

Logbook

MAN4HEALTH

João Paulo Coelho



*Conexão, aquisição e publicação de dados a partir da estação
meteorológica SenseCAP*

Setembro 2022

Conteúdo

1. Hardware e conexão do SenseCAP	4
2. Comunicação com SenseCAP via Modbus	9
2.1. Usando o Modbus Poll	10
2.2. Usando o <i>Serial Port Monitor</i>	12
3. Comunicação com python.....	15
3.1. Instalação do python.....	15
3.2. Script Python para comunicação com SenseCAP: ver 1.0.....	15
3.3. Utilizando o MinimalModbus no Python	16
3.4. Publicação e subscrição MQTT usando Python.....	18
3.5. Execução periódica em Python	19
4. Modelo de dados do SenseCAP e entidades de contexto.....	19
4.1.1. Criar o Service no FIWARE.....	21
5. Deploy no Raspberry Pi	27
5.1. Usando o Raspberry Pi (Modelo 4)	27
5.2. Raspberry Pi modelo 3 B+	28
5.2.1. Executar o script periodicamente	32
5.2.2. Flow no Node-Red para observar as publicações:	33
5.2.3. Executar o script pelo startup	34
5.2.4. Executar em Loop.....	34
5.3. Raspberry Pi modelo 1 B+	36
5.4. Binding do nome da porta série ao hardware.....	38
5.4.1. Novo script Python usando hardware binding port name	39
6. Credenciais utilizadas no projeto	40
6.1. Login:	40
6.2. Password:	41
7. Conversão entre irradiância solar e iluminância	41

1. Hardware e conexão do SenseCAP

A comunicação com a estação meteorológica pode ser feita usando dois protocolos distintos: MODBUS-RTU/ASCII (sobre RS485) e SDI-12. Este *logbook* explora apenas a utilização do primeiro. Para isso, é necessário adquirir uma bridge USB/RS485 como aquela que apresento na Figura 1 e que, à data da escrita deste documento, pode ser adquirida [aqui](#).



Figura 1 - Adaptador USB-RS485.

A instalação deste adaptador é trivial e, no final, este dispositivo é enumerado como se pode ver na Figura 2.

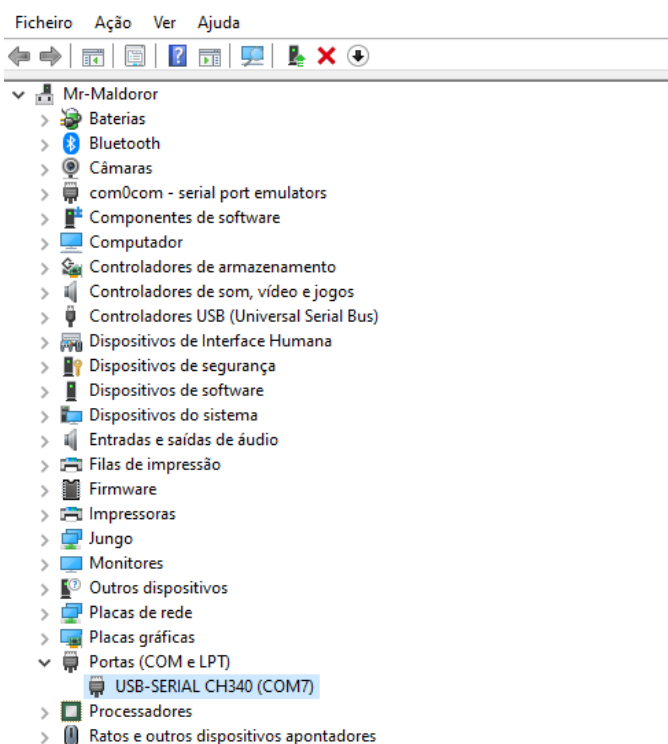


Figura 2 - Resultado da enumeração da bridge USB/RS485.

Internamente, o SenseCAP tem o aspeto representado na Figura 3. Como se pode ver, o cabo com seis condutores que leva os dados e a alimentação para fora da unidade está conetada a uma régua de terminais.



Figura 3 - Detalhe da ligação do cabo de alimentação e dados do SenseCAP.

