

# COMUNICACIONES: BUSES DE CONTROL

---

USO DE LA RPI COMO CONTROLADOR LÓGICO PROGRAMABLE (CLP)

presentado por:

Francisco Javier Ferrández Pastor

# COMUNICACIONES

## USO DE LA RPI COMO CLP

---

- La RPI tiene un conjunto de recursos y posibilidades de desarrollo a un coste bajo.
- El hardware de control y comunicación que embarca puede ser utilizado desde diferentes lenguajes (C, Python) para desarrollar aplicaciones de comunicación, monitorización, control y servicios específicos.
- A destacar: puertos serie, puertos usb, interfaz de comunicación I2C y SPI, puerto ethernet, puerto de control I/O, acceso en bus para cámara.
- Con estos interfaces se pueden plantear desarrollos con diferentes niveles de comunicación: bus de campo, bus de control, red e Industrial Internet of Things.

# COMUNICACIONES

## USO DE LA RPI COMO CLP

---

- En el momento de plantearnos un proyecto de algún tipo de tecnología de comunicación industrial, a cualquier nivel de la pirámide de automatización podemos utilizar la RPI como control lógico para realizar prototipos y adquirir experiencia y conocimientos en el aprovechamiento de las tecnologías objeto de estudio.
- Los pasos a seguir son:
- Realizar búsqueda en plataformas de proyectos rpi (<https://www.hackster.io>, <https://www.raspberrypi.org>, o búsqueda orientada en google).

# COMUNICACIONES: MODBUS

USO DE LA RPI COMO CLP

---

- Como primer ejemplo: vamos a instalar en la RPI una librería que nos ayude a trabajar con el protocolo MODBUS.
- La primera duda a resolver es si tenemos las interfaces hardware necesarias para comunicarnos con las especificaciones de dicho protocolo.
- En nuestro caso la respuesta es que disponemos de un conversor USB-Serial 485 compatible con la RPI.
- El hardware serial 485 es un soporte físico con el que modbus puede trabajar. Por lo tanto tenemos lo necesario para trabajar cn el protocolo MODBUS-485



# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

- Como ejemplo 1:
  - Instalar en la RPI una aplicación que nos ayude a trabajar con el protocolo MODBUS para controlar la comunicación en un bus serie
- La primera duda a resolver es si tenemos las interfaces hardware necesarias para comunicarnos con las especificaciones de dicho protocolo.
- En nuestro caso la respuesta es que disponemos de un conversor USB-Serial 485 compatible con la RPI. Como la RPI embarca USB, se puede conectar.
- El hardware serial 485 es un soporte físico con el que modbus puede trabajar. Por lo tanto tenemos todo lo necesario para trabajar con el protocolo MODBUS-485



# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

---

- RPI cuenta con proyectos desarrollados y aplicaciones relacionadas
- En nuestro caso vamos a utilizar la aplicación ofrecida en:  
<https://github.com/epsilonrt/mbpoll>
- En este link se encuentra el código para instalar el programa. Como no se dan instrucciones de instalación, debemos clonar a nuestro escritorio mediante el comando:

```
git clone https://github.com/epsilonrt/mbpoll
```

# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

- A continuación seguimos con comandos indicados

### Installation

The installation instructions are in the INSTALL file.

For example, for a debian system:

- Install dependencies:

```
sudo apt-get update  
sudo apt-get install cmake build-essential
```

- Generate Makefile with cmake:

```
cd mbpoll  
mkdir build  
cd build  
cmake ..
```

- Compile and install mbpoll:

```
make  
sudo make install
```

# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

- Una vez instalada la aplicación se comprueba su funcionamiento. En los SO Linux, el puerto serie COM tiene otra denominación que hay que conocer una vez se conecta el dispositivo USB-Serial.

### Examples

The following command is used to read the input registers 1 and 2 of the slave at address 33 connected through RTU /dev/ttyUSB2 (38400 Bd)

```
mbpoll -a 33 -b 38400 -t 3 -r 1 -c 2 /dev/ttyUSB2

mbpoll 0.1-10 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright (c) 2015 epsilonRT, All rights reserved.
This software is governed by the CeCILL license <http://www.cecill.info>

Protocol configuration: Modbus RTU
Slave configuration...: address = [33]
                        start reference = 1, count = 2
Communication.....: /dev/ttyUSB2, 38400-8E1
                        t/o 1.00 s, poll rate 1000 ms
Data type.....: 16-bit register, input register table

-- Polling slave 33... Ctrl-C to stop
[1]:      9997
[2]:      10034
-- Polling slave 33... Ctrl-C to stop
[1]:      10007
[2]:      10034
-- Polling slave 33... Ctrl-C to stop
[1]:      10007
[2]:      10034
-- Polling slave 33... Ctrl-C to stop
[1]:      10007
[2]:      10034
-- Polling slave 33... Ctrl-C to stop
[1]:      10007
[2]:      10034
^C--- /dev/ttyUSB2 poll statistics ---
4 frames transmitted, 4 received, 0 errors, 0.0% frame loss

everything was closed.
Have a nice day !
```

# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

- Lo único que queda es determinar que puerto utiliza nuestro Arduino cuando esta conectado al Raspberry Pi.
- Para ello antes de conectar el USB ejecutamos el siguiente comando  
`>> ls /dev/tty*`
- Y veremos que nos proporciona un listado de diferentes dispositivos. Luego simplemente conectamos el USB, volvemos a ejecutar el comando y veremos como aparecerá un dispositivo adicional.
- En caso de prueba aparece como: `/dev/ttyUSB0`
- Este es el puerto de entradas y salidas de tramas del protocolo. Como ejemplo se envía una petición trama al SLAVE 1 solicitando los dos primeros registros:  
`mbpoll -a 24 -b 9600 -m rtu -t 4 -r 1 -c 4 /dev/ttyUSB0`

# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

```
mbpoll -a 24 -b 9600 -m rtu -t 4 -r 1 -c 4 /dev/ttyUSB0
```

```
pi@raspberrypi:~ $ mbpoll -a 24 -b 9600 -m rtu -t 4 -r 1 -c 4 /dev/ttyUSB0
mbpoll 1.2-6 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright (c) 2015 epsilonRT, All rights reserved.
This software is governed by the CeCILL license <http://www.cecill.info>

Protocol configuration: Modbus RTU
Slave configuration...: address = [24]
                        start reference = 1, count = 4
Communication.....: /dev/ttyUSB0,      9600-8E1
                        t/o 1.00 s, poll rate 1000 ms
Data type.....: 16-bit register, output (holding) register table

-- Polling slave 24... Ctrl-C to stop)
[1]: 22738
[2]: 22749
[3]: 22706
[4]: 0
-- Polling slave 24... Ctrl-C to stop)
[1]: 22746
[2]: 22757
[3]: 22714
[4]: 0
-- Polling slave 24... Ctrl-C to stop)
[1]: 22699
[2]: 22710
[3]: 22667
[4]: 0
^C--- /dev/ttyUSB0 poll statistics ---
3 frames transmitted, 3 received, 0 errors, 0.0% frame loss

everything was closed.
Have a nice day !
pi@raspberrypi:~ $
```

# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

- Como ejemplo 2:
  - ❑ Instalar en la RPI librerías que nos ayude a trabajar con el protocolo MODBUS para controlar la comunicación en un bus serie
- La primera duda a resolver es si tenemos las interfaces hardware necesarias para comunicarnos con las especificaciones de dicho protocolo.
- En nuestro caso la respuesta es que disponemos de un conversor USB-Serial 485 compatible con la RPI. Como la RPI embarca USB, se puede conectar.
- El hardware serial 485 es un soporte físico con el que modbus puede trabajar. Por lo tanto tenemos todo lo necesario para trabajar con el protocolo MODBUS-485



# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

- Por regla general, la RPI cuenta con plataformas donde ya se han desarrollado librerías para facilitar la creación de aplicaciones.
- En nuestro caso vamos a instalar las librerías MINIMALMODBUS, LIBMODBUS y PYMODBUS

<http://minimalmodbus.readthedocs.io/en/master/apiminimalmodbus.html#>

<https://github.com/stephane/libmodbus/wiki/Libmodbus-on-Raspberry-pi>

<http://riptideio.github.io/pymodbus/>

- Estas librerías están suficientemente documentada para su uso. Muy importante para que la curva de aprendizaje y aprovechamiento sea lo más optimizada posible.

# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

- La instalación de librerías o software en RPI es sencillo y se ofrece desde diferentes opciones: descargar en formato comprimido, instalar a partir de comandos del SO (git clone) o mediante aplicaciones dedicadas (pip install)
- Para la descarga o instalación directa la RPI debe estar conectada a internet

# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

Raspberry Pi has gained popularity in recent days and a lot of enthusiasts / hobbyist would like to implement libmodbus on RPi to host a modbus master / slave. Here is "how to install libmodbus" guide on RPi (Raspbian).

### 1) Installing dependancies.

```
sudo apt-get install -y autoconf libtool
```

### 2) Clone libmodbus master branch by,

```
git clone https://github.com/stephane/libmodbus/
```

### 3) Enter the directory.

```
cd libmodbus
```

### 4) ./autogen.sh && ./configure --prefix=/usr && make && sudo make install

### 5) To test the libmodbus, try examples under tests folder.

To compile code, use `gcc filename.c -o filename $(pkg-config --libs --cflags libmodbus)`

#### Additional notes :

If you're using linux distros (other than debian based), such as Arch Linux or PiDora, you must use relevant package manager options in step #1. (like pacman or yum)

In order to use native RPi serial port for RTU mode, you MUST disable serial console by using `sudo raspi-config` as it conflicts with modes RTU communication.

Native serial port is listed as `ttyAMA0` under `/dev/` directory.

If you're using UART to RS485 chip like 75176 or anything similar, you would like to check the libmodbus raspberry pi fork which makes use of a GPIO to enable rx-tx. (Link :

# libmodbus

A Modbus library for Linux, Mac OS X, FreeBSD, QNX and Win32



[News](#) [Download](#) [Documentation](#) [Report a Bug](#)

# COMUNICACIONES: MODBUS

USO DE LA RPI COMO CLP

## Pymodbus

A full modbus protocol written in python

[View on GitHub](#)

[Download .zip](#)

[Download .tar.gz](#)

- En nuestro caso seguimos las instrucciones
- Importante: comprobar que la instalación es correcta y no hay mensajes de error en su realización

```
git clone git://github.com/bashwork/pymodbus.git  
cd pymodbus  
python setup.py install
```

# COMUNICACIONES: MODBUS

USO DE LA RPI COMO CLP

## Pymodbus

A full modbus protocol written in python

[View on GitHub](#)

[Download .zip](#)

[Download .tar.gz](#)

- Una vez instalada la librería se analizan programas de ejemplo sencillos que utilizan la librería y que sean útiles para nuestras intenciones
- En nuestro ejemplo queremos que la RPI actúe como servidor MODBUS que controle el bus serie RS485

# COMUNICACIONES: MODBUS

USO DE LA RPI COMO CLP

## Pymodbus

A full modbus protocol written in python

[View on GitHub](#)

[Download .zip](#)

[Download .tar.gz](#)

The screenshot shows the PyModbus documentation page. At the top, there's a dark teal header with the title "Pymodbus" and a subtitle "A full modbus protocol written in python". Below the header are three buttons for GitHub, ZIP, and TAR.GZ downloads. The main content area has a white background. On the left, there's a sidebar with a blue header "PyModbus latest" and a search bar. The sidebar contains a "CONTENTS:" section with links to various PyModbus versions and features like "Summary", "Features", "Use Cases", and "Examples". At the bottom of the sidebar is a "linode" logo with text about Linux cloud hosting. The main content area starts with a "Welcome to PyModbus's document" header. Below it is a "Contents:" section with a hierarchical list of PyModbus components and versions. The entire page is framed by a thick teal border.

Welcome to PyModbus's document

**Contents:**

- [Summary](#)
- [Features](#)
  - Client Features
  - Server Features
- [Use Cases](#)
  - Example Code
  - Installing
  - Current Work In Progress
  - Development Instructions
  - Contributing
  - License Information
- [Version 1.4.0](#)
- [Version 1.3.2](#)
- [Version 1.3.1](#)
- [Version 1.3.0.rc2](#)
- [Version 1.3.0.rc1](#)
- [Version 1.2.0](#)
- [Version 1.1.0](#)
- [Version 1.0.0](#)
- [Version 0.9.0](#)
- [Examples](#)
  - Asynchronous Client Example
  - Asynchronous Processor Example
  - Asynchronous Server Example

# COMUNICACIONES: MODBUS

USO DE LA RPI COMO CLP

## Pymodbus

A full modbus protocol written in python

[View on GitHub](#)

