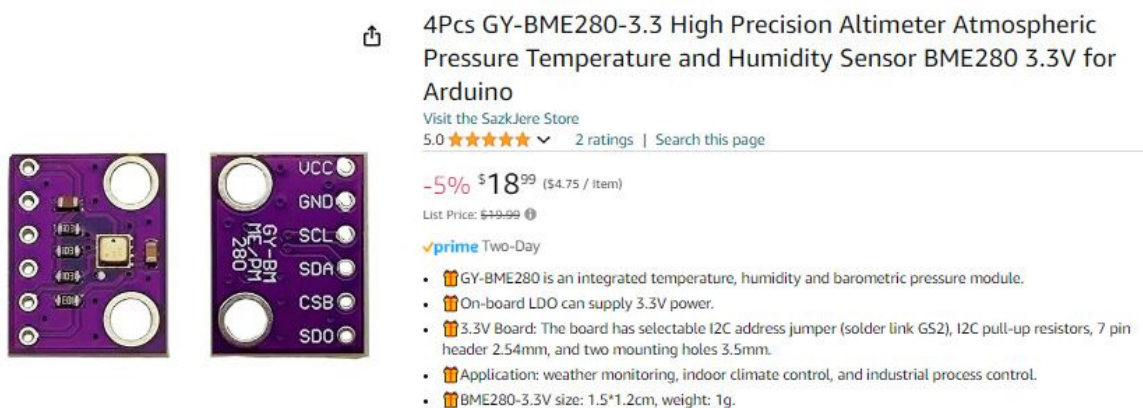
or

$$\Delta P = 5.418 \text{ hPa}$$

One atmosphere is 101,325 Pa or 1013.25 hPa. If you can measure a pressure change of 5.418 hPa, you can measure speed by measuring dynamic pressure. (Note: this method works better the faster you're traveling. Hence, it's use with aircraft rather than with vehicles.)

## Digital Pressure Sensor (BME280, BMP280)

<https://randomnerdtutorials.com/raspberry-pi-pico-bme280-micropython/>



4Pcs GY-BME280-3.3 High Precision Altimeter Atmospheric Pressure Temperature and Humidity Sensor BME280 3.3V for Arduino

Visit the SazkJere Store  
5.0 ★★★★★ 2 ratings | Search this page

-5% \$18<sup>99</sup> (\$4.75 / Item)  
List Price: ~~\$19.99~~

prime Two-Day

- GY-BME280 is an integrated temperature, humidity and barometric pressure module.
- On-board LDO can supply 3.3V power.
- 3.3V Board: The board has selectable I2C address jumper (solder link GS2), I2C pull-up resistors, 7 pin header 2.54mm, and two mounting holes 3.5mm.
- Application: weather monitoring, indoor climate control, and industrial process control.
- BME280-3.3V size: 1.5\*1.2cm, weight: 1g.

There are several ways to measure pressure as well. From Digikey or Amazon, you can find analog pressure sensors: sensors which output a voltage proportional to

- Absolute pressure (pressure relative to zero),
- Differential pressure (pressure difference between A and B), or
- Gage pressure (pressure relative to one atmosphere).

You can also use a digital pressure sensor such as a BME280 or BMP280. These sensors measure

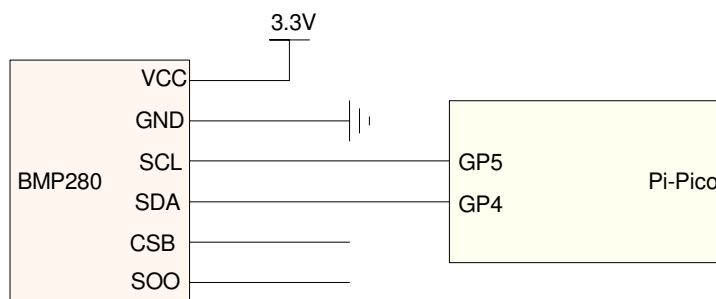
- Temperature (resolution 0.01C),
- Pressure (resolution 0.01 hPa), and
- Relative humidity (resolution 0.01%, BME280 only)

These sensors also support I2C and SPI communications. Digital sensors are pretty easy to interface, so let's go with these.