De acordo com as folhas de dados, cada um desses condutores está associado a um sinal de dados ou alimentação como se ilustra na Figura 4.

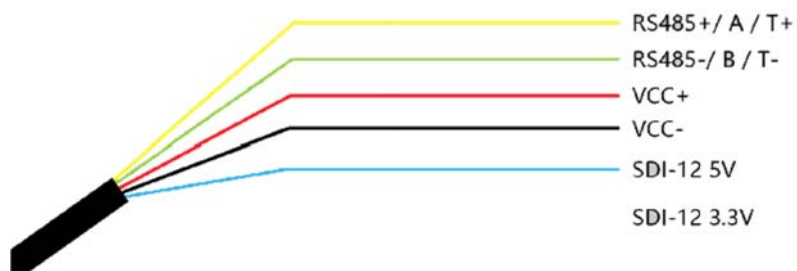


Figura 4 - Pinout do cabo de dados da estação SenseCAP.

Neste contexto, a ligação entre o SenseCAP e a bridge USB/RS485 foi realizada de acordo com a imagem da Figura 5.

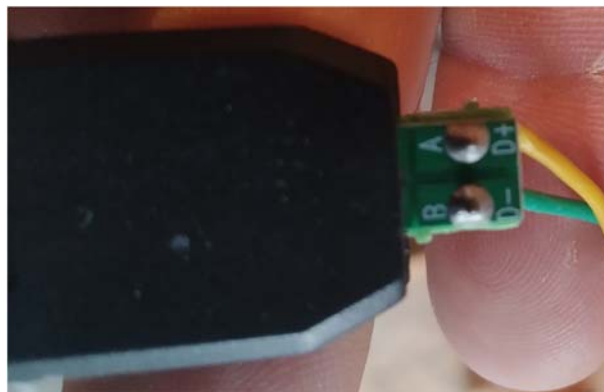


Figura 5 - Ligação entre o adaptador RS485 e os condutores provenientes do SenseCAP.

Como se pode ver, o condutor de cor amarelo é ligado ao terminal **A** do adaptador USB/RS485 e o condutor de cor verde ao terminal **B**. Para além das linhas de dados, é necessário também proceder à alimentação do módulo. De acordo com as folhas de dados, o módulo permite ser alimentado por tensões que vão desde os 3.6V e os 16V. Neste caso, e por questões que dizem respeito à uniformização dos valores da tensão de alimentação utilizada em outros dispositivos, uma tensão de 12V foi escolhida. A Figura 6 mostra as ligações associadas a um primeiro ensaio. Pode ver-se que os condutores branco e azul não são utilizados na configuração de comunicação adotada.

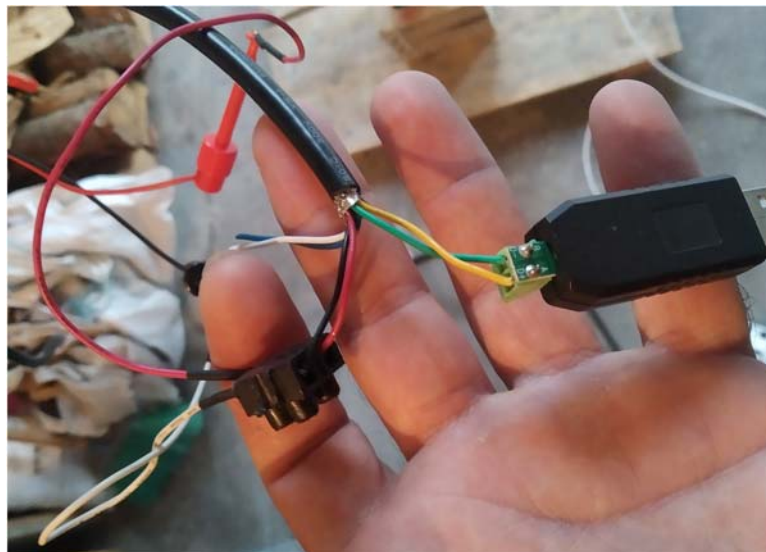


Figura 6 - Ligação do cabo proveniente do SenseCAP.

No final, o SenseCAP deve ser ligado de forma mais robusta ao sistema de comunicação e registo de dados. Para isso, optou-se pela utilização de conetores com SP13 (com IP68) como aqueles que se mostram na Figura 7.



Figura 7 - Conetor SP13 utilizado na ligação do SenseCAP ao armário onde a aquisição e publicação dos dados será realizado.

Em particular, as ligações dos quatro pinos deste conetor foram definidas como se mostra no diagrama da Figura 8. O pino 1 e 2 destinados à alimentação da unidade e os pinos 3 e 4 aos dados.

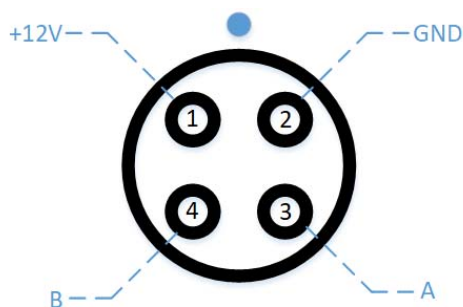


Figura 8 - Diagrama de ligação das linhas de alimentação e dados à ficha SP13.

Na Figura 9 apresenta-se a ligação, usando o conector SP13, ao conversor USB/RS485 e a uma fonte de alimentação de 12V usando um Jack de 1.3 mm.



Figura 9 - Ligação do SenseCAP no meu laboratório usando a ficha SP13.

No final, este conector será ligado a um dos portos de uma caixa de derivação que se irá situar no interior do poste onde o SenseCAP será instalado. A Figura 10 mostra o aspeto dessa caixa de distribuição onde, para já, podem ver-se dois terminais de ligação: um para a gateway e outro para o SenseCAP. No entanto, um terceiro conector deverá ser incluído de modo a poder conectar a câmara de vigilância.

A Figura 11 mostra o barramento de terminais que se encontra nesse caixa de derivação. A necessidade desta caixa de derivação é, fundamentalmente, para realizar a alteração da seção dos cabos que seguem para o interior do armário de distribuição e permitir que estes elementos possam ser facilmente removidos caso seja necessário.



Figura 10 - Caixa de distribuição onde o conetor será ligado.

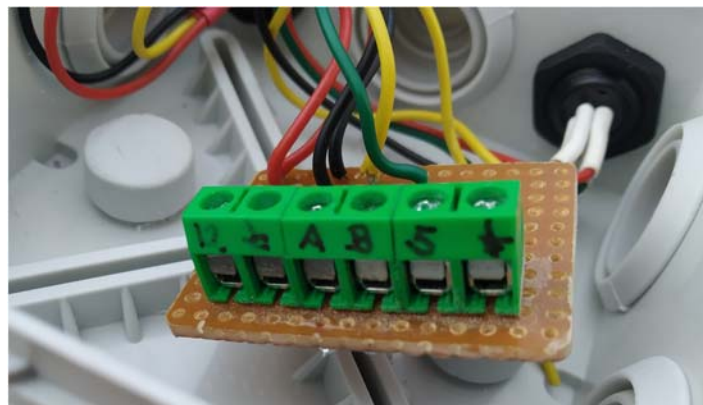


Figura 11 - Barra de terminais existente no interior da caixa de distribuição.

2. Comunicação com SenseCAP via Modbus

O módulo SenseCAP permite a medição de quatro grandezas meteorológicas: a temperatura do ar, a humidade do ar, a pressão atmosférica e a intensidade da luz. A Tabela 1 apresenta as gamas de medida e exatidões associadas a cada uma das quatro variáveis:

Tabela 1 - Características metrológicas do SenseCAP.

Variável	Gama de Medida	Exatidão	Resolução
Temperatura do Ar	-40°C a +80°C	±0.2 °C	±0.1 °C
Humidade do Ar	0% a 100%	±1.5 %	1%
Pressão Atmosférica	300 hPa a 1100 hPa	± 1 hPa	1 Pa
Intensidade Luminosa ¹	0 -188 kLux	±5 %	45 mLux

A máxima taxa de solicitação dos dados é 1 segundo. É necessário garantir um delay de 1 segundos após a ativação do SenseCAP antes de começar a realizar a poll dos dados.

As medições estão disponíveis nos seguintes registos:

Read-only Register				
Air Temperature	0x0000	int16 (complement), read-only	3, 4	Value=Register Value * 0.01 Unit: °C
Air Humidity	0x0001	uint1, read-only	3, 4	Value= Register Value * 0.01 Unit: %RH
Barometric Pressure MSB	0x0002	uint16, read-only	3, 4	Value=(uint32)(Register 0x0002 << 16 + Register 0x0003) Unit: Pa
Barometric Pressure LSB	0x0003	uint16, read-only	3, 4	
Light Intensity MSB	0x0004	uint16, read-only	3, 4	Value=((uint32)(Register 0x0004 << 16 + Register 0x0005)) * 0.001 Unit: Lux
Light Intensity LSB	0x0005	uint16, read-only	3, 4	

Figura 12 - Registos associado à leitura dos dados do SenseCAP.

¹ Uma fonte refere que, para converter de Lux para W/m², basta multiplicar pelo fator 0.0079. Outra fonte sugere que a irradiação solar de 1000 W/m² equivale a aproximadamente 120.000 Lux. Neste caso o fator de conversão é 0.00833.

Esta tabela permite, a partir dos valores dos respectivos registos, formar os valores medidos por cada um dos quatro sensores. Por exemplo, se a leitura dos registos forem aqueles apresentados na Tabela 2.

Tabela 2 - Exemplo de valores disponíveis nos registos num determinado instante de tempo.

Endereço (offset)	Valor		
0	2122	Temperatura do Ar	21.22 °C
1	5695	Humidade do Ar	59.95 %
2	1	Pressão Atmosférica	95.947 kPa
3	30411		
4	35	Intensidade luminosa	2331.844 Lux
5	-27452		

2.1. Usando o Modbus Poll

Numa primeira fase, a comunicação entre a estação meteorológica é feita recorrendo ao software **Modbus Poll** cuja versão completa pode ser encontrada, localmente, em:

...\Projeto MAN4HEALTH\Hardware\SenseCAP ORCH S4

Depois do software instalado², apresenta-se o front-end do programa com o formato ilustrado na Figura 13.

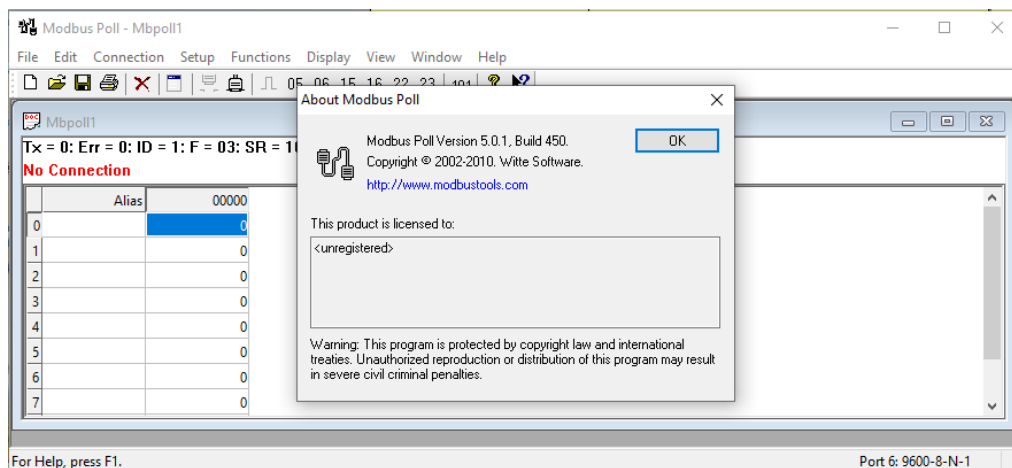


Figura 13 - Front-end do software Modbus Poll (ver. 5.0.1)

² Neste caso a versão 5.01. No entanto, os primeiros ensaios foram feitos com uma versão mais recente, mas que expirou passado 30 dias.

O primeiro passo, consiste na parametrização da ligação. Para isso, acedendo ao menu “Connection”, executar a opção “Connection Setup”. Uma vez aberta a janela, preencher os campos de acordo com a imagem da Figura 14.

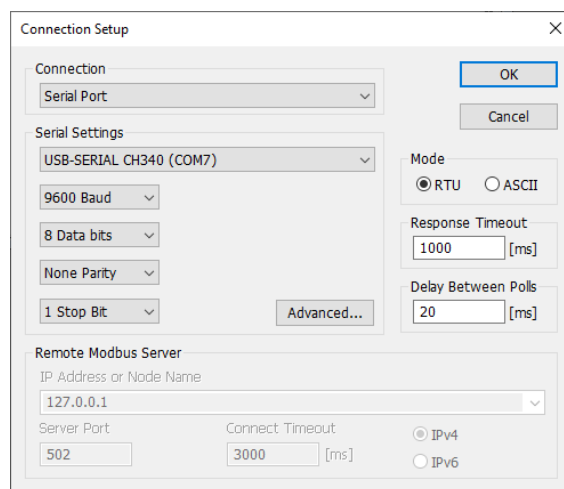


Figura 14 - Configuração da ligação.

Em seguida, no mesmo menu, aceder a “Read/Write Definition” e preencher o ID e registos de acordo com a imagem da Figura 15. Executando a ligação obtém-se o resultado ilustrado na Figura 16. O resultado do processo de leitura automática, com uma taxa igual a 1 segundo, é apresentado na Figura 16.

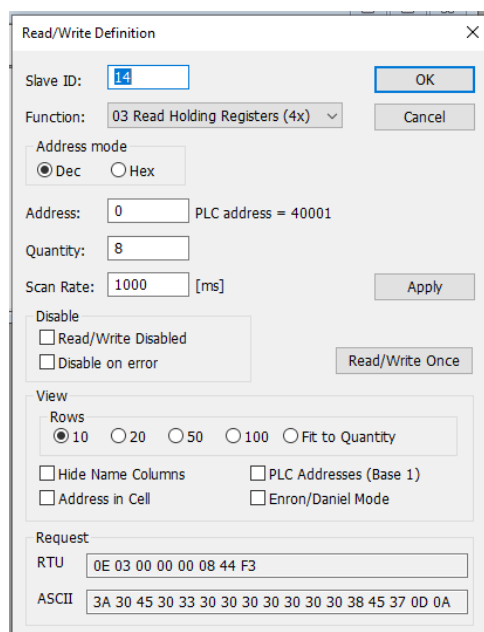


Figura 15 - Parametrização do endereço do Slave e dos registos de dados.

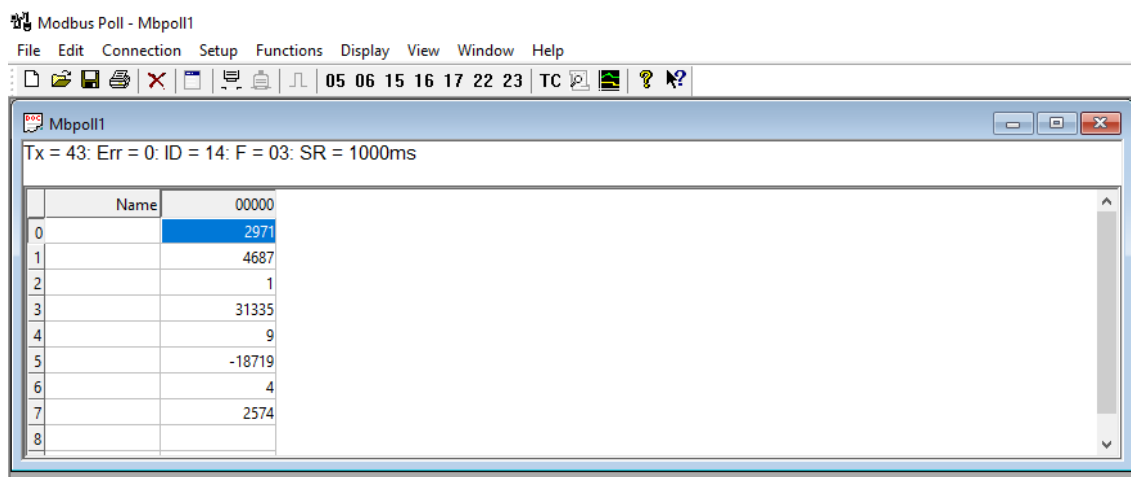


Figura 16 - Resultado da operação de leitura do SenseCAP via Modbus Poll.

2.2. Usando o Serial Port Monitor

Alternativamente, é possível comunicar com o SenseCAP utilizando o Serial Port Monitor³. Neste caso, depois de instalar o software, é necessário começar por criar uma nova sessão como se mostra na Figura 17

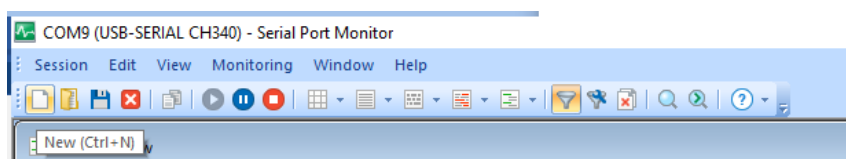


Figura 17 - Criação de uma nova sessão no Serial Port monitor

O que leva ao aparecimento da janela ilustrada na Figura 18 que deve ser parametrizada com os valores indicados. Neste caso, o adaptador USB/RS485 encontra-se associado à porta COM9 e deve ser selecionado o visualizador Modbus RTU.

Para que o SenseCAP envie os dados é preciso solicitá-los primeiro. Nas folhas de dados a forma de o fazer está bem detalhada pelo que aqui apenas apresento um exemplo. Admitindo que se pretende inquirir sobre o valor de todos os registos associados aos quatro sensores, cria-se um payload com o seguinte formato:

Tabela 3 - Formato do payload para solicitar os valores medidos pelos sensores.

Endereço do Slave (1 byte)	Código da função (1 byte)	Endereço do registo inicial (2 bytes)	Numero de registos a serem lidos (2 bytes)	CRC (2 bytes)
0E	03	0000	0008	44F3

³ Foi instalada a versão 7.0 disponível, localmente, na página do projeto.

O endereço do **slave** é o que vem por defeito⁴ programado no SenseCAP. O código da função é 0x03 (mais informação sobre o assunto no datasheet).

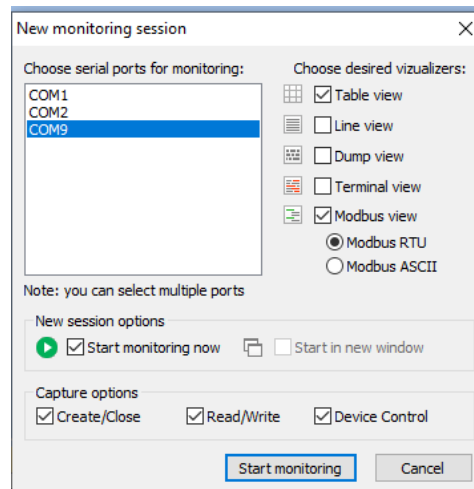


Figura 18 - Criação de nova sessão do tipo ModBus RTU.

Relativamente ao CRC, para este exemplo foi calculado usando o aplicativo WEB disponível no endereço: <https://crccalc.com/>. A Figura 19 mostra de que forma esta aplicação pode ser utilizada para determinar o CRC do payload atual⁵.

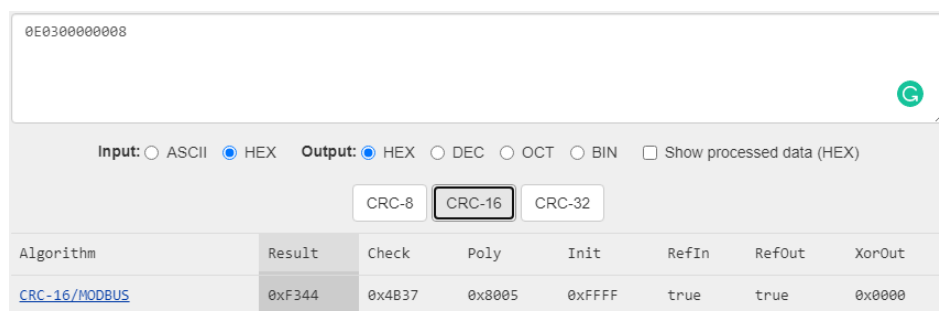


Figura 19 - Determinação do CRC usando o CRC CALC.

Finalmente, colocando a sequência hexadecimal:

0E 03 00 00 00 08 44 F3

no *Serial Port Monitor*, leva ao resultado apresentado na Figura 20.

⁴ Na realidade existe alguma informação contraditória no sentido de que existem locais onde se referem a esse endereço com 0x00.
⁵ Observe-se a inversão do low byte e do high byte.

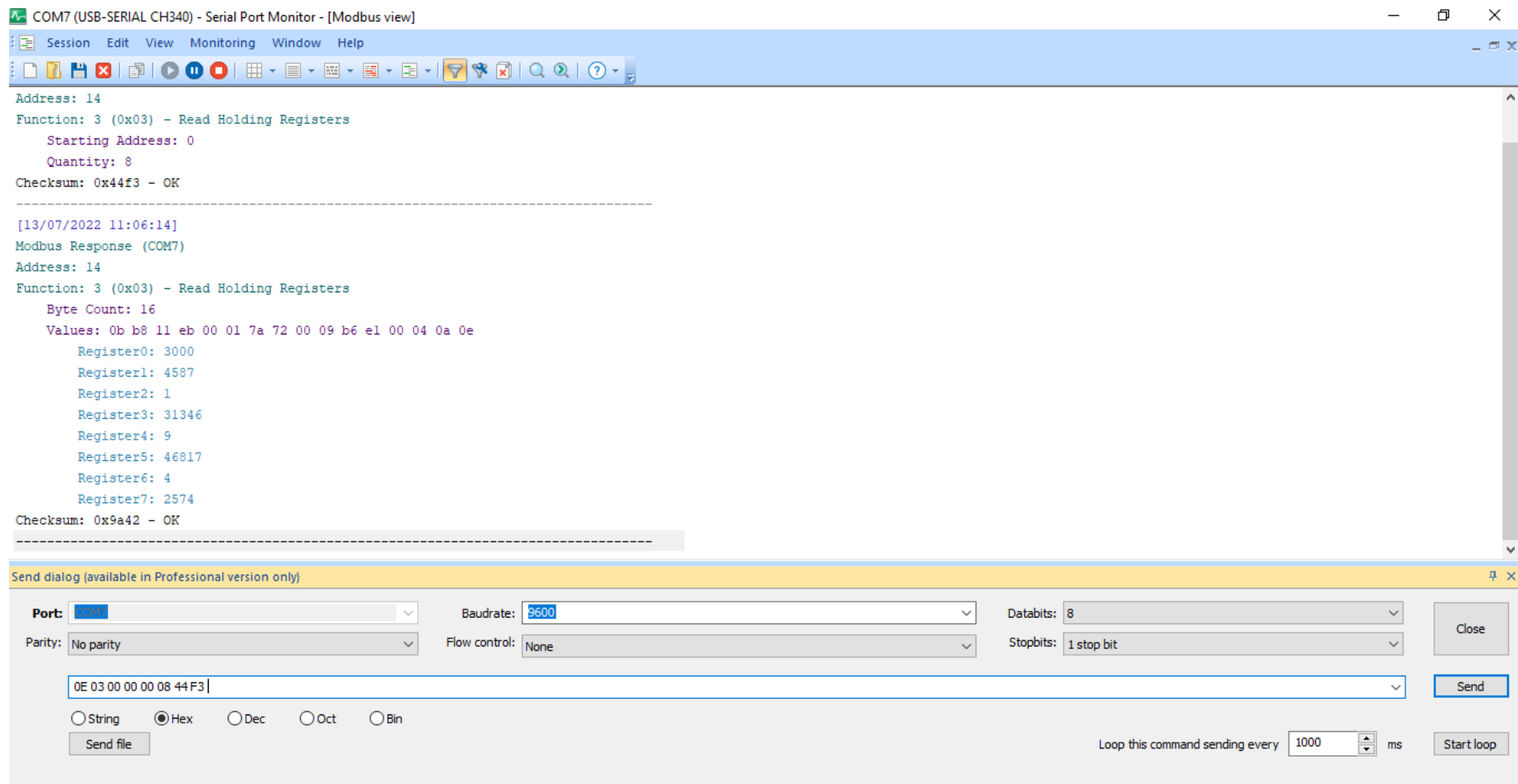


Figura 20 - Resposta do sensor à solicitação de envio dos dados dos sensores.

3. Comunicação com python

3.1. Instalação do