# Logbook

## MAN4HEALTH

João Paulo Coelho

*Ligação do regulador de carga da bateria ao Raspberry Pi e configuração do WiFi dongle*

**Outubro 2022**

# Conteúdo

# 1. Instalação do dongle WiFi

Este *logbook* documenta o processo de ligação do Raspberry Pi ao regulador de carga da bateria. Para além de publicar, via MQTT, os valores relacionados com o EPEVER, também será responsável por registar e publicar os valores provenientes da estação meteorológica SenseCAP. Para isso, é necessária uma ligação WiFi à gateway LoRa existente. Visto que o Raspberry Pi será instalado no interior de um armário metálico, procurou-se uma placa de rede WiFi que permitisse a ligação de uma antena exterior via conetores SMA. De entre as possibilidades avaliadas, optou-se pelo equipamento de marca ***aqprox***, versão APPUSB600DA como se mostra na Figura 1.

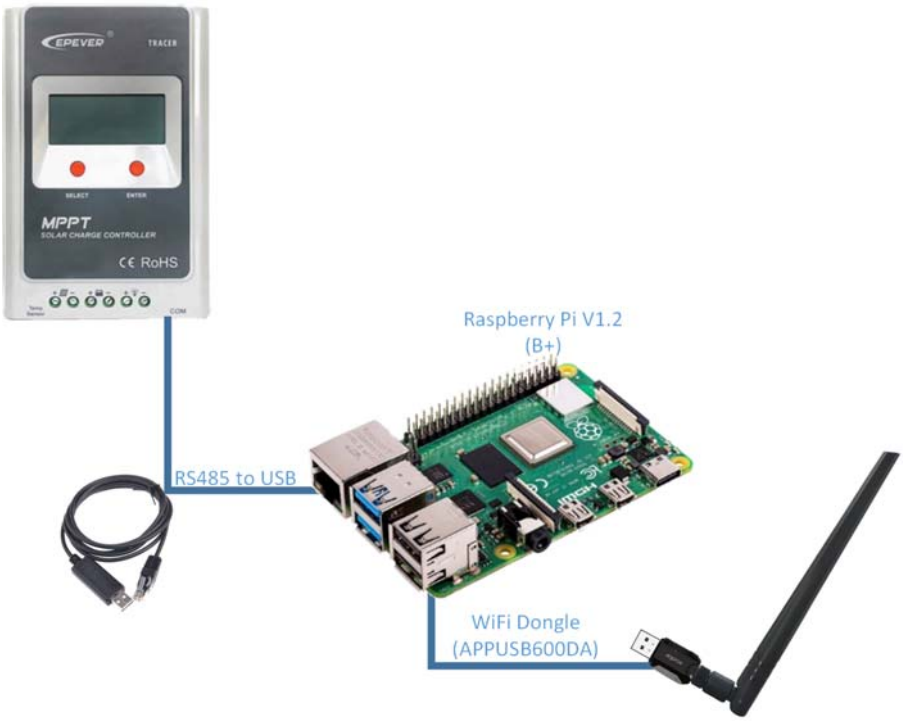



*Figura 1 - Ligação do Raspberry Pi ao EPVER e dongle WiFI*

Detalhes sobre este produto podem ser consultados na página do fabricante cujo link é:

https://approx.es/producto/appusb600da-adaptador-usb-wifi-600mb-antena/

A documentação, assim como os drivers, encontram-se nessa página e foram descarregados para memória futura.

Este dispositivo de rede WiFi será instalado no Raspberry Pi selecionado para ser integrado no quadro de controlo. Em particular, e dado que no momento em que escrevo este documento é impossível adquirir novas unidades do Raspberry Pi, será utilizada a versão 1.2 B+ deste dispositivo.

## 1.1. Raspberry Pi V1.2 (B+)

O Raspberry Pi utilizado neste projeto é constituído por um System-on-Chip manufaturado pela Micron com a referência BCM2708/BCM2835. As folhas de dados do BCM2835 podem ser descarregadas a partir de:

https://www.raspberrypi.org/app/uploads/2012/02/BCM2835-ARM-Peripherals.pdf

Existe algumas questões que se prendem com o fato de muitas vezes o SoC ser designado por BCM2708 e outras por BCM2835. Na verdade, a discrepância deve-se à designação do silício e do pacote do chip. Originalmente, a matriz de silício é conhecida como **BCM2708**. Ao ser adicionado, no mesmo encapsulamento, 256 MB de DRAM, passou a ser conhecido como **BCM2763**. Mas quando a DRAM é anexada ao topo do encapsulamento do processador passou a ser conhecido como **BCM2835**.

Neste ponto tenho instalado a seguinte versão do *kernel* nesse Raspberry Pi que posso determinar a parir da execução do seguinte comando:

```
$ uname –a
```

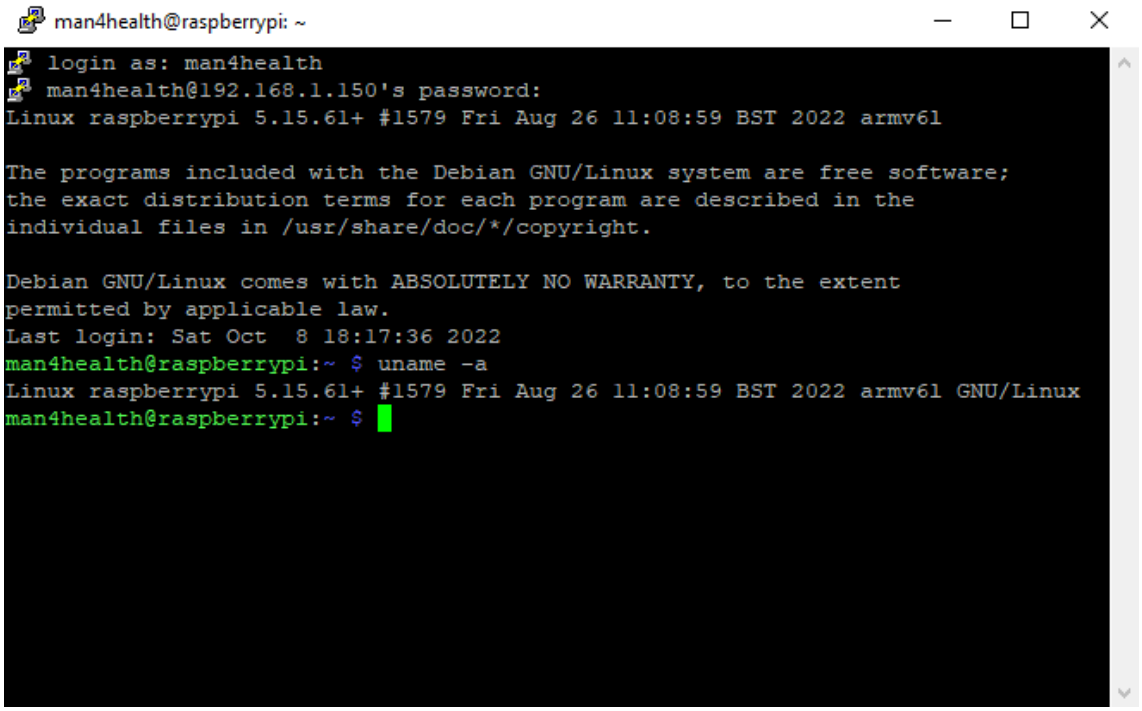



*Figura 2 - Resultado da execução do comando "uname -a".*

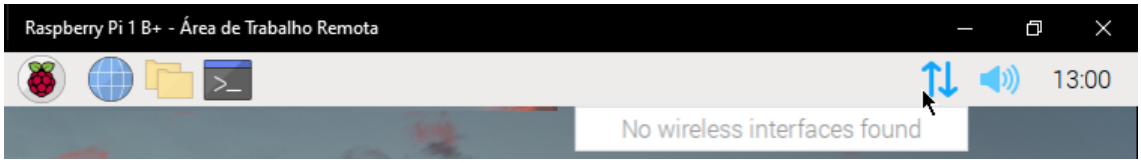
Que resultou no seguinte:

```
Linux raspberrypi 5.15.61+ #1579 Fri Aug 26 11:08:59 BST 2022 armv6l GNU/Linux
```

Este *kernel* pode ser descarregado a partir do repositório GitHub seguindo o endereço:

https://github.com/raspberrypi/linux/tarball/rpi-5.15.y

Ao contrário do que acontece com o Windows, a instalação do dongle WiFi não é "plug-and-play". Conetar o dongle a uma das quatro portas USB não resulta na instalação imediata e funcionamento do dispositivo.




No entanto, o dispositivo é reconhecido pelo Raspberry Pi como se pode constatar a partir da execução do comando:

```
$ lsusb
```

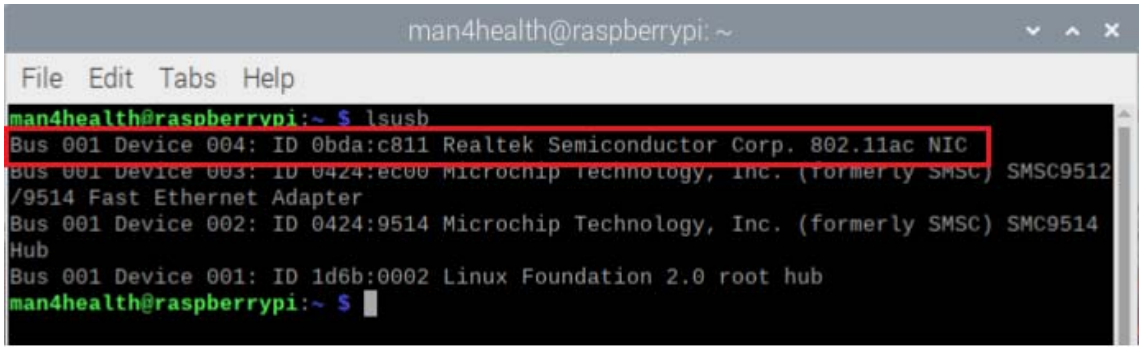



*Figura 3 - Resultado da execução de lsusb*

Oura forma consiste na instalação do hardinfo que permite analisar as especificações de todo o hardware do Raspberry Pi. A instalação do hardinfo é feita através de:

```
sudo apt-get install -y hardinfo
```

No final, executando:

```
$ hardinfo
```

Leva à seguinte interface gráfica:

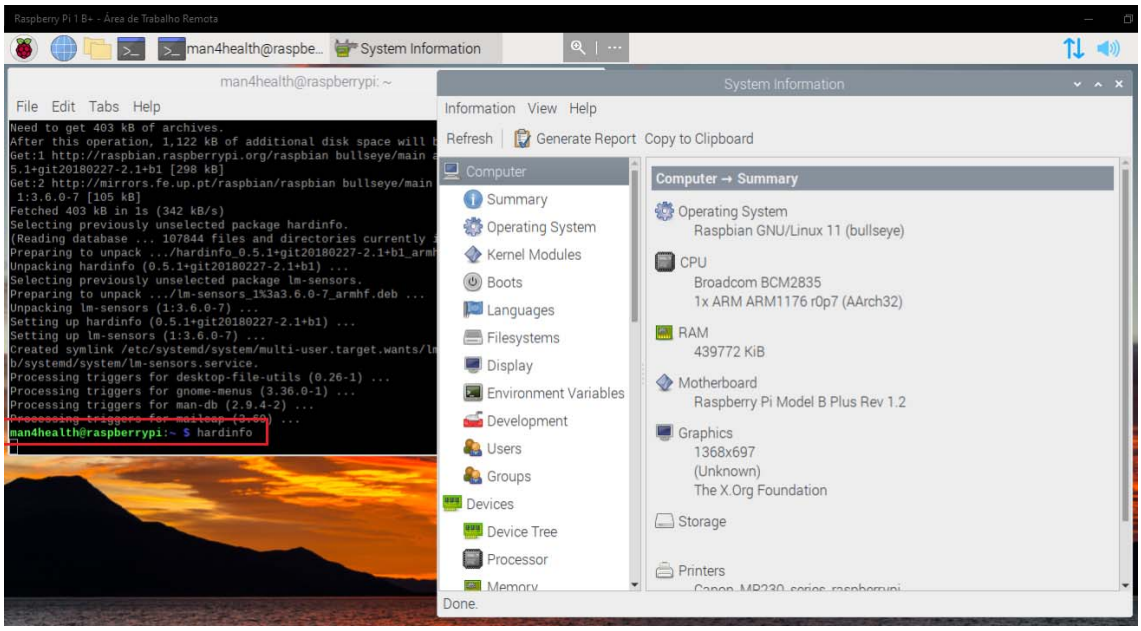



*Figura 4 - Resultado da execução de hardinfo.*

Antes de mais, e como não tinha informaçãoa cerca do chipset da placa de rede, liguei-a a um computador com Windows 10 instalado. O reconhecimento foi imediato e, acedendo ao Gestor de dispositivos, como se pode ver na Figura 5, obtive a informação de que se trata do **Realtek 8811CU**.