One way to develop code for a sensor is to dig through the data sheets. Another is to do a web search and hope someone else has written drivers for these chips. Sure enough, someone has. To get the drivers,

- Go to <https://randomnerdtutorials.com/raspberry-pi-pico-bme280-micropython/>
- Download the BME280 library
- Download the BME280 driver code

Following the schematics for I2C communications (SPI left for a homework assignment)



BMP280 set up for I2C communications

and you're reading temperature, humidity, and pressure

- BMP280 has temperature and pressure (humidity reads as 0.00%)
- BME280 has temperature, humidity, and pressure

```
# Raspberry Pi Pico: BME280 Code
# Complete project details at
https://RandomNerdTutorials.com/raspberry-pi-pico-bme280-micropython/

from machine import Pin, I2C
from time import sleep
import BME280

# Initialize I2C communication
i2c = I2C(id=0, scl=Pin(5), sda=Pin(4), freq=10000)

while True:
    try:
        # Initialize BME280 sensor
        bme = BME280.BME280(i2c=i2c)

        # Read sensor data
        tempC = bme.temperature
        hum = bme.humidity
        pres = bme.pressure

        # Print sensor readings
        print('-----')
        print('Temperature: ', tempC)
        print('Humidity: ', hum)
        print('Pressure: ', pres)

    except Exception as e:
        # Handle any exceptions during sensor reading
        print('An error occurred:', e)

    sleep(5)
```

shell

```
Temperature: 21.64C
Humidity: 0.00%
Pressure: 986.70hPa
```

## Humidity Sensor (DHT11 and DHT22 Temperature and Humidity)

<https://docs.micropython.org/en/latest/esp8266/tutorial/dht.html>



2pcs DHT22/AM2302 Digital Humidity and Temperature Sensor Module for Arduino Raspberry Pi, Temp Humidity Gauge Monitor Electronic Practice DIY Replace SHT11 SHT15

Brand: Gowoops  
 4.4 ★★★★★ 331 ratings  
 50+ bought in past month

\$11.80

prime  
 FREE Returns

Coupon:  Apply 5% coupon Shop items > | Terms

Earn 5% back (\$0.59 in rewards) on the amount charged to your Prime Visa.

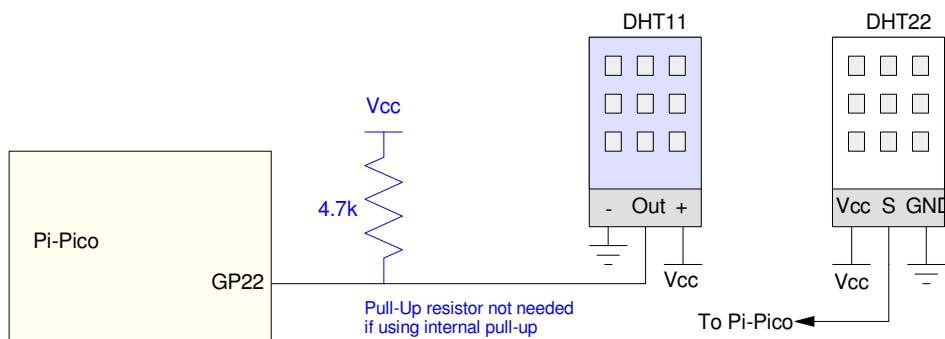
Brand: Gowoops  
 Graphics Card: Dedicated  
 Description:  
 Specific Uses For Product: Business  
 Screen Size: 2.6 Millimeters

DHT22 Digital Temperature and Humidity Sensor (Amazon)

Next, let's look at measuring relative humidity using a DHT11 (blue) or DHT22 (white) digital temperature and humidity sensor. These are sensors which operate over a range of

- 3.3V to 6.0V
- Relative Humidity (RH): 0% to 100% (2% accuracy)
- Temperature: -40C to +125C (0.2C accuracy)
- Resolution
  - 1C and 1% RH (DHT11)
  - 0.1C and 0.1% RH (DHT12)

Wiring of each of these sensors are similar: each needs power, ground, and a single wire back to the Pi-Pico for data transfer. The data line requires a pull-up resistor - but the internal pull-up resistor on the Pi-Pico can be used instead.

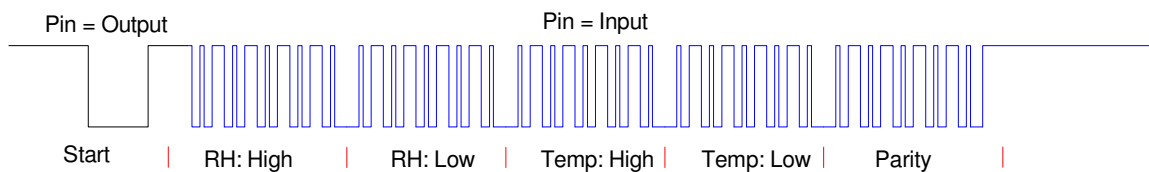


Wiring of a DHT11 or DHT22 sensor. The pull-up resistor can be replaced with an internal pull-up on a Pi-Pico

Communications over the single wire interface takes place as follows. First, the Pi-Pico sets its IO pin to output and sends a 500us low pulse followed by a 20-40us high pulse. This tells the DHT sensor to start a data conversion and return the previous temperature and humidity reading. (To get a current reading, a double-read needs to be executed).



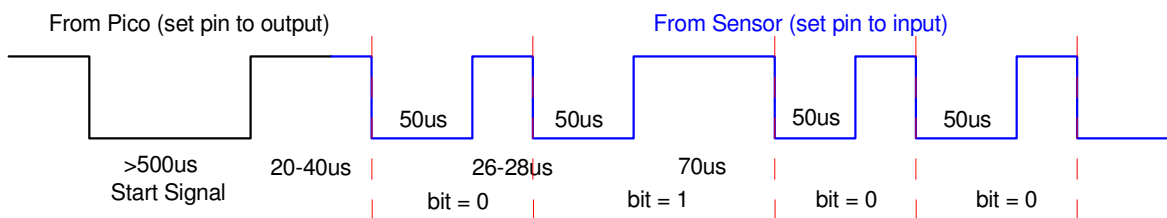
Once a start signal has been received, the DHT sensor responds with 40 bits, corresponding to the humidity (16-bit number), temperature (16-bit 2's compliment number), and a parity byte.



Each reading consists of five bytes (40 bits)

1's and 0's are encoded by the duration of the high pulse from the DHT sensor:

- Logic 0: 26-28us high pulse
- Logic 1: 70us high pulse



1's and 0's are determined by the pulse width of the high signal

With this, a low-level routine to read the relative humidity and temperature can be written. Since timing is critical, these routines would probably be C or assembler routines.

In Micro-Python, such routines have already been written and are part of the `dht` library - making these sensors *much* easier to use. The contents of the `dht` library can be found as follows:

```
>>> import dht
>>> help(dht)

object <module 'dht' from 'dht.py'> is of type module
  DHTBase -- <class 'DHTBase'>
  __version__ -- 0.1.0
  DHT11 -- <class 'DHT11'>
  DHT22 -- <class 'DHT22'>

>> help(dht.DHT11)

object <class 'DHT11'> is of type type
  __module__ -- dbt
  humidity -- <function humidity at 0x20013c00>
  temperature -- <function temperature at 0x200137c0>
  __qualname__ -- DHT11
```

To read a DHT11 sensor, the following Python code works. Note that the results are integers.

```
# DHT11 (blue)
import dht, machine, time

d11 = dht.DHT11(machine.Pin(22))

for i in range(0,3):
    d11.measure()
    Temp = d11.temperature()
    RH = d11.humidity()
    print(i, 'RH = ', RH, '%   Temp = ', Temp, 'C')
    time.sleep(0.5)
```

shell

```
0 RH = 56 %   Temp = 25 C
1 RH = 58 %   Temp = 21 C
2 RH = 58 %   Temp = 21 C
```

To read a DHT22 sensor, a similar program works (note with the DHT22, the resolution is 0.1)

```
# DHT22 (white)
import dht, machine, time

d22 = dht.DHT22(machine.Pin(22))

for i in range(0,3):
    d22.measure()
    Temp = d11.temperature()
    RH = d22.humidity()
    print(i, 'RH = ', RH, '%   Temp = ', Temp, 'C')
    time.sleep(0.5)
```

shell

```
0 RH = 57.7 %   Temp = 21.2 C
1 RH = 61.7 %   Temp = 21.5 C
2 RH = 62.9 %   Temp = 21.6 C
```

Note that Digikey reports these two sensors as being obsolete. You get more readings with an additional decimal place of accuracy, using a standard interface (I2C or SPI) with the BME280. But then, the Pi-Pico comes with the DHT libraries already installed.

## Summary

Measuring wind speed, air pressure, or humidity is also fairly easy with a Pi-Pico. Especially if someone else has already written the drivers.

## References

Pi-Pico and MicroPython

- [https://github.com/geekpi/pico\\_breakboard\\_kit](https://github.com/geekpi/pico_breakboard_kit)
- [https://micropython.org/download/RPI\\_PICO/](https://micropython.org/download/RPI_PICO/)
- <https://learn.pimoroni.com/article/getting-started-with-pico>
- <https://www.w3schools.com/python/default.asp>
- <https://docs.micropython.org/en/latest/pyboard/tutorial/index.html>
- <https://docs.micropython.org/en/latest/library/index.html>

- <https://www.fredscave.com/02-about.html>

#### Pi-Pico Breadboard Kit

- <https://wiki.52pi.com/index.php?title=EP-0172>

#### Other

- <https://docs.sunfounder.com/projects/sensorkit-v2-pi/en/latest/>
- <https://electrocredible.com/raspberry-pi-pico-external-interrupts-button-micropython/>
- <https://peppe8o.com/adding-external-modules-to-micropython-with-raspberry-pi-pico/>
- <https://randomnerdtutorials.com/projects-raspberry-pi-pico/>
- <https://randomnerdtutorials.com/projects-esp32-esp8266-micropython/>