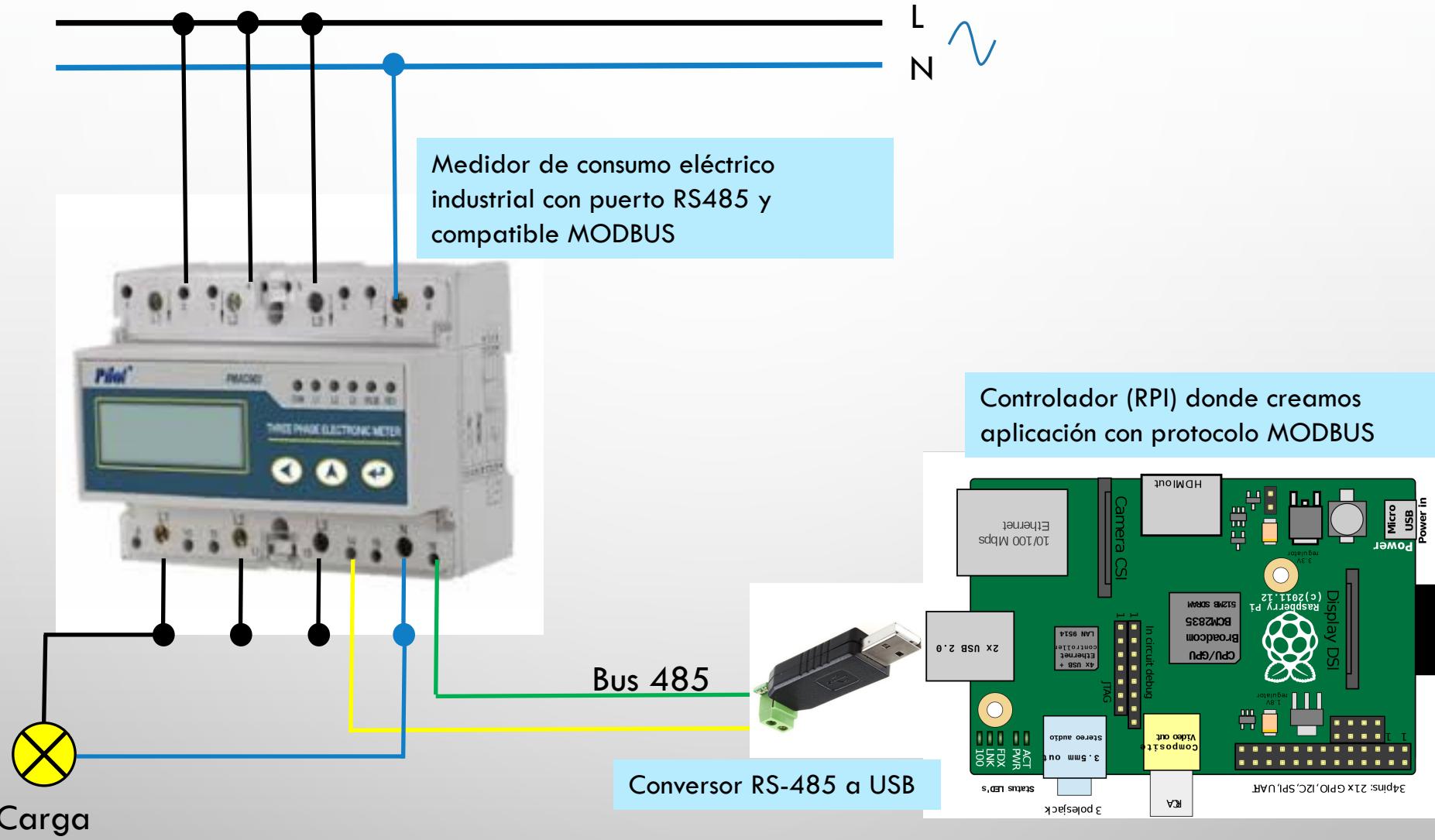
[Download .zip](#)

[Download .tar.gz](#)

- Hay un código ya implementado para una aplicación de un servidor MODBUS

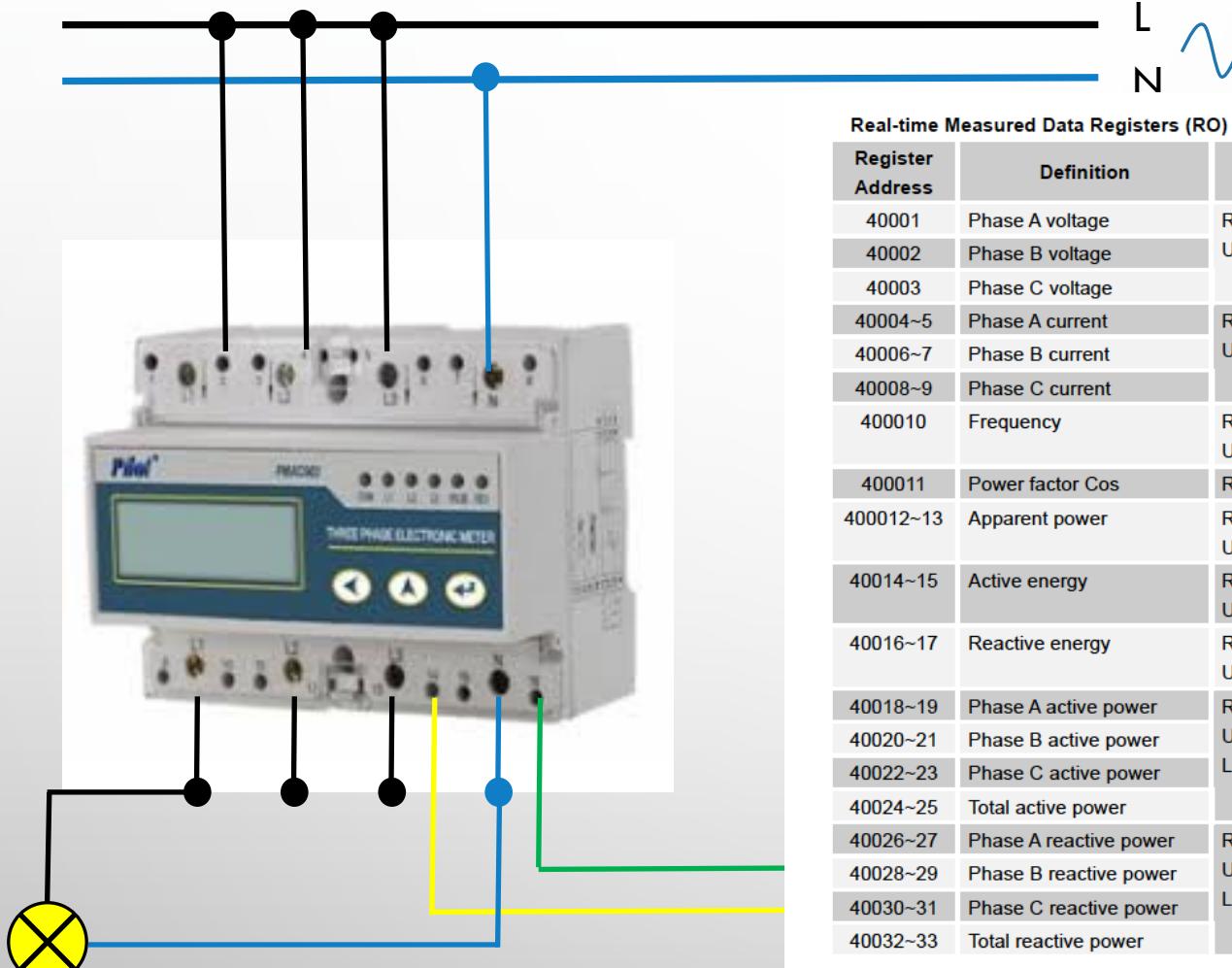
# COMUNICACIONES: MODBUS

USO DE LA RPI COMO CLP



# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP



Real-time Measured Data Registers (RO)

Register Address	Definition	Remarks
40001	Phase A voltage	Reading is the value magnified by 100 times. Unit is V
40002	Phase B voltage	
40003	Phase C voltage	
40004~5	Phase A current	Reading is the value magnified by 1000 times. Unit is A
40006~7	Phase B current	
40008~9	Phase C current	
400010	Frequency	Reading is the value magnified by 100 times. Unit is Hz
400011	Power factor Cos	Reading is the value magnified by 1000 times.
400012~13	Apparent power	Reading is the value magnified by 100 times. Unit is VA. Low 16 bit is at front (40012)
40014~15	Active energy	Reading is the value magnified by 100 times. Unit is KWH. Low 16 bit is at front (40014)
40016~17	Reactive energy	Reading is the value magnified by 100 times. Unit is KVARH. Low 16 bit is at front (40016)
40018~19	Phase A active power	Reading is the value magnified by 100 times.
40020~21	Phase B active power	Unit is W
40022~23	Phase C active power	Low 16 bit is at front
40024~25	Total active power	
40026~27	Phase A reactive power	Reading is the value magnified by 100 times.
40028~29	Phase B reactive power	Unit is VA
40030~31	Phase C reactive power	Low 16 bit is at front
40032~33	Total reactive power	

Carga

# COMUNICACIONES: MODBUS

USO DE LA RPI COMO CLP

Aplicación en Python que utiliza la librería MINIMALMODBUS para leer el primer HOLDING register del PMAC903, medidor de consumo energético trifásico

```
#!/usr/bin/env python
import minimalmodbus
minimalmodbus.BAUDRATE = 9600

medidor = minimalmodbus.Instrument('/dev/ttyUSB0', 24) # port name, slave address

## Read voltage ##
V1 = medidor.read_register(0, 2,3) # Registernumber, number of decimals
print V1

## Change temperature setpoint (SP) ##
##NEW_TEMPERATURE = 95
#instrument.write_register(2, NEW_TEMPERATURE, 1) # Registernumber, value, number
```

# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

Aplicación extendida que obtiene información de varios registros del medidor

```
#!/usr/bin/env python
import minimalmodbus
minimalmodbus.BAUDRATE = 9600
medidor = minimalmodbus.Instrument('/dev/ttyUSB0', 24) # port name, slave ad

## Read voltage holding registers##
V1 = medidor.read_register(0, 0,3) # Registernumber, number of decimals
print "La tension en la fase 1 es...", V1
V2 = medidor.read_register(1, 0,3) # Registernumber, number of decimals
print "La tension en la fase 2 es...", V2
V3 = medidor.read_register(2, 0,3) # Registernumber, number of decimals
print "La tension en la fase 3 es...", V3
## Change temperature setpoint (SP) ##
##NEW_TEMPERATURE = 95
#instrument.write_register(24, NEW_TEMPERATURE, 1) # Registernumber, value,
```

```
pi@raspberrypi:~ $ python prueba3.py
La tension en la fase 1 es... 23022
La tension en la fase 2 es... 23034
La tension en la fase 3 es... 22990
pi@raspberrypi:~ $ python prueba3.py
La tension en la fase 1 es... 22996
La tension en la fase 2 es... 23032
La tension en la fase 3 es... 22989
pi@raspberrypi:~ $ python prueba3.py
La tension en la fase 1 es... 23045
La tension en la fase 2 es... 23056
La tension en la fase 3 es... 23012
pi@raspberrypi:~ $ python prueba3.py
La tension en la fase 1 es... 23052
La tension en la fase 2 es... 23063
La tension en la fase 3 es... 23019
```

# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

Aplicación extendida, con bucle, que obtiene información de varios registros del medidor

```
#!/usr/bin/env python
import minimalmodbus
import time
minimalmodbus.BAUDRATE = 9600
medidor = minimalmodbus.Instrument('/dev/ttyUSB0', 24) # port name, slave address !
## Read voltage holding registers ##
while 1:
    V1 = medidor.read_register(0, 0,3) # Registernumber, number of decimals
    print "La tension en la fase 1 es...", V1
    V2 = medidor.read_register(1, 0,3) # Registernumber, number of decimals
    print "La tension en la fase 2 es...", V2
    V3 = medidor.read_register(2, 0,3) # Registernumber, number of decimals
    print "La tension en la fase 3 es...", V3
    print "-----"
    time.sleep(5)
## Change temperature setpoint (SP) ##
##NEW_TEMPERATURE = 95
#instrument.write_register(24, NEW_TEMPERATURE, 1) # Registernumber, value, number
```

```
pi@raspberrypi:~ $ python prueba4.py
La tension en la fase 1 es... 23025
La tension en la fase 2 es... 23036
La tension en la fase 3 es... 22992
-----
La tension en la fase 1 es... 22897
La tension en la fase 2 es... 22908
La tension en la fase 3 es... 22865
-----
La tension en la fase 1 es... 22938
La tension en la fase 2 es... 22949
La tension en la fase 3 es... 22906
-----
La tension en la fase 1 es... 23069
La tension en la fase 2 es... 23080
La tension en la fase 3 es... 23036
-----
La tension en la fase 1 es... 23128
La tension en la fase 2 es... 23139
La tension en la fase 3 es... 23095
-----
```

# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

Aplicación extendida, con bucle, que obtiene información de varios registros del medidor y captura fecha y hora de la RPI

```
#!/usr/bin/env python
import minimalmodbus
import time
minimalmodbus.BAUDRATE = 9600
medidor = minimalmodbus.Instrument('/dev/ttyUSB0', 24) # port name, slave address (in decimal)

## Read voltage holding registers ##
while 1:
    ahora=time.strftime("%c")
    print ahora
    print "-----"
    V1 = medidor.read_register(0, 0,3) # Registernumber, number of decimals
    print "La tension en la fase 1 es...", V1
    V2 = medidor.read_register(1, 0,3) # Registernumber, number of decimals
    print "La tension en la fase 2 es...", V2
    V3 = medidor.read_register(2, 0,3) # Registernumber, number of decimals
    print "La tension en la fase 3 es...", V3
    print "-----"
    time.sleep(5)

## Change temperature setpoint (SP) ##
##NEW_TEMPERATURE = 95
#instrument.write_register(24, NEW_TEMPERATURE, 1) # Registernumber, value, num

Sun Feb 18 11:42:02 2018
La tension en la fase 1 es... 23107
La tension en la fase 2 es... 23119
La tension en la fase 3 es... 23074
-----
Sun Feb 18 11:42:07 2018
La tension en la fase 1 es... 23101
La tension en la fase 2 es... 23112
La tension en la fase 3 es... 23068
-----
Sun Feb 18 11:42:12 2018
La tension en la fase 1 es... 22956
La tension en la fase 2 es... 22967
La tension en la fase 3 es... 22923
```

# COMUNICACIONES: MODBUS

## USO DE LA RPI COMO CLP

Aplicación extendida, con bucle, que obtiene información de varios registros del medidor y captura fecha y hora de la RPI, además de detectar cambios en el consumo

```
#!/usr/bin/env python
import minimalmodbus
import time
import math
minimalmodbus.BAUDRATE = 9600
medidor = minimalmodbus.Instrument('/dev/ttyUSB0', 24) # port name
I1_anterior=0.0

## Read voltage holding registers ##
while 1:
    ahora= time.strftime("%c")
    print ".....CICLO....."
    V1 = medidor.read_register(0, 2,3) # Registernumber, number of bytes
    print "La tension en la fase 1 es...", V1, "V"
    V2 = medidor.read_register(1, 2,3) # Registernumber, number of bytes
    print "La tension en la fase 2 es...", V2, "V"
    V3 = medidor.read_register(2, 2,3) # Registernumber, number of bytes
    print "La tension en la fase 3 es...", V3, "V"
    I11=float(medidor.read_register(4, 0,3))
    I12=medidor.read_register(3, 0,3)
    I1=((I11*65536)+I12)/1000
    E11=float(medidor.read_register(14, 0,3))
    E12=float(medidor.read_register(13, 0,3))
    E1=((E11*65536)+E12)/100
    print "La corriente en la fase 1 es...", I1, "A"
    print "La energia consumida de las fases es...", E1, "KWH"
    print ahora
    if math.fabs(I1-I1_anterior)>0.05:
        print "CAMBIO DE CONSUMO"
    I1_anterior=I1
    print ".....FIN CICLO ....."
    print ".....FIN CICLO ....."
    time.sleep(5)
## Change temperature setpoint (SP) ##
```

```
.....CICLO.....  
La tension en la fase 1 es... 228.18 V  
La tension en la fase 2 es... 228.3 V  
La tension en la fase 3 es... 227.87 V  
La corriente en la fase 1 es... 0.1 A  
La energia consumida de las fases es.. 1.96 KWH  
Sun Feb 18 16:15:20 2018  
.....FIN CICLO .....
```

```
.....CICLO.....  
La tension en la fase 1 es... 227.9 V  
La tension en la fase 2 es... 228.02 V  
La tension en la fase 3 es... 227.59 V  
La corriente en la fase 1 es... 0.1 A  
La energia consumida de las fases es.. 1.96 KWH  
Sun Feb 18 16:15:25 2018  
.....FIN CICLO .....
```

```
.....CICLO.....
```

# COMUNICACIONES: EN-OCEAN (PROTOCOLO RF)

## USO DE LA RPI COMO CLP

---

La Radio Frecuencia tiene bandas libres (frecuencias) para el desarrollo de aplicaciones Industriales y de investigación.

La Radio Frecuencia ofrece ventajas frente a otros tipo de buses tanto inalámbrico en bandas muy saturadas como las cableadas.

Uno de los buses RF más robustos utilizados en inmótica (automatización de edificios) como en domótica (automatización de viviendas) es el protocolo en-Ocean.

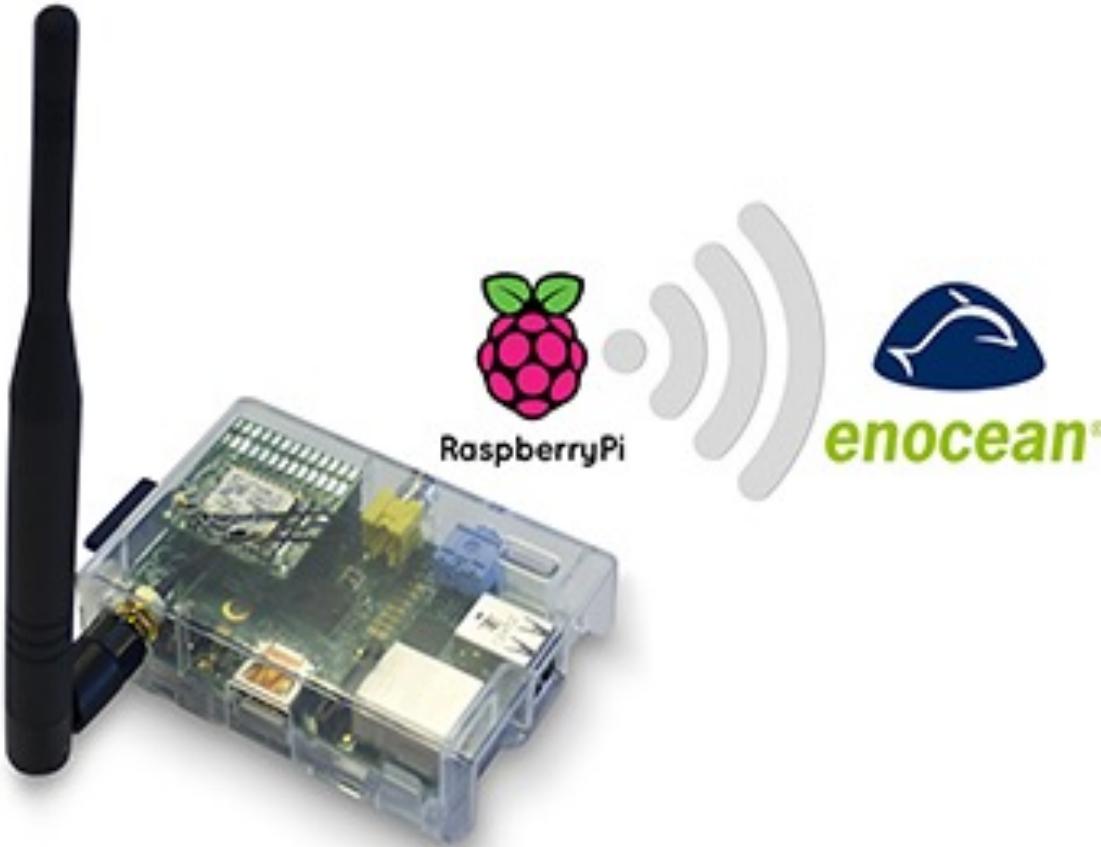
Es un protocolo que utiliza como interfaz el puerto serie del puerto I/O de la Raspberry o bien el puerto USB de la propia Rasberry.

Sobre este protocolo inalámbrico en RF interesará saber cómo gestionar la información de los telegramas RF que los dispositivos emiten cuando hay un evento o información a enviar.

La interpretación del protocolo serie es más complejo que el desarrollado por MODBUS, por lo que su uso a nivel práctico será conocer cómo configurar la comunicación y cómo capturar las tramas RF enviadas por los dispositivos al CLP.

# COMUNICACIONES: EN-OCEAN (PROTOCOLO RF)

USO DE LA RPI COMO CLP

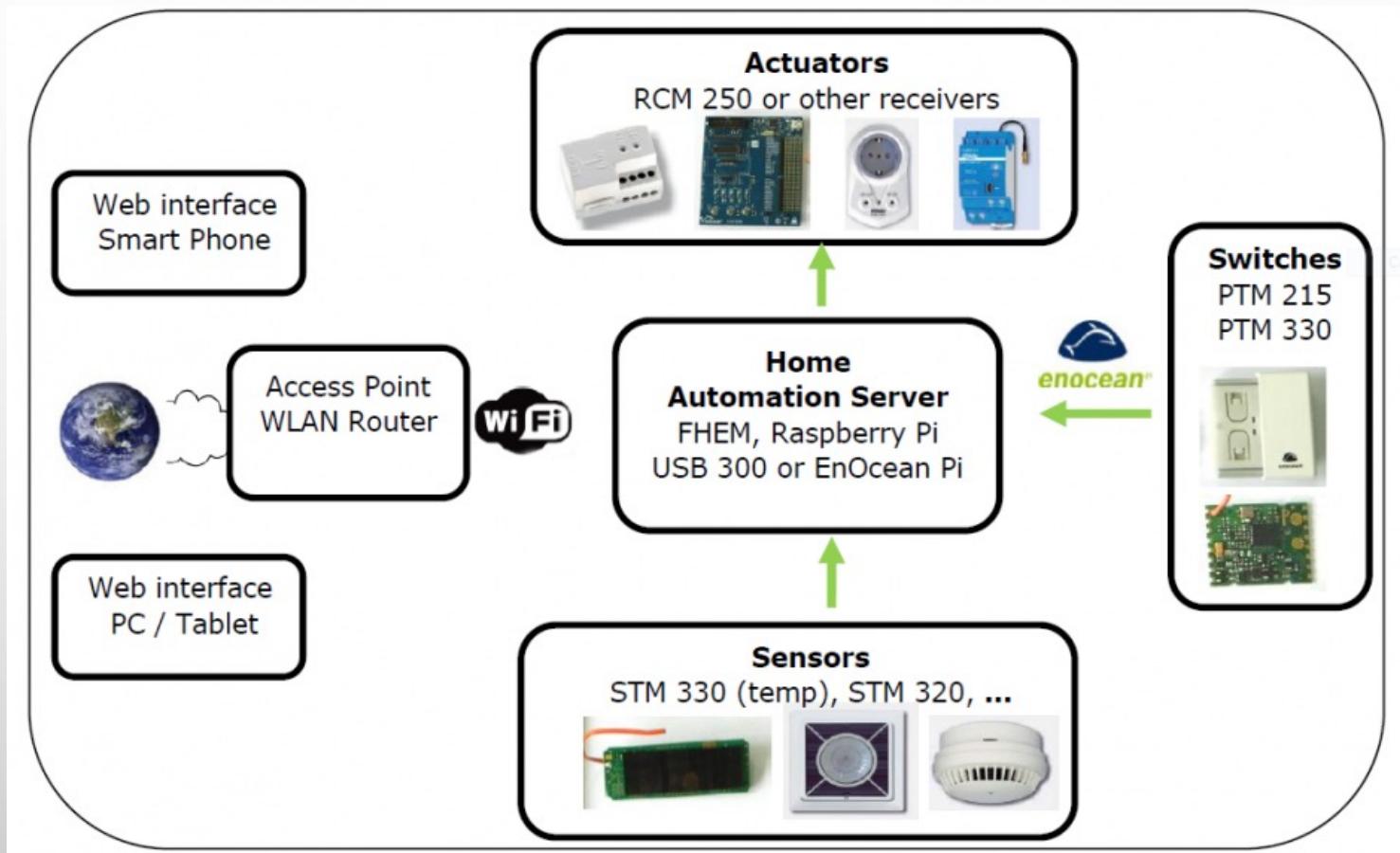


Enocean PI  
Connect your Raspberry Pi  
to Enocean World



# COMUNICACIONES: EN-OCEAN (PROTOCOLO RF)

USO DE LA RPI COMO CLP



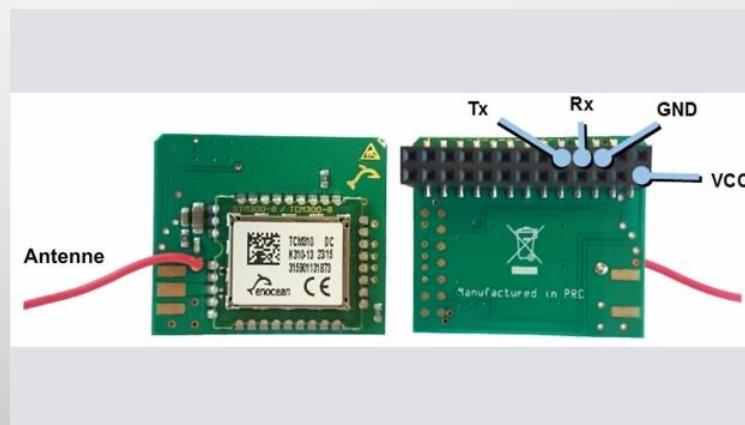
# COMUNICACIONES: EN-OCEAN (PROTOCOLO RF)

## USO DE LA RPI COMO CLP

De igual modo que para la instalación de MODBUS, para el protocolo ENOCEAN hay una guía de instalación para utilizar la RaspberryPi como CLP. Hay dos posibilidades de conexión: usar un USB o bien conectarse al puerto serie del puerto mediante la I/O de la Raspberry.

### ■ Como ejemplo 3:

- Instalar en la RPI librerías y recursos para utilizarla como pasarela para dispositivos del protocolo inalámbrico En-Ocean utilizando el puerto serie del I/O



# COMUNICACIONES: EN-OCEAN (PROTOCOLO RF)

## USO DE LA RPI COMO CLP

En el enlace:

<https://www.element14.com/community/roadTestReviews/1825/l/raspberry-pi-and-enocean-internet-of-things-pack-review>

Se indican los comandos necesarios. Una vez instalada se cuenta con un servicio web de conexión a los dispositivos, donde se extrae la información de todos. También se pueden utilizar dos comandos para listar los dispositivos enlazados y para capturar las tramas enviadas por dichos dispositivos

```
pi@raspberrypi:~ $ /opt/fhem/fhem.pl localhost:7072 list
Type list <name> for detailed info.

Global:
  global          (<no definition>)

TCM:
  TCM310_0        (opened)

FHEMWEB:
  FHEMWEB:192.168.1.113:58902 (Connected)
    WEB            (Initialized)
    WEBphone       (Initialized)
    WEBtablet      (Initialized)

EnOcean:
  En0_sensor_0002065A (7)
  En0_switch_00137A4F (A0)

eventTypes:
  eventTypes     (active)

notify:
  initialUsbCheck (active)

FileLog:
  FileLog_En0_sensor_0002065A (active)
  FileLog_En0_switch_00137A4F (active)
  Logfile          (active)

autocreate:
  autocreate     (active)

telnet:
  telnet:127.0.0.1:43262 (Connected)
    telnetPort      (Initialized)
```

```
pi@raspberrypi:~ $
pi@raspberrypi:~ $
pi@raspberrypi:~ $ /opt/fhem/fhem.pl localhost:7072 "list En0_switch_00137A4F STATE"
En0_switch_00137A4F AI

pi@raspberrypi:~ $ /opt/fhem/fhem.pl localhost:7072 "list En0_switch_00137A4F STATE"
En0_switch_00137A4F B0

pi@raspberrypi:~ $ /opt/fhem/fhem.pl localhost:7072 "list En0_switch_00137A4F STATE"
En0_switch_00137A4F BI

pi@raspberrypi:~ $ /opt/fhem/fhem.pl localhost:7072 "list En0_switch_00137A4F STATE"
En0_switch_00137A4F AI

pi@raspberrypi:~ $ /opt/fhem/fhem.pl localhost:7072 "list En0_switch_00137A4F STATE"
En0_switch_00137A4F A0

pi@raspberrypi:~ $ /opt/fhem/fhem.pl localhost:7072 "list En0_switch_00137A4F STATE"
En0_switch_00137A4F A0,B0

pi@raspberrypi:~ $ /opt/fhem/fhem.pl localhost:7072 "list En0_switch_00137A4F STATE"
En0_switch_00137A4F AI,BI
```

# COMUNICACIONES: EN-OCEAN (PROTOCOLO RF)

## USO DE LA RPI COMO CLP

Programa en python que crea 3 comandos de lectura de dispositivo enOcean  
Para tres dispositivos diferentes cambiará la referencia “list EnO\_.xxxxxxxx”  
Para más dispositivos se incrementarán los comandos

```
from subprocess import Popen, PIPE
import shlex

comando1='/opt/fhem/fhem.pl localhost:7072 "list En0_switch_00137A4F STATE"'
comando2='/opt/fhem/fhem.pl localhost:7072 "list En0_switch_00137A4F STATE"'
comando3='/opt/fhem/fhem.pl localhost:7072 "list En0_switch_00137A4F STATE"'
comandos=[comando1,comando2,comando3]

for i in comandos:
    args=shlex.split(i)
    s=Popen(args,stdout=PIPE,stderr=PIPE)
    stdout, stderr=s.communicate()
    print stdout
```

# COMUNICACIONES PROTOCOLO SERIE

## USO DE LA RPI COMO CLP

---

### INTERFACES DE COMUNICACIÓN SERIE RASPBERRY-PI

El controlador RPI tiene diferentes interfaces de comunicación serie (TTL, USB, I2C y SPI) a las que se puede acceder a través de los diferentes protocolos que los definen.

#### SENSOR CO2 CON CONEXIÓN SERIE TTL

##### Antecedentes

Log into your Pi and run sudo raspi-config.

Go to **5 Interfacing Options**, then **P6 Serial**. Select **No** on the login shell and **Yes** on the serial port hardware.

##### Instalación de librerías PYTHON

```
sudo apt install python3-pip
```

```
pip3 install pyserial
```

##### Sensor setup

The sensor has three interfaces, analog output, PWM output and a 9600 baud 3V/5V TTL UART interface. You basically send a request and the sensor responds.

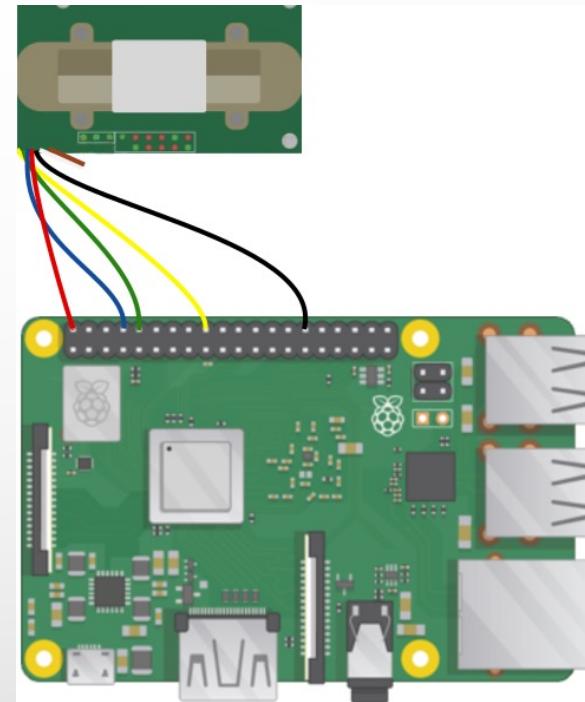
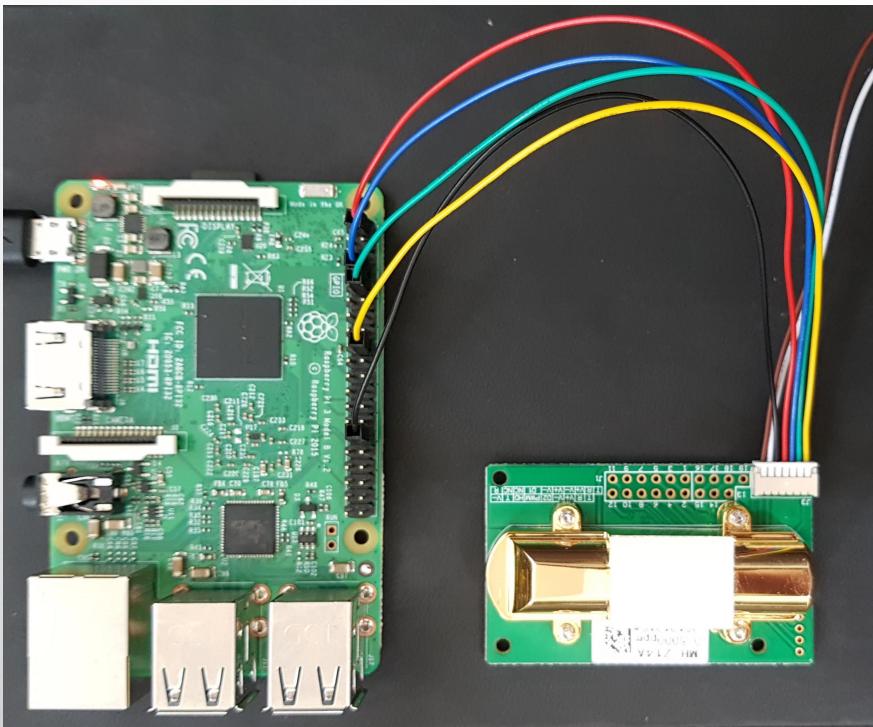
I manually rewired the cable that came with the sensor, to have female jumper DuPont wires to attach to the Pi's GPIO pins. You can also use F-M jumper wires and use the PWM interface instead.

The sensor pins are:

PWM (yellow) TXD (green) RXD (blue) VCC (red) GND (black) | AnalogOutput (white) HD (brown) and need to be attached to the Pi's pins like this:

# COMUNICACIONES TTL SERIE

USO DE LA RPI COMO CLP



# COMUNICACIONES TTL SERIE

## USO DE LA RPI COMO CLP

### Python code

```
#!/usr/bin/env python3
import serial
import time

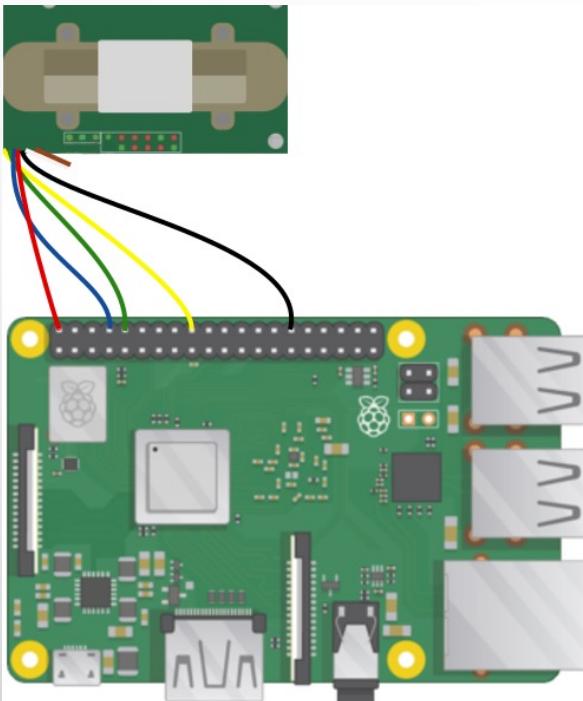
class CO2Sensor():
    request = [0xff, 0x01, 0x86, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79]

    def __init__(self, port='/dev/ttys0'):
        self.serial = serial.Serial(
            port = port,
            timeout = 1
        )

    def get(self):
        self.serial.write(bytarray(self.request))
        response = self.serial.read(9)
        if len(response) == 9:
            current_time = time.strftime('%H:%M:%S', time.localtime())
            return {"time": current_time, "ppa": (response[2] << 8) | response[3], "temp": response[4]}
        return -1

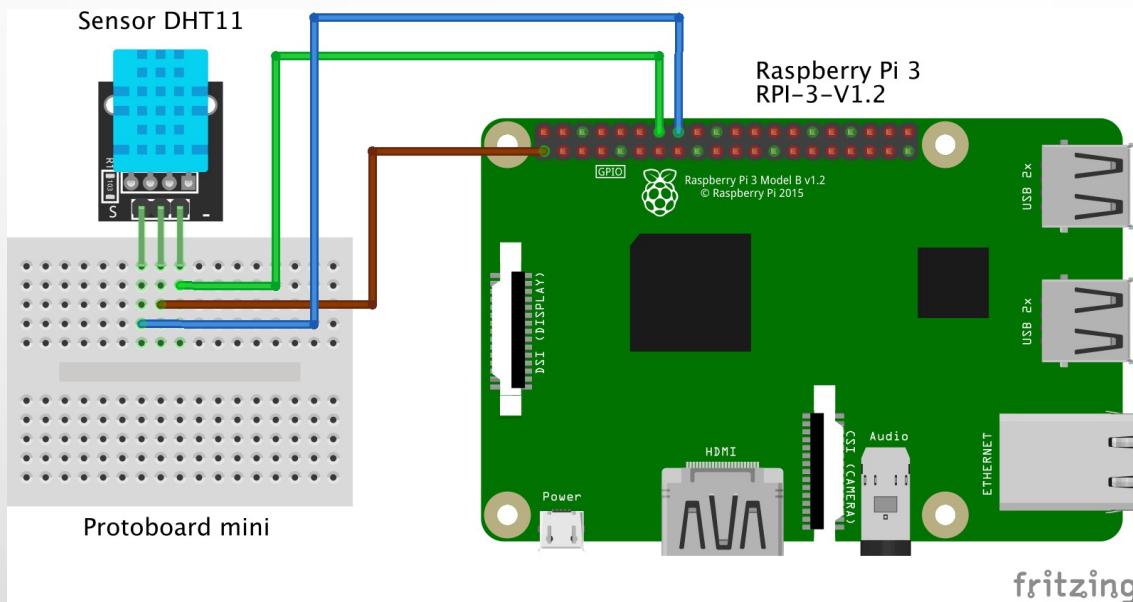
    def main():
        # other Pi versions might need CO2Sensor('/dev/ttys0')
        sensor = CO2Sensor()
        print(sensor.get())

        if __name__ == '__main__':
            main()
```



# COMUNICACIONES: PUERTO GPIO

## USO DE LA RPI COMO CLP



```
sudo pip3 install adafruit-circuitpython-dht  
mkdir proyectos cd proyectos  
git clone https://github.com/internetdelascosas/RaspberryPi-DHT11.git  
cd RaspberryPi-DHT11
```

# COMUNICACIONES: PUERTO GPIO

USO DE LA RPI COMO CLP

---

<https://camo.githubusercontent.com/a1ae4f445a0bc4c28904fadcc05d86b58099abcd/68747470733a2f2f7869616f6d692d6d692e636f6d2f75706c6f6164732f636b2f7869616f6d692d666c6f7765722d6d6f6e69746f722d3030312e6a7067>



<https://github.com/ChrisScheffler/miflora>