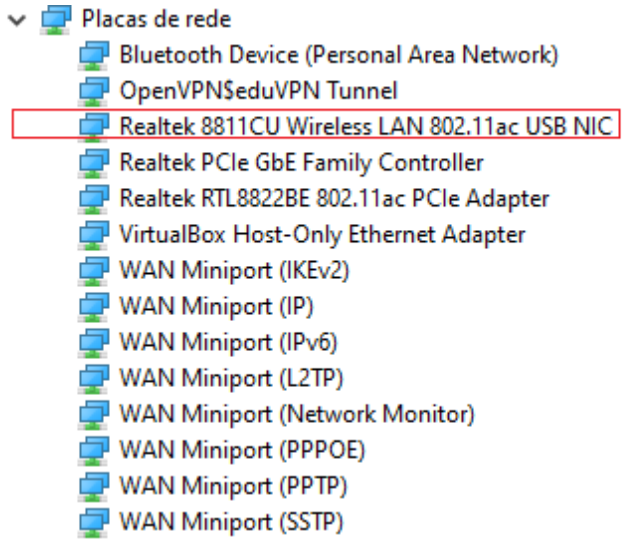



*Figura 5 - Informação fornecida pelo gestor de dispositivos do Windows 10 acerca da placa de rede USB.*

Depois de alguma pesquisa na WEB encontrei o seguinte repositório GIT com o processo de instalação:

https://github.com/morrownr/8821cu

Antes de ligar a placa de rede ao porto USB do Raspberry Pi, executar a seguinte sequencia de comandos:

```
$ sudo apt update && sudo apt upgrade
$ sudo apt install -y raspberrypi-kernel-headers build-essential bc dkms git
$ mkdir -p ~/src
$ cd ~/src
$ git clone https://github.com/morrownr/8821cu-20210118.git
$ cd ~/src/8821cu-20210118
$ ./ARM_RPI.sh
$ sudo ./install-driver.sh
$ sudo reboot
```

Depois do Raspberry Pi reinicializar, conetar a placa de rede à porta USB. Aparece a indicação para escolher o país e logo a seguir a placa aparece ativa e deteta as redes WiFi na vizinhança.

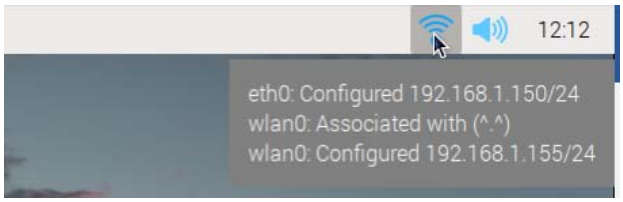
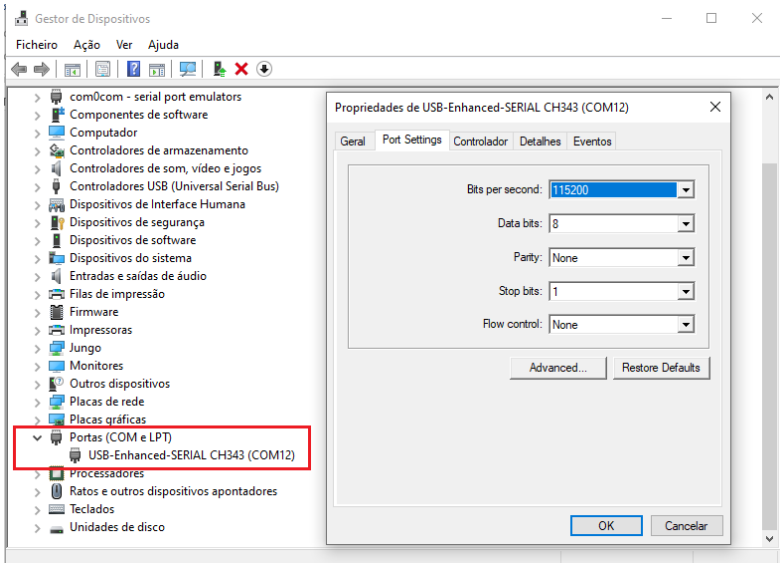
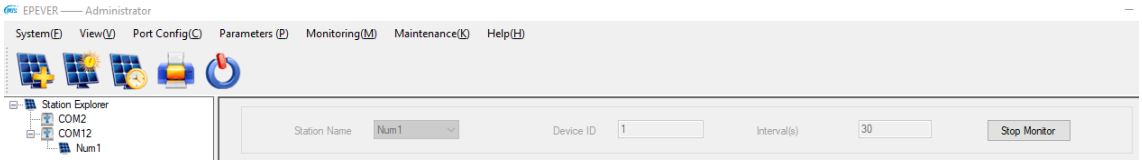


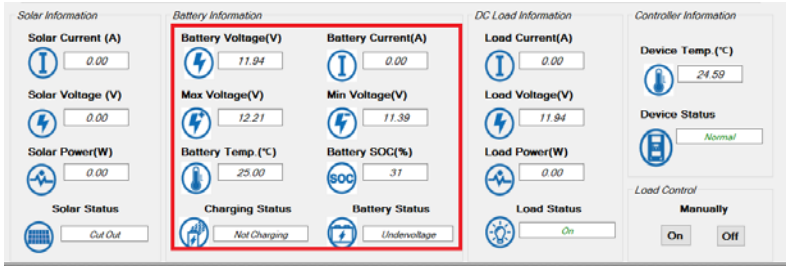

*Figura 6 - Resultado final da instalação dos drivers da placa de rede.*

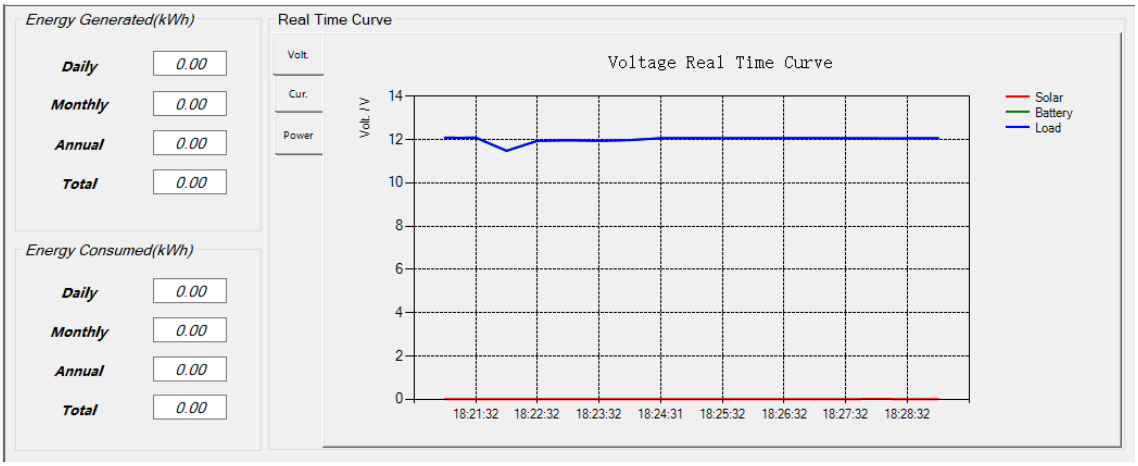
# 2. Comunicação com EPEVER via RS485

## 2.1. Usando o software para PC

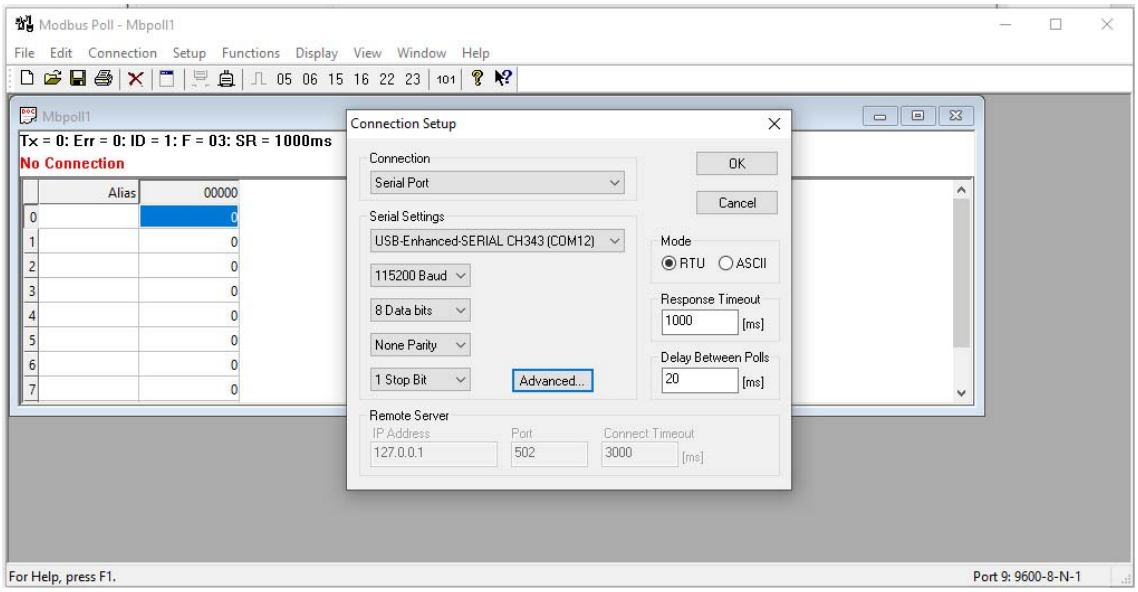


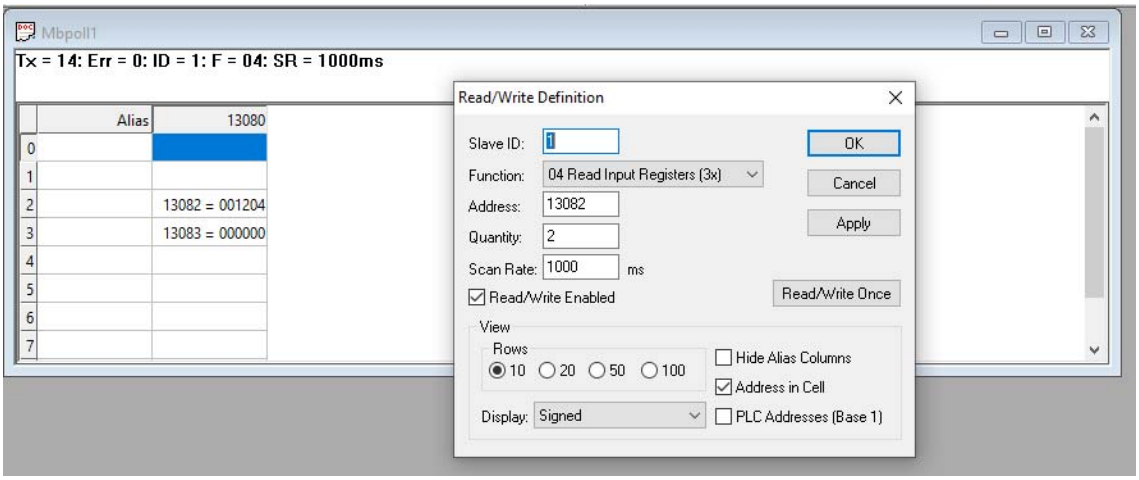



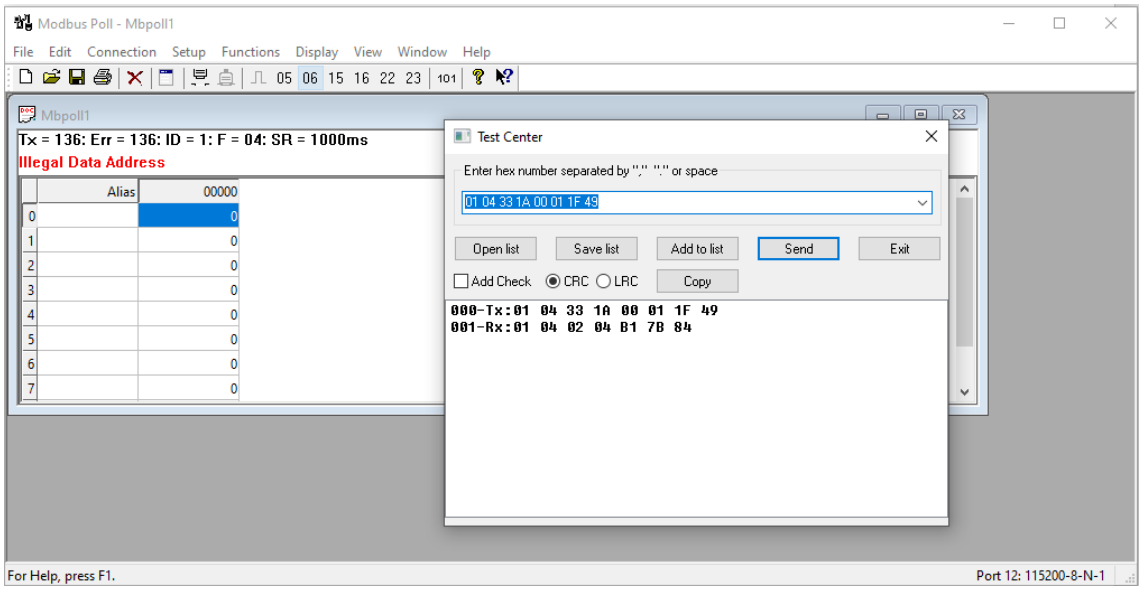



## 2.2. Usando Modbus Poll



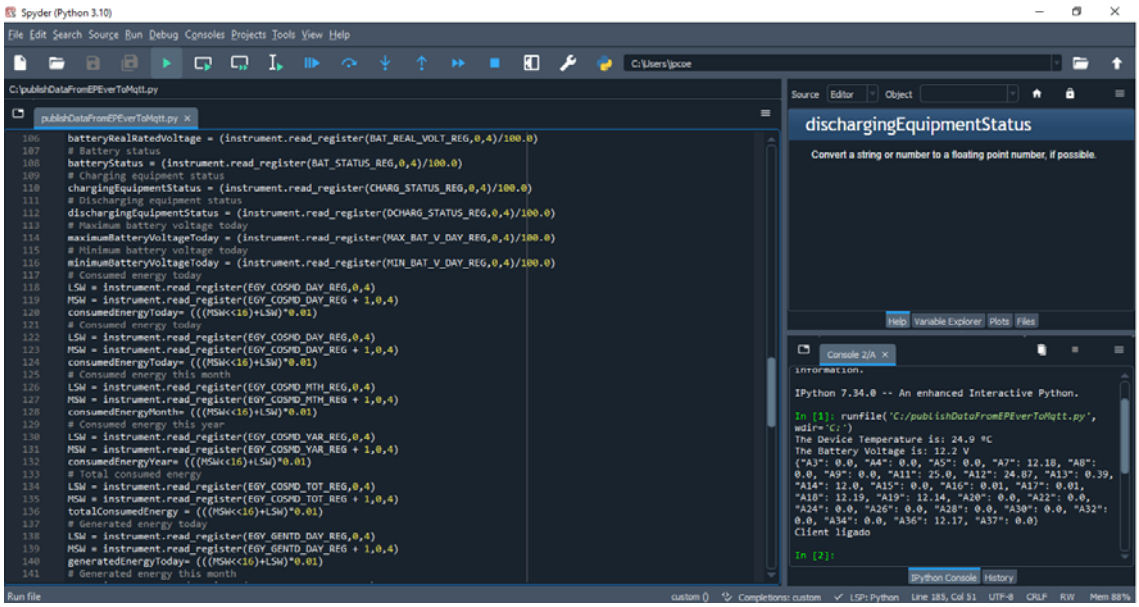

**O campo Address deve ser especificado em decimal e não em hexadecimal!!**

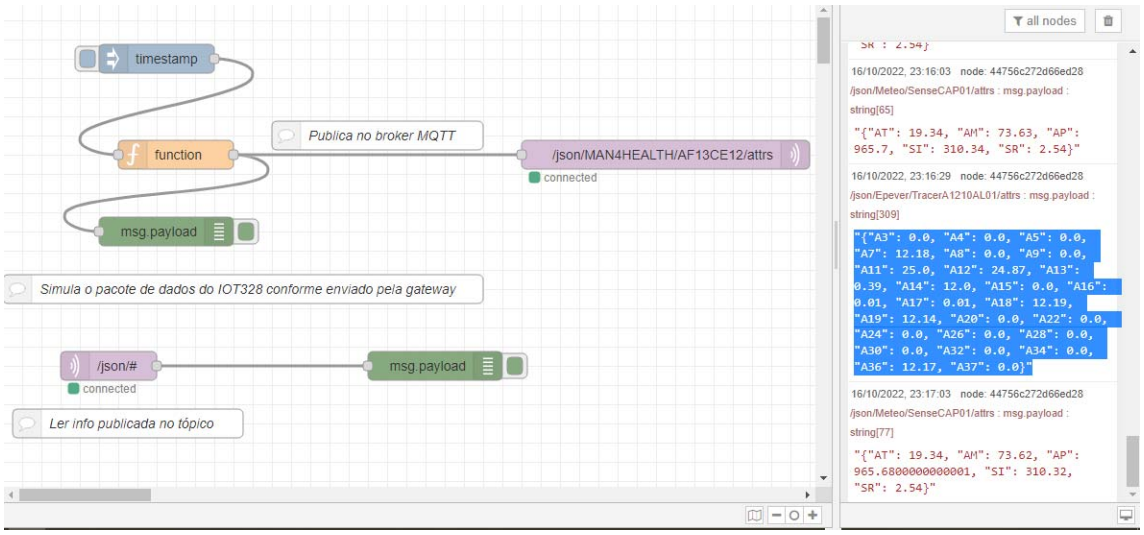



## 2.3. Protocolo de comunicação

- Utiliza o protocolo Modbus sobre RS485 (standard Modbus-RTU)
- Configuração mestre/escravo
- Comunicação iniciada pelo mestre suportando até 247 escravos
- Por defeito, o ID do controlador é “1” podendo ser alterado via software. Depois de alterado o ID, o controlador deve ser reiniciado.
- Os parâmetros de comunicação são:
    - Baud rate = 115200
    - Data bis = 8
    - Stop bits = 1
    - Sem controlo de fluxo de dados
    - Cada frame possui 32 bits (dois registos de 16 bits)

## 2.4. Comunicação usando Python (PC)




```
[{"id":"014d99ad544dca6d","type":"tab","label":"Flow
4","disabled":false,"info":""},{"id":"a57ccdd637a08551","type":"inject","z":"0
14d99ad544dca6d","name":"","props":[{"p":"payload"},{"p":"topic","vt":"str"}],
"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payloadType"
:"date","x":160,"y":60,"wires":[["1a3956c98e958232"]]},{"id":"1a3956c98e958232
","type":"function","z":"014d99ad544dca6d","name":"","func":"var
gateway_payload={};\nvar RSSI = Math.round(-10*(1+3*Math.random()));\nvar SNR
= Math.round(10*(1+1.5*Math.random()));\nvar bateria =
Math.round(10*Math.random())/10;\nvar humidadeSolo1 =
Math.round(100*(0.5+0.4*Math.random()));\nvar humidadeSolo2 =
```

```
Math.round(100*(0.5+0.4*Math.random()));\nvar humidadeSolo3 =
Math.round(100*(0.5+0.4*Math.random()));\nvar temperaturaSolo1 =
Math.round(30*(0.5+0.4*Math.random()));\nvar temperaturaSolo2 =
Math.round(30*(0.5+0.4*Math.random()));\nvar temperaturaSolo3 =
Math.round(30*(0.5+0.4*Math.random()));\nvar temperaturaAr1 =
Math.round(30*(0.5+0.4*Math.random()));\nvar humidadeAr1 =
Math.round(30*(0.5+0.4*Math.random()));\nvar radiacaoSolar1 =
Math.round(30*(0.5+0.4*Math.random()));\nvar profHumidadeSolo1 = -5;\nvar
profHumidadeSolo2 = -10;\nvar profHumidadeSolo3 = -20;\nvar
profTemperaturaSolo1 = -5;\nvar profTemperaturaSolo2 = -10;\nvar
profTemperaturaSolo3 = -20;\nvar alturaTemperaturaAr1 = 5;\nvar
alturaHumidadeAr1 = 5;\nvar alturaRadiacaoSolar1 = 5;\n\nmsg.payload={\n
\"RSSI\": RSSI,\n    \"SNR\": SNR,\n    \"B\": bateria,\n    \"HS\":[\n
{\"d\":humidadeSolo1,\"h\":profHumidadeSolo1},\n
{\"d\":humidadeSolo2,\"h\":profHumidadeSolo2},\n
{\"d\":humidadeSolo3,\"h\":profHumidadeSolo3}\n            ],\n    \"TS\":[\n
{\"d\":temperaturaSolo1,\"h\":profTemperaturaSolo1},\n
{\"d\":temperaturaSolo2,\"h\":profTemperaturaSolo2},\n
{\"d\":temperaturaSolo3,\"h\":profTemperaturaSolo3}\n            ],\n
\"HA\":[\n            {\"d\":humidadeAr1,\"h\":alturaHumidadeAr1}\n
],\n    \"TA\":[\n
{\"d\":temperaturaAr1,\"h\":alturaTemperaturaAr1}\n            ],\n
\"RS\":[\n            {\"d\":radiacaoSolar1,\"h\":alturaRadiacaoSolar1}\n
]\n};\n\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":180,"y
":160,"wires":[["dfcb7ab1470b223b","ba31fd3b6312c436"]]},{"id":"dfcb7ab1470b22
3b","type":"debug","z":"014d99ad544dca6d","name":"","active":true,"tosidebar":
true,"console":false,"tostatus":false,"complete":"payload","targetType":"msg",
"statusVal":"","statusType":"auto","x":170,"y":240,"wires":[]},{"id":"cce59153
ef78709f","type":"comment","z":"014d99ad544dca6d","name":"Simula o pacote de
dados do IOT328 conforme enviado pela gateway","info":"3 sensores de humidade
do solo a diferentes profundidades\n3 sensores de temperatura do solo\n1
sensor de temperatura do ar\n1 sensor de humidade do ar\n1 sensor de radiação
solar\n1 indicação de
bateria","x":250,"y":300,"wires":[]},{"id":"ba31fd3b6312c436","type":"mqtt
out","z":"014d99ad544dca6d","name":"","topic":"/json/MAN4HEALTH/AF13CE12/attrs
","qos":"","retain":"","respTopic":"","contentType":"","userProps":"","correl"
:"","expiry":"","broker":"b6aedbb0641104f5","x":690,"y":160,"wires":[]},{"id":
"905d3b59489ead86","type":"mqtt
in","z":"014d99ad544dca6d","name":"","topic":"/json/#","qos":"2","datatype":"a
uto","broker":"b6aedbb0641104f5","nl":false,"rap":true,"rh":0,"x":110,"y":380,
"wires":[["44756c272d66ed28"]]},{"id":"44756c272d66ed28","type":"debug","z":"0
14d99ad544dca6d","name":"","active":true,"tosidebar":true,"console":false,"tos
tatus":false,"complete":"false","statusVal":"","statusType":"auto","x":450,"y"
:380,"wires":[]},{"id":"36778ad7ecf91c38","type":"comment","z":"014d99ad544dca
6d","name":"Ler info publicada no
```

```
tópico","info":"","x":130,"y":440,"wires":[]},{"id":"5b862d2ac6ce699b","type":
"comment","z":"014d99ad544dca6d","name":"Publica no broker
MQTT","info":"","x":390,"y":140,"wires":[]},{"id":"b6aedbb0641104f5","type":"m
qtt-
broker","name":"","broker":"localhost","port":"1883","clientid":"","usetls":fa
lse,"protocolVersion":"4","keepalive":"60","cleansession":true,"birthTopic":""
,"birthQos":"0","birthPayload":"","birthMsg":{},"closeTopic":"","closeQos":"0"
,"closePayload":"","closeMsg":{},"willTopic":"","willQos":"0","willPayload":""
,"willMsg":{},"sessionExpiry":""}]
```


### 2.4.1. Script Python

É necessário instalar as packages: minimalmodbus e paho

```
# -*- coding: utf-8 -*-
"""
16/outubro/2022
@author: jpcoelho
"""

import minimalmodbus
import paho.mqtt.client as mqttclient
import time
import json

def on_connect(client, usedata,flags,rc):
    if rc==0:
        print("Client ligado")
        global connected
        connected=True
    else:
        print("Falha de ligação do cliente")

# Configuração MQTT
connected = False
broker_address = "localhost"
port = 1883
# user="man4health"
# password="#Man4Health"
client = mqttclient.Client("MQTT")
#client.username_pw_set(user,password=password)
client.on_connect = on_connect

## Tópico MQTT
apikey = "Epever"
deviceid = "TracerA1210AL01"
protocol = "json"
topic = "/" + protocol +"/" + apikey + "/" + deviceid + "/attrs"

# Configuração Modbus
PORT='COM12'

# Endereços
PV_IN_VOLTAGE_REG = 0x3100
PV_IN_CURRENT_REG = 0x3101
PV_IN_POWER_REG   = 0x3102 # L + H
LOAD_VOLTAGE_REG  = 0x310C
LOAD_CURRENT_REG  = 0x310D
LOAD_POWER_REG    = 0x310E # L + H
BAT_TEMP_REG      = 0x3110
DEV_TEMP_REG      = 0x3111
BAT_CAPACITY_REG  = 0x311A
```

```
BAT_REAL_VOLT_REG = 0x311D
BAT_STATUS_REG    = 0x3200
CHARG_STATUS_REG  = 0x3201
DCHARG_STATUS_REG = 0x3202
MAX_BAT_V_DAY_REG = 0x3302
MIN_BAT_V_DAY_REG = 0x3303
EGY_COSMD_DAY_REG = 0x3304 # L + H
EGY_COSMD_MTH_REG = 0x3306 # L + H
EGY_COSMD_YAR_REG = 0x3308 # L + H
EGY_COSMD_TOT_REG = 0x330A # L + H
EGY_GENTD_DAY_REG = 0x330C # L + H
EGY_GENTD_MTH_REG = 0x330E # L + H
EGY_GENTD_YAR_REG = 0x3310 # L + H
EGY_GENTD_TOT_REG = 0x3312 # L + H
BAT_MEAS_VOLT_REG = 0x331A
BAT_MEAS_CURT_REG = 0x331B # L + H

SLAVE_ADDRESS = 1

# Configura instrumento
instrument =
minimalmodbus.Instrument(PORT,SLAVE_ADDRESS,mode=minimalmodbus.MODE_RTU)

# Parametros do instrumento
instrument.serial.baudrate = 115200        # Baud
instrument.serial.bytesize = 8
instrument.serial.parity   = minimalmodbus.serial.PARITY_NONE
instrument.serial.stopbits = 1
instrument.serial.timeout  = 0.5          # seconds
# print(instrument)

# Fecha porto
instrument.close_port_after_each_call = True
instrument.clear_buffers_before_each_transaction = True

# PV array input voltage
PVarrayInputVoltage = (instrument.read_register(PV_IN_VOLTAGE_REG,0,4)/100.0)
# PV array input current
PVarrayInputCurrent = (instrument.read_register(PV_IN_CURRENT_REG,0,4)/100.0)
# PV array input power
LSW = instrument.read_register(PV_IN_POWER_REG,0,4)
MSW = instrument.read_register(PV_IN_POWER_REG + 1,0,4)
PVarrayInputPower= (((MSW<<16)+LSW)*0.01)
# Load voltage
loadVoltage = (instrument.read_register(LOAD_VOLTAGE_REG,0,4)/100.0)
# Load current
loadCurrent = (instrument.read_register(LOAD_CURRENT_REG,0,4)/100.0)
# Load power
LSW = instrument.read_register(LOAD_POWER_REG,0,4)
MSW = instrument.read_register(LOAD_POWER_REG + 1,0,4)
loadPower= (((MSW<<16)+LSW)*0.01)
# Battery temperature
batteryTemperature = (instrument.read_register(BAT_TEMP_REG,0,4)/100.0)
# Device temperature
deviceTemperature = (instrument.read_register(DEV_TEMP_REG,0,4)/100.0)
# Battery SOC
batterySOC = (instrument.read_register(BAT_CAPACITY_REG,0,4)/100.0)
# Battery's real rated voltage
batteryRealRatedVoltage =
(instrument.read_register(BAT_REAL_VOLT_REG,0,4)/100.0)
# Battery status
batteryStatus = (instrument.read_register(BAT_STATUS_REG,0,4)/100.0)
# Charging equipment status
chargingEquipmentStatus =
(instrument.read_register(CHARG_STATUS_REG,0,4)/100.0)
# Discharging equipment status
dischargingEquipmentStatus =
(instrument.read_register(DCHARG_STATUS_REG,0,4)/100.0)
```

```
# Maximum battery voltage today
maximumBatteryVoltageToday =
(instrument.read_register(MAX_BAT_V_DAY_REG,0,4)/100.0)
# Minimum battery voltage today
minimumBatteryVoltageToday =
(instrument.read_register(MIN_BAT_V_DAY_REG,0,4)/100.0)
# Consumed energy today
LSW = instrument.read_register(EGY_COSMD_DAY_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_DAY_REG + 1,0,4)
consumedEnergyToday= (((MSW<<16)+LSW)*0.01)
# Consumed energy today
LSW = instrument.read_register(EGY_COSMD_DAY_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_DAY_REG + 1,0,4)
consumedEnergyToday= (((MSW<<16)+LSW)*0.01)
# Consumed energy this month
LSW = instrument.read_register(EGY_COSMD_MTH_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_MTH_REG + 1,0,4)
consumedEnergyMonth= (((MSW<<16)+LSW)*0.01)
# Consumed energy this year
LSW = instrument.read_register(EGY_COSMD_YAR_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_YAR_REG + 1,0,4)
consumedEnergyYear= (((MSW<<16)+LSW)*0.01)
# Total consumed energy
LSW = instrument.read_register(EGY_COSMD_TOT_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_TOT_REG + 1,0,4)
totalConsumedEnergy = (((MSW<<16)+LSW)*0.01)
# Generated energy today
LSW = instrument.read_register(EGY_GENTD_DAY_REG,0,4)
MSW = instrument.read_register(EGY_GENTD_DAY_REG + 1,0,4)
generatedEnergyToday= (((MSW<<16)+LSW)*0.01)
# Generated energy this month
LSW = instrument.read_register(EGY_GENTD_MTH_REG,0,4)
MSW = instrument.read_register(EGY_GENTD_MTH_REG + 1,0,4)
generatedEnergyMonth= (((MSW<<16)+LSW)*0.01)
# Generated energy this year
LSW = instrument.read_register(EGY_GENTD_YAR_REG,0,4)
MSW = instrument.read_register(EGY_GENTD_YAR_REG + 1,0,4)
generatedEnergyYear= (((MSW<<16)+LSW)*0.01)
# Total generated energy
LSW = instrument.read_register(EGY_GENTD_TOT_REG,0,4)
MSW = instrument.read_register(EGY_GENTD_TOT_REG + 1,0,4)
totalGeneratedEnergy= (((MSW<<16)+LSW)*0.01)
# Battery voltage
batteryVoltage = (instrument.read_register(BAT_MEAS_VOLT_REG,0,4)/100.0)
# Battery current
LSW = instrument.read_register(BAT_MEAS_CURT_REG,0,4)
MSW = instrument.read_register(BAT_MEAS_CURT_REG + 1,0,4)
batteryCurrent= (((MSW<<16)+LSW)*0.01)

# Mostra valores (debug apenas)
print('The Device Temperature is: %.1f ºC\r' % deviceTemperature)
print('The Battery Voltage is: %.1f V\r' % batteryVoltage)

# Criar payload json

payload = { "A3": PVarrayInputVoltage,
            "A4": PVarrayInputCurrent,
            "A5": PVarrayInputPower,
            "A7": loadVoltage,
            "A8": loadCurrent,
            "A9": loadPower,
            "A11": batteryTemperature,
            "A12": deviceTemperature,
            "A13": batterySOC,
            "A14": batteryRealRatedVoltage,
            "A15": batteryStatus,
            "A16": chargingEquipmentStatus,
            "A17": dischargingEquipmentStatus,
```

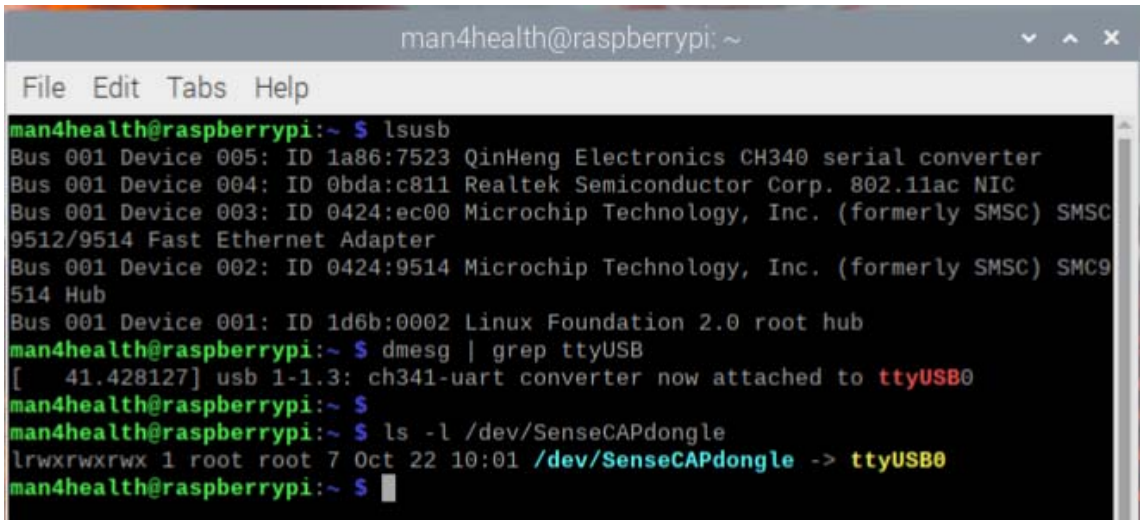
```
            "A18": maximumBatteryVoltageToday,
            "A19": minimumBatteryVoltageToday,
            "A20": consumedEnergyToday,
            "A22": consumedEnergyMonth,
            "A24": consumedEnergyYear,
            "A26": totalConsumedEnergy,
            "A28": generatedEnergyToday,
            "A30": generatedEnergyMonth,
            "A32": generatedEnergyYear,
            "A34": totalGeneratedEnergy,
            "A36": batteryVoltage,
            "A37": batteryCurrent
           }

message = json.dumps(payload)

print(message)

# Publicar dados no broker
try:
    client.connect(broker_address,port=port)

    client.loop_start()

    while connected != True:
        time.sleep(0.2)

    client.publish(topic,message)
    client.loop_stop()
except:
    print("Impossível conetar ao broker MQTT ")
```

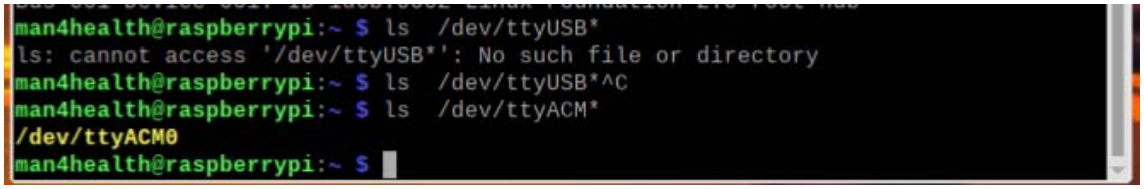
## 2.5. Instalar RS485 dongle no Raspberry Pi

```
udevadm info --name=/dev/ttyUSB0 --attribute-walk
```

```
Bus 001 Device 008: ID 1a86:55d3 QinHeng Electronics USB Single Serial
Bus 001 Device 007: ID 1a86:7523 QinHeng Electronics CH340 serial converter
Bus 001 Device 004: ID 0bda:c811 Realtek Semiconductor Corp. 802.11ac NIC
Bus 001 Device 003: ID 0424:ec00 Microchip Technology, Inc. (formerly SMSC) SMSC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Microchip Technology, Inc. (formerly SMSC) SMC9514 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
man4health@raspberrypi:~ $
```

```
ACTION=="add", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="7523",
SYMLINK+="SenseCAPdongle"
ACTION=="add", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="55d3",
SYMLINK+="EPEVERdongle"
```




https://rfc1149.net/blog/2013/03/05/what-is-the-difference-between-devttyusbx-and-devttyacmx/

Existem duas maneiras com os dispositivos série USB enumeram. Podem ser enumerados como /dev/ttyUSBx ou como /dev/ttyACMx. No caso (ttyUSB), o dispositivo apresenta-se como uma porta série simples. No caso do ttyACM, o dispositivo apresenta-se como um modem USB (mesmo que isso não seja verdade). Neste último caso, o dispositivo aceita comandos AT.

## 2.6. Script Python portado para RPi: ver 1.0

```
# -*- coding: utf-8 -*-
"""
22/outubro/2022
@author: jpcoelho
"""

import minimalmodbus
import paho.mqtt.client as mqttclient
import time
import json

def on_connect(client, usedata,flags,rc):
    if rc==0:
        print("Client ligado")
        global connected
        connected=True
```

```
    else:
        print("Falha de ligação do cliente")

# Configuração MQTT
connected = False
# broker_address = "192.168.1.68"
broker_address = "193.136.195.25" # Servidor IPB
# broker_address = "mrmaldoror.hopto.org"
# broker_address = "localhost"
port = 1883
# user="man4health"
# password="#Man4Health"
client = mqttclient.Client("MQTT")
#client.username_pw_set(user,password=password)
client.on_connect = on_connect

## Tópico MQTT
apikey = "Epever"
deviceid = "TracerA1210AL01"
protocol = "json"
topic = "/" + protocol +"/" + apikey + "/" + deviceid + "/attrs"

# Configuração Modbus
PORT='/dev/ttyACM0'

# Endereços
PV_IN_VOLTAGE_REG = 0x3100
PV_IN_CURRENT_REG = 0x3101
PV_IN_POWER_REG   = 0x3102 # L + H
LOAD_VOLTAGE_REG  = 0x310C
LOAD_CURRENT_REG  = 0x310D
LOAD_POWER_REG    = 0x310E # L + H
BAT_TEMP_REG      = 0x3110
DEV_TEMP_REG      = 0x3111
BAT_CAPACITY_REG  = 0x311A
BAT_REAL_VOLT_REG = 0x311D
BAT_STATUS_REG    = 0x3200
CHARG_STATUS_REG  = 0x3201
DCHARG_STATUS_REG = 0x3202
MAX_BAT_V_DAY_REG = 0x3302
MIN_BAT_V_DAY_REG = 0x3303
EGY_COSMD_DAY_REG = 0x3304 # L + H
EGY_COSMD_MTH_REG = 0x3306 # L + H
EGY_COSMD_YAR_REG = 0x3308 # L + H
EGY_COSMD_TOT_REG = 0x330A # L + H
EGY_GENTD_DAY_REG = 0x330C # L + H
EGY_GENTD_MTH_REG = 0x330E # L + H
EGY_GENTD_YAR_REG = 0x3310 # L + H
EGY_GENTD_TOT_REG = 0x3312 # L + H
BAT_MEAS_VOLT_REG = 0x331A
BAT_MEAS_CURT_REG = 0x331B # L + H

SLAVE_ADDRESS = 1

# Configura instrumento
instrument =
minimalmodbus.Instrument(PORT,SLAVE_ADDRESS,mode=minimalmodbus.MODE_RTU)

# Parametros do instrumento
instrument.serial.baudrate = 115200        # Baud
instrument.serial.bytesize = 8
instrument.serial.parity   = minimalmodbus.serial.PARITY_NONE
instrument.serial.stopbits = 1
instrument.serial.timeout  = 0.5          # seconds
# print(instrument)

# Fecha porto
instrument.close_port_after_each_call = True
```

```
instrument.clear_buffers_before_each_transaction = True

# PV array input voltage
PVarrayInputVoltage = (instrument.read_register(PV_IN_VOLTAGE_REG,0,4)/100.0)
# PV array input current
PVarrayInputCurrent = (instrument.read_register(PV_IN_CURRENT_REG,0,4)/100.0)
# PV array input power
LSW = instrument.read_register(PV_IN_POWER_REG,0,4)
MSW = instrument.read_register(PV_IN_POWER_REG + 1,0,4)
PVarrayInputPower= (((MSW<<16)+LSW)*0.01)
# Load voltage
loadVoltage = (instrument.read_register(LOAD_VOLTAGE_REG,0,4)/100.0)
# Load current
loadCurrent = (instrument.read_register(LOAD_CURRENT_REG,0,4)/100.0)
# Load power
LSW = instrument.read_register(LOAD_POWER_REG,0,4)
MSW = instrument.read_register(LOAD_POWER_REG + 1,0,4)
loadPower= (((MSW<<16)+LSW)*0.01)
# Battery temperature
batteryTemperature = (instrument.read_register(BAT_TEMP_REG,0,4)/100.0)
# Device temperature
deviceTemperature = (instrument.read_register(DEV_TEMP_REG,0,4)/100.0)
# Battery SOC
batterySOC = (instrument.read_register(BAT_CAPACITY_REG,0,4)/100.0)
# Battery's real rated voltage
batteryRealRatedVoltage =
(instrument.read_register(BAT_REAL_VOLT_REG,0,4)/100.0)
# Battery status
batteryStatus = (instrument.read_register(BAT_STATUS_REG,0,4)/100.0)
# Charging equipment status
chargingEquipmentStatus =
(instrument.read_register(CHARG_STATUS_REG,0,4)/100.0)
# Discharging equipment status
dischargingEquipmentStatus =
(instrument.read_register(DCHARG_STATUS_REG,0,4)/100.0)
# Maximum battery voltage today
maximumBatteryVoltageToday =
(instrument.read_register(MAX_BAT_V_DAY_REG,0,4)/100.0)
# Minimum battery voltage today
minimumBatteryVoltageToday =
(instrument.read_register(MIN_BAT_V_DAY_REG,0,4)/100.0)
# Consumed energy today
LSW = instrument.read_register(EGY_COSMD_DAY_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_DAY_REG + 1,0,4)
consumedEnergyToday= (((MSW<<16)+LSW)*0.01)
# Consumed energy today
LSW = instrument.read_register(EGY_COSMD_DAY_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_DAY_REG + 1,0,4)
consumedEnergyToday= (((MSW<<16)+LSW)*0.01)
# Consumed energy this month
LSW = instrument.read_register(EGY_COSMD_MTH_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_MTH_REG + 1,0,4)
consumedEnergyMonth= (((MSW<<16)+LSW)*0.01)
# Consumed energy this year
LSW = instrument.read_register(EGY_COSMD_YAR_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_YAR_REG + 1,0,4)
consumedEnergyYear= (((MSW<<16)+LSW)*0.01)
# Total consumed energy
LSW = instrument.read_register(EGY_COSMD_TOT_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_TOT_REG + 1,0,4)
totalConsumedEnergy = (((MSW<<16)+LSW)*0.01)
# Generated energy today
LSW = instrument.read_register(EGY_GENTD_DAY_REG,0,4)
MSW = instrument.read_register(EGY_GENTD_DAY_REG + 1,0,4)
generatedEnergyToday= (((MSW<<16)+LSW)*0.01)
# Generated energy this month
LSW = instrument.read_register(EGY_GENTD_MTH_REG,0,4)
MSW = instrument.read_register(EGY_GENTD_MTH_REG + 1,0,4)
```

```
generatedEnergyMonth= (((MSW<<16)+LSW)*0.01)
# Generated energy this year
LSW = instrument.read_register(EGY_GENTD_YAR_REG,0,4)
MSW = instrument.read_register(EGY_GENTD_YAR_REG + 1,0,4)
generatedEnergyYear= (((MSW<<16)+LSW)*0.01)
# Total generated energy
LSW = instrument.read_register(EGY_GENTD_TOT_REG,0,4)
MSW = instrument.read_register(EGY_GENTD_TOT_REG + 1,0,4)
totalGeneratedEnergy= (((MSW<<16)+LSW)*0.01)
# Battery voltage
batteryVoltage = (instrument.read_register(BAT_MEAS_VOLT_REG,0,4)/100.0)
# Battery current
LSW = instrument.read_register(BAT_MEAS_CURT_REG,0,4)
MSW = instrument.read_register(BAT_MEAS_CURT_REG + 1,0,4)
batteryCurrent= (((MSW<<16)+LSW)*0.01)

# Mostra valores (debug apenas)
print('The Device Temperature is: %.1f ºC\r' % deviceTemperature)
print('The Battery Voltage is: %.1f V\r' % batteryVoltage)

# Criar payload json

payload = { "A3": PVarrayInputVoltage,
            "A4": PVarrayInputCurrent,
            "A5": PVarrayInputPower,
            "A7": loadVoltage,
            "A8": loadCurrent,
            "A9": loadPower,
            "A11": batteryTemperature,
            "A12": deviceTemperature,
            "A13": batterySOC,
            "A14": batteryRealRatedVoltage,
            "A15": batteryStatus,
            "A16": chargingEquipmentStatus,
            "A17": dischargingEquipmentStatus,
            "A18": maximumBatteryVoltageToday,
            "A19": minimumBatteryVoltageToday,
            "A20": consumedEnergyToday,
            "A22": consumedEnergyMonth,
            "A24": consumedEnergyYear,
            "A26": totalConsumedEnergy,
            "A28": generatedEnergyToday,
            "A30": generatedEnergyMonth,
            "A32": generatedEnergyYear,
            "A34": totalGeneratedEnergy,
            "A36": batteryVoltage,
            "A37": batteryCurrent
           }

message = json.dumps(payload)

print(message)

# Publicar dados no broker
try:
    client.connect(broker_address,port=port)

    client.loop_start()

    while connected != True:
        time.sleep(0.2)

    client.publish(topic,message)
    client.loop_stop()
except:
    print("Impossível conetar ao broker MQTT ")
```

## 2.7. Script Python portado para RPi: ver 1.01

```
# -*- coding: utf-8 -*-
"""
16/outubro/2022
@author: jpcoelho
"""

import minimalmodbus
import paho.mqtt.client as mqttclient
import time
import json

def on_connect(client, usedata,flags,rc):
    if rc==0:
        print("Client ligado")
        global connected
        connected=True
    else:
        print("Falha de ligação do cliente")

# Configuração MQTT
connected = False
# broker_address = "192.168.1.68"
broker_address = "193.136.195.25" # Servidor IPB
# broker_address = "mrmaldoror.hopto.org"
# broker_address = "localhost"
port = 1883
# user="man4health"
# password="#Man4Health"
client = mqttclient.Client("MQTT")
#client.username_pw_set(user,password=password)
client.on_connect = on_connect

## Tópico MQTT
apikey = "Epever"
deviceid = "TracerA1210AL01"
protocol = "json"
topic = "/" + protocol +"/" + apikey + "/" + deviceid + "/attrs"

# Configuração Modbus
PORT='/dev/EPEVERdongle'

# Endereços
PV_IN_VOLTAGE_REG = 0x3100
PV_IN_CURRENT_REG = 0x3101
PV_IN_POWER_REG   = 0x3102 # L + H
LOAD_VOLTAGE_REG  = 0x310C
LOAD_CURRENT_REG  = 0x310D
LOAD_POWER_REG    = 0x310E # L + H
BAT_TEMP_REG      = 0x3110
DEV_TEMP_REG      = 0x3111
BAT_CAPACITY_REG  = 0x311A
BAT_REAL_VOLT_REG = 0x311D
BAT_STATUS_REG    = 0x3200
CHARG_STATUS_REG  = 0x3201
DCHARG_STATUS_REG = 0x3202
MAX_BAT_V_DAY_REG = 0x3302
MIN_BAT_V_DAY_REG = 0x3303
EGY_COSMD_DAY_REG = 0x3304 # L + H
EGY_COSMD_MTH_REG = 0x3306 # L + H
EGY_COSMD_YAR_REG = 0x3308 # L + H
EGY_COSMD_TOT_REG = 0x330A # L + H
EGY_GENTD_DAY_REG = 0x330C # L + H
EGY_GENTD_MTH_REG = 0x330E # L + H
EGY_GENTD_YAR_REG = 0x3310 # L + H
EGY_GENTD_TOT_REG = 0x3312 # L + H
```

```
BAT_MEAS_VOLT_REG = 0x331A
BAT_MEAS_CURT_REG = 0x331B # L + H

SLAVE_ADDRESS = 1

# Configura instrumento
instrument =
minimalmodbus.Instrument(PORT,SLAVE_ADDRESS,mode=minimalmodbus.MODE_RTU)

# Parametros do instrumento
instrument.serial.baudrate = 115200        # Baud
instrument.serial.bytesize = 8
instrument.serial.parity   = minimalmodbus.serial.PARITY_NONE
instrument.serial.stopbits = 1
instrument.serial.timeout  = 0.5          # seconds
# print(instrument)

# Fecha porto
instrument.close_port_after_each_call = True
instrument.clear_buffers_before_each_transaction = True

# PV array input voltage
PVarrayInputVoltage = (instrument.read_register(PV_IN_VOLTAGE_REG,0,4)/100.0)
# PV array input current
PVarrayInputCurrent = (instrument.read_register(PV_IN_CURRENT_REG,0,4)/100.0)
# PV array input power
LSW = instrument.read_register(PV_IN_POWER_REG,0,4)
MSW = instrument.read_register(PV_IN_POWER_REG + 1,0,4)
PVarrayInputPower= (((MSW<<16)+LSW)*0.01)
# Load voltage
loadVoltage = (instrument.read_register(LOAD_VOLTAGE_REG,0,4)/100.0)
# Load current
loadCurrent = (instrument.read_register(LOAD_CURRENT_REG,0,4)/100.0)
# Load power
LSW = instrument.read_register(LOAD_POWER_REG,0,4)
MSW = instrument.read_register(LOAD_POWER_REG + 1,0,4)
loadPower= (((MSW<<16)+LSW)*0.01)
# Battery temperature
batteryTemperature = (instrument.read_register(BAT_TEMP_REG,0,4)/100.0)
# Device temperature
deviceTemperature = (instrument.read_register(DEV_TEMP_REG,0,4)/100.0)
# Battery SOC
batterySOC = (instrument.read_register(BAT_CAPACITY_REG,0,4)/100.0)
# Battery's real rated voltage
batteryRealRatedVoltage =
(instrument.read_register(BAT_REAL_VOLT_REG,0,4)/100.0)
# Battery status
batteryStatus = (instrument.read_register(BAT_STATUS_REG,0,4)/100.0)
# Charging equipment status
chargingEquipmentStatus =
(instrument.read_register(CHARG_STATUS_REG,0,4)/100.0)
# Discharging equipment status
dischargingEquipmentStatus =
(instrument.read_register(DCHARG_STATUS_REG,0,4)/100.0)
# Maximum battery voltage today
maximumBatteryVoltageToday =
(instrument.read_register(MAX_BAT_V_DAY_REG,0,4)/100.0)
# Minimum battery voltage today
minimumBatteryVoltageToday =
(instrument.read_register(MIN_BAT_V_DAY_REG,0,4)/100.0)
# Consumed energy today
LSW = instrument.read_register(EGY_COSMD_DAY_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_DAY_REG + 1,0,4)
consumedEnergyToday= (((MSW<<16)+LSW)*0.01)
# Consumed energy today
LSW = instrument.read_register(EGY_COSMD_DAY_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_DAY_REG + 1,0,4)
consumedEnergyToday= (((MSW<<16)+LSW)*0.01)
```

```
# Consumed energy this month
LSW = instrument.read_register(EGY_COSMD_MTH_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_MTH_REG + 1,0,4)
consumedEnergyMonth= (((MSW<<16)+LSW)*0.01)
# Consumed energy this year
LSW = instrument.read_register(EGY_COSMD_YAR_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_YAR_REG + 1,0,4)
consumedEnergyYear= (((MSW<<16)+LSW)*0.01)
# Total consumed energy
LSW = instrument.read_register(EGY_COSMD_TOT_REG,0,4)
MSW = instrument.read_register(EGY_COSMD_TOT_REG + 1,0,4)
totalConsumedEnergy = (((MSW<<16)+LSW)*0.01)
# Generated energy today
LSW = instrument.read_register(EGY_GENTD_DAY_REG,0,4)
MSW = instrument.read_register(EGY_GENTD_DAY_REG + 1,0,4)
generatedEnergyToday= (((MSW<<16)+LSW)*0.01)
# Generated energy this month
LSW = instrument.read_register(EGY_GENTD_MTH_REG,0,4)
MSW = instrument.read_register(EGY_GENTD_MTH_REG + 1,0,4)
generatedEnergyMonth= (((MSW<<16)+LSW)*0.01)
# Generated energy this year
LSW = instrument.read_register(EGY_GENTD_YAR_REG,0,4)
MSW = instrument.read_register(EGY_GENTD_YAR_REG + 1,0,4)
generatedEnergyYear= (((MSW<<16)+LSW)*0.01)
# Total generated energy
LSW = instrument.read_register(EGY_GENTD_TOT_REG,0,4)
MSW = instrument.read_register(EGY_GENTD_TOT_REG + 1,0,4)
totalGeneratedEnergy= (((MSW<<16)+LSW)*0.01)
# Battery voltage
batteryVoltage = (instrument.read_register(BAT_MEAS_VOLT_REG,0,4)/100.0)
# Battery current
LSW = instrument.read_register(BAT_MEAS_CURT_REG,0,4)
MSW = instrument.read_register(BAT_MEAS_CURT_REG + 1,0,4)
batteryCurrent= (((MSW<<16)+LSW)*0.01)

# Mostra valores (debug apenas)
print('The Device Temperature is: %.1f ºC\r' % deviceTemperature)
print('The Battery Voltage is: %.1f V\r' % batteryVoltage)

# Criar payload json

payload = { "A3": PVarrayInputVoltage,
            "A4": PVarrayInputCurrent,
            "A5": PVarrayInputPower,
            "A7": loadVoltage,
            "A8": loadCurrent,
            "A9": loadPower,
            "A11": batteryTemperature,
            "A12": deviceTemperature,
            "A13": batterySOC,
            "A14": batteryRealRatedVoltage,
            "A15": batteryStatus,
            "A16": chargingEquipmentStatus,
            "A17": dischargingEquipmentStatus,
            "A18": maximumBatteryVoltageToday,
            "A19": minimumBatteryVoltageToday,
            "A20": consumedEnergyToday,
            "A22": consumedEnergyMonth,
            "A24": consumedEnergyYear,
            "A26": totalConsumedEnergy,
            "A28": generatedEnergyToday,
            "A30": generatedEnergyMonth,
            "A32": generatedEnergyYear,
            "A34": totalGeneratedEnergy,
            "A36": batteryVoltage,
            "A37": batteryCurrent
          }
```

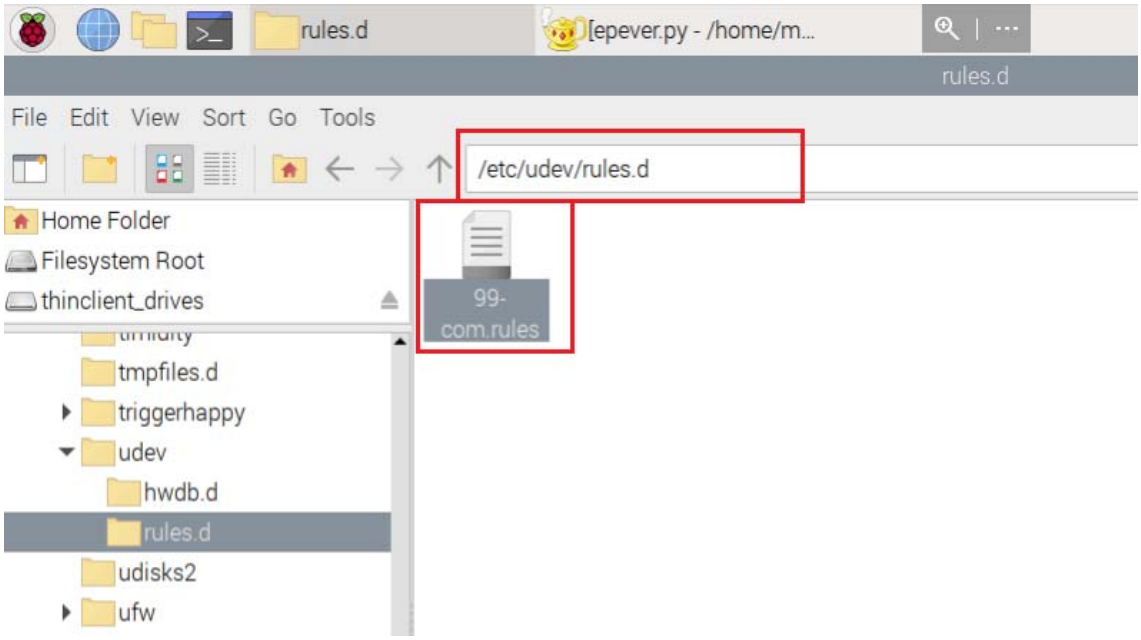
```
message = json.dumps(payload)

print(message)

# Publicar dados no broker
try:
    client.connect(broker_address,port=port)

    client.loop_start()

    while connected != True:
        time.sleep(0.2)

    client.publish(topic,message)
    client.loop_stop()
except:
    print("Impossível conetar ao broker MQTT ")
```

## 2.8. Forçar a que os portos COM tenha sempre a mesma designação

How to bind USB device under a static name

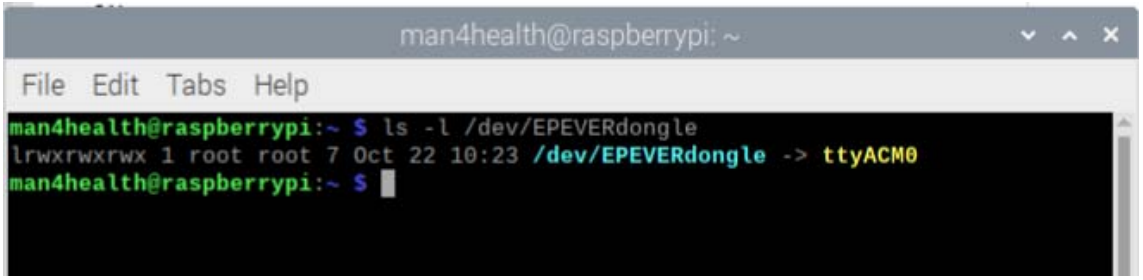
Verificar as variáveis associadas ao dispositivo executando no terminal:

```
$ sudo udevadm info -a -p $(udevadm info -q path -n /de/ttyUSB0)
```



```
SUBSYSTEM=="input", GROUP="input", MODE="0660"
SUBSYSTEM=="i2c-dev", GROUP="i2c", MODE="0660"
SUBSYSTEM=="spidev", GROUP="spi", MODE="0660"
SUBSYSTEM=="bcm2835-gpiomem", GROUP="gpio", MODE="0660"
SUBSYSTEM=="rpivid-*", GROUP="video", MODE="0660"

KERNEL=="vcsm-cma", GROUP="video", MODE="0660"
SUBSYSTEM=="dma_heap", GROUP="video", MODE="0660"

SUBSYSTEM=="gpio", GROUP="gpio", MODE="0660"
SUBSYSTEM=="gpio", KERNEL=="gpiochip*", ACTION=="add", PROGRAM="/bin/sh -c
'chgrp -R gpio /sys/class/gpio && chmod -R g=u /sys/class/gpio'"
SUBSYSTEM=="gpio", ACTION=="add", PROGRAM="/bin/sh -c 'chgrp -R gpio /sys%p &&
chmod -R g=u /sys%p'"

# PWM export results in a "change" action on the pwmchip device (not "add" of
a new device), so match actions other than "remove".
SUBSYSTEM=="pwm", ACTION!="remove", PROGRAM="/bin/sh -c 'chgrp -R gpio /sys%p
&& chmod -R g=u /sys%p'"

KERNEL=="ttyAMA0", PROGRAM="/bin/sh -c '\
      ALIASES=/proc/device-tree/aliases; \
      if cmp -s $$ALIASES/uart0 $$ALIASES/serial0; then \
            echo 0;\
      elif cmp -s $$ALIASES/uart0 $$ALIASES/serial1; then \
            echo 1; \
      else \
            exit 1; \
      fi\
'", SYMLINK+="serial%c"

KERNEL=="ttyAMA1", PROGRAM="/bin/sh -c '\
      ALIASES=/proc/device-tree/aliases; \
      if [ -e /dev/ttyAMA0 ]; then \
            exit 1; \
      elif cmp -s $$ALIASES/uart0 $$ALIASES/serial0; then \
            echo 0;\
      elif cmp -s $$ALIASES/uart0 $$ALIASES/serial1; then \
            echo 1; \
      else \
            exit 1; \
      fi\
'", SYMLINK+="serial%c"

KERNEL=="ttyS0", PROGRAM="/bin/sh -c '\
      ALIASES=/proc/device-tree/aliases; \
      if cmp -s $$ALIASES/uart1 $$ALIASES/serial0; then \
            echo 0; \
      elif cmp -s $$ALIASES/uart1 $$ALIASES/serial1; then \
            echo 1; \
      else \
            exit 1; \
      fi \
'", SYMLINK+="serial%c"

ACTION=="add", SUBSYSTEM=="vtconsole", KERNEL=="vtcon1", RUN+="/bin/sh -c '\
      if echo RPi-Sense FB | cmp -s /sys/class/graphics/fb0/name; then \
            echo 0 > /sys$devpath/bind; \
      fi; \
'"
```

```
ACTION=="add", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="7523",
SYMLINK+="SenseCAPdongle"

ACTION=="add", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="55d3",
SYMLINK+="EPEVERdongle"
```




### 2.9. Execução periódica do script Python

Adicionar ao topo do epever.py:

```
#! /usr/bin/python3
```

tornar o script executável com:

```
$ sudo chmod +x epever.py
```

Abrir ficheiro

```
$ crontab -e
```

*/2 * * * * /home/man4health/man4health/EPEVER/epever.py

## 3. Cabo RS485/USB alternativo

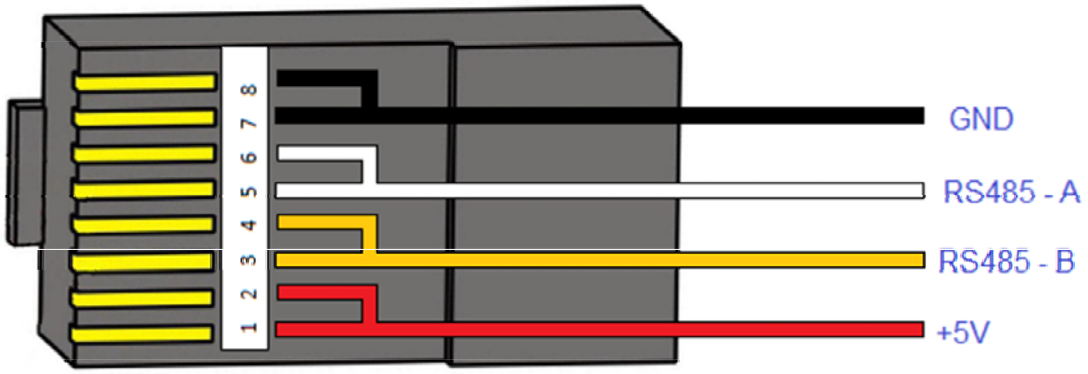
Na altura do projeto, foi encomendado um cabo (bridge RS485/USB) à empresa CHL. No entanto, eles enviaram um trocado onde a ficha de ligação à unidade de gestão da bateria em vez de ser RJ45 era um formato DIN com 4 pinos. Entretanto, e como não era claro a relação entre o pinout dos dois tipos de fichas, foi adquirido um segundo cabo com a ficha correta. Entretanto, através de processo simples de tentativa e erro, foi possível determinar o pinout do cabo que foi adquirido inicialmente e esta seção trata de documentar o processo e os resultados obtidos dessa abordagem.

O cabo, depois de cortado, deixava ver quatro condutores de cores diferentes: vermelho, preto, amarelo e branco. Pelo código de cores, era provável que os condutores vermelho e preto se referissem à saída de tensão DC (5 Volts) e os restantes dois condutores ao protocolo RS485 (A/B).

Efetivamente, usando um multímetro, foi possível comprovar que a tensão entre esses dois condutores era, aproximadamente, 5V. Ficou por desvendar qual dos condutores restantes se referiam ao A e ao B do protocolo. Usando o Modbus Poll não foi difícil descobrir a sequência correta. No entanto, foi necessário configurar o driver associado ao cabo (o que não era necessário no cabo que foi adquirido no AliExpress).

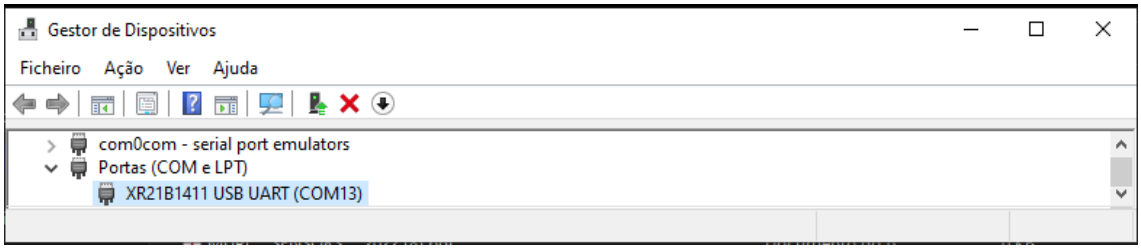
## 3.1. Cabo

Na imagem em baixo apresenta-se a configuração dos condutores e a sua conexão a uma ficha RJ45.

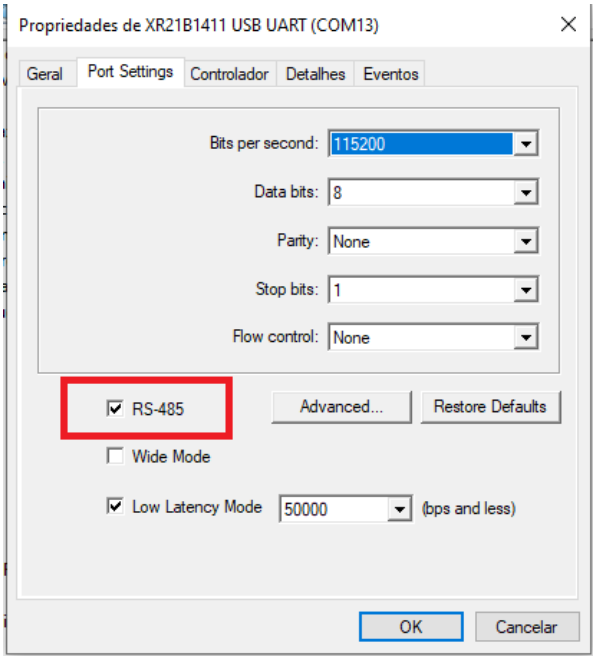



## 3.2. Device Manager

Depois de inserir a bridge RS485/USB no computador, o dispositivo foi enumerado como se mostra na figura a seguir.

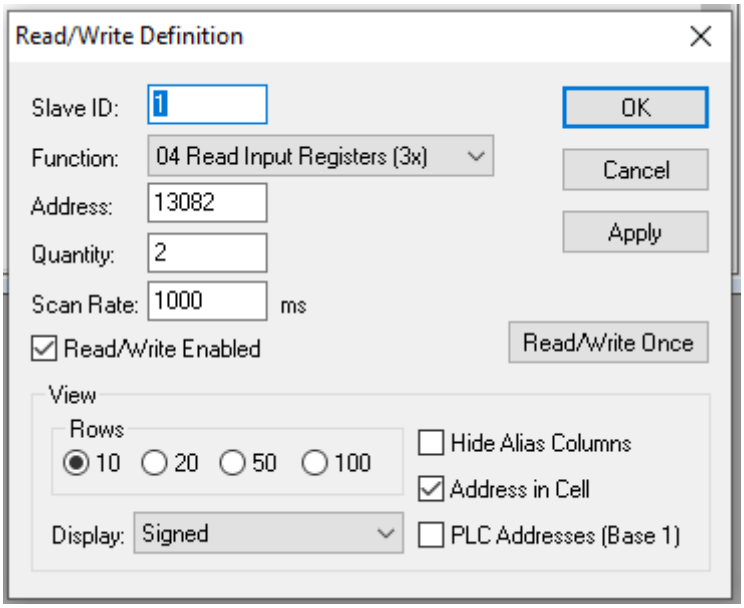


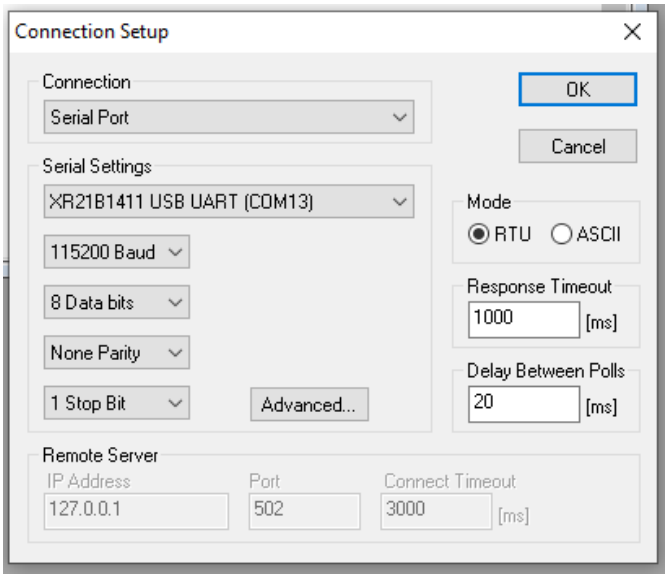
Foi preciso algum tempo para descobrir que para este cabo, e ao contrário do que aconteceu para o segundo que foi adquirido via AliExpress, era necessário parametrizar o driver como se mostra na figura seguinte:



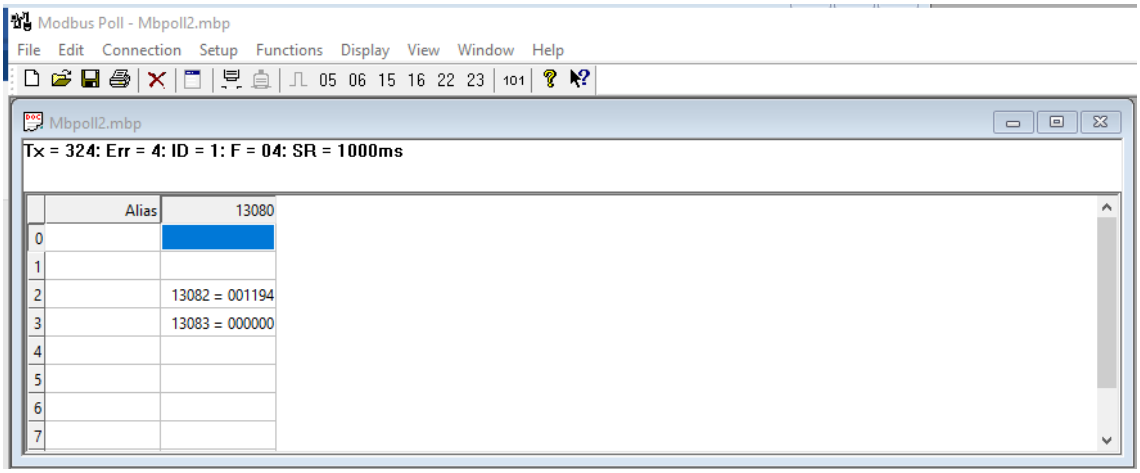

## 3.3. Modbus Poll

Para testar o funcionamento do cabo, foi utilizado o software Modbus Poll parametrizado como se mostra na duas próximas figuras.






O resultado obtido após execução do software pode ser observado na figura que se segue:



# 4. Modelo de dados

```
BatteryManagement:
  title: Battery Management
  type: object
  description: Masurements obtained from the battery management systems
  properties:
    id:
      description: Battery Management ID
      type: string
      format: uri
      example: urn:ngsi-ld:WBatteryManagement:EPEVERtracer01
      x-ngsi:
        type: Property

    type:
      description: NGSI Entity type
      title: Type
      enum:
        - BatteryManagement
      type: string
      x-ngsi:
        type: Property
        model: https://schema.org/Text

    PVarrayInputVoltage:
      description: PV array input voltage
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
            type: Property
            units: VLT
        unitCode:
          type: string
        observedAt:
          description: The date and time of this observation as defined by RFC 3339
          type: string
          format: date-time
          x-ngsi:
            model: https://schema.org/DateTime
            type: Property

    PVarrayInputCurrent:
      description: PV array input current
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
            type: Property
            units: AMP
        unitCode:
          type: string
        observedAt:
          description: The date and time of this observation as defined by RFC 3339
          type: string
          format: date-time
          x-ngsi:
            model: https://schema.org/DateTime
            type: Property

    PVarrayInputPower:
      description: PV array input power
      type: object
      properties:
        value:
          anyOf:
          - type: string
```

```
        - type: number
        x-ngsi:
          type: Property
          units: WTT
      unitCode:
        type: string
      observedAt:
        description: The date and time of this observation as defined by RFC 3339
        type: string
        format: date-time
        x-ngsi:
          model: https://schema.org/DateTime
          type: Property

  loadVoltage:
    description: Load voltage
    type: object
    properties:
      value:
        anyOf:
        - type: string
        - type: number
        x-ngsi:
          type: Property
          units: VLT
      unitCode:
        type: string
      observedAt:
        description: The date and time of this observation as defined by RFC 3339
        type: string
        format: date-time
        x-ngsi:
          model: https://schema.org/DateTime
          type: Property

  loadCurrent:
    description: Load current
    type: object
    properties:
      value:
        anyOf:
        - type: string
        - type: number
        x-ngsi:
          type: Property
          units: AMP
      unitCode:
        type: string
      observedAt:
        description: The date and time of this observation as defined by RFC 3339
        type: string
        format: date-time
        x-ngsi:
          model: https://schema.org/DateTime
          type: Property

  loadPower:
    description: Load power
    type: object
    properties:
      value:
        anyOf:
        - type: string
        - type: number
        x-ngsi:
          type: Property
          units: WTT
      unitCode:
        type: string
      observedAt:
        description: The date and time of this observation as defined by RFC 3339
        type: string
        format: date-time
        x-ngsi:
          model: https://schema.org/DateTime
          type: Property
```

```
    batteryTemperature:
      description: Battery temperature
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
            type: Property
            units: CEL
        unitCode:
          type: string
        observedAt:
          description: The date and time of this observation as defined by RFC 3339
          type: string
          format: date-time
          x-ngsi:
            model: https://schema.org/DateTime
            type: Property

    deviceTemperature:
      description: Device temperature
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
            type: Property
            units: CEL
        unitCode:
          type: string
        observedAt:
          description: The date and time of this observation as defined by RFC 3339
          type: string
          format: date-time
          x-ngsi:
            model: https://schema.org/DateTime
            type: Property

    batterySOC:
      description: The remaining capacity percentage of the battery
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
            type: Property
            units: P1
        unitCode:
          type: string
        observedAt:
          description: The date and time of this observation as defined by RFC 3339
          type: string
          format: date-time
          x-ngsi:
            model: https://schema.org/DateTime
            type: Property

    batteryRealRatedVoltage:
      description: Current system rated voltage
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
            type: Property
            units: VLT
        unitCode:
          type: string
        observedAt:
```

```
        description: The date and time of this observation as defined by RFC 3339
        type: string
        format: date-time
        x-ngsi:
          model: https://schema.org/DateTime
          type: Property

  batteryStatus:
    description: Battery status (16 bits)
    type: object
    properties:
      value:
        anyOf:
        - type: string
        - type: number
        x-ngsi:
          type: Property
      observedAt:
        description: The date and time of this observation as defined by RFC 3339
        type: string
        format: date-time
        x-ngsi:
          model: https://schema.org/DateTime
          type: Property

  chargingEquipmentStatus:
    description: Charging equipment status (16 bits)
    type: object
    properties:
      value:
        anyOf:
        - type: string
        - type: number
        x-ngsi:
          type: Property
      observedAt:
        description: The date and time of this observation as defined by RFC 3339
        type: string
        format: date-time
        x-ngsi:
          model: https://schema.org/DateTime
          type: Property

  dischargingEquipmentStatus:
    description: Discharging equipment status (16 bits)
    type: object
    properties:
      value:
        anyOf:
        - type: string
        - type: number
        x-ngsi:
          type: Property
      observedAt:
        description: The date and time of this observation as defined by RFC 3339
        type: string
        format: date-time
        x-ngsi:
          model: https://schema.org/DateTime
          type: Property

  maximumBatteryVoltageToday:
    description: Maximum battery voltage today
    type: object
    properties:
      value:
        anyOf:
        - type: string
        - type: number
        x-ngsi:
          type: Property
          units: VLT
      unitCode:
        type: string
      observedAt:
        description: The date and time of this observation as defined by RFC 3339
        type: string
```

```
        format: date-time
        x-ngsi:
          model: https://schema.org/DateTime
          type: Property

    minimumBatteryVoltageToday:
      description: Minimum battery voltage today
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
            type: Property
            units: VLT
        unitCode:
          type: string
        observedAt:
          description: The date and time of this observation as defined by RFC 3339
          type: string
          format: date-time
          x-ngsi:
            model: https://schema.org/DateTime
            type: Property

    consumedEnergyToday:
      description: Consumed energy today
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
            type: Property
            units: KWH
        unitCode:
          type: string
        observedAt:
          description: The date and time of this observation as defined by RFC 3339
          type: string
          format: date-time
          x-ngsi:
            model: https://schema.org/DateTime
            type: Property

    consumedEnergyMonth:
      description: Consumed energy this month
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
            type: Property
            units: KWH
        unitCode:
          type: string
        observedAt:
          description: The date and time of this observation as defined by RFC 3339
          type: string
          format: date-time
          x-ngsi:
            model: https://schema.org/DateTime
            type: Property

    consumedEnergyYear:
      description: Consumed energy this year
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
```

```
          type: Property
          units: KWH
      unitCode:
        type: string
      observedAt:
        description: The date and time of this observation as defined by RFC 3339
        type: string
        format: date-time
        x-ngsi:
          model: https://schema.org/DateTime
          type: Property

  totalConsumedEnergy:
    description: Total consumed energy
    type: object
    properties:
      value:
        anyOf:
        - type: string
        - type: number
        x-ngsi:
          type: Property
          units: KWH
      unitCode:
        type: string
      observedAt:
        description: The date and time of this observation as defined by RFC 3339
        type: string
        format: date-time
        x-ngsi:
          model: https://schema.org/DateTime
          type: Property

  generatedEnergyToday:
    description: Generated energy today
    type: object
    properties:
      value:
        anyOf:
        - type: string
        - type: number
        x-ngsi:
          type: Property
          units: KWH
      unitCode:
        type: string
      observedAt:
        description: The date and time of this observation as defined by RFC 3339
        type: string
        format: date-time
        x-ngsi:
          model: https://schema.org/DateTime
          type: Property

  generatedEnergyMonth:
    description: Generated energy this month
    type: object
    properties:
      value:
        anyOf:
        - type: string
        - type: number
        x-ngsi:
          type: Property
          units: KWH
      unitCode:
        type: string
      observedAt:
        description: The date and time of this observation as defined by RFC 3339
        type: string
        format: date-time
        x-ngsi:
          model: https://schema.org/DateTime
          type: Property

  generatedEnergyYear:
    description: Generated energy this year
```

```
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
            type: Property
            units: KWH
        unitCode:
          type: string
        observedAt:
          description: The date and time of this observation as defined by RFC 3339
          type: string
          format: date-time
          x-ngsi:
            model: https://schema.org/DateTime
            type: Property

    totalGeneratedEnergy:
      description: Total generated energy
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
            type: Property
            units: KWH
        unitCode:
          type: string
        observedAt:
          description: The date and time of this observation as defined by RFC 3339
          type: string
          format: date-time
          x-ngsi:
            model: https://schema.org/DateTime
            type: Property

    batteryVoltage:
      description: Battery voltage
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
            type: Property
            units: VLT
        unitCode:
          type: string
        observedAt:
          description: The date and time of this observation as defined by RFC 3339
          type: string
          format: date-time
          x-ngsi:
            model: https://schema.org/DateTime
            type: Property

    batteryCurrent:
      description: PV array input power
      type: object
      properties:
        value:
          anyOf:
          - type: string
          - type: number
          x-ngsi:
            type: Property
            units: AMP
        unitCode:
          type: string
        observedAt:
          description: The date and time of this observation as defined by RFC 3339
          type: string
```

```
          format: date-time
          x-ngsi:
            model: https://schema.org/DateTime
            type: Property

    refControlBoard:
      title: Reference to -ControlBoard-
      description: Reference of the -Terrain- to which this sensor belongs.
      anyOf:
      - description: Property. Identifier format of any NGSI entity
        maxLength: 256
        minLength: 1
        pattern: ^[\w\-\.\{\}\$\+\*\[\]`|~^@!,:\\]+$
        type: string
      - description: Property. Identifier format of any NGSI entity
        format: uri
        type: string
      x-ngsi:
        type: Relationship

  required:
    - id
    - type
    - refDevice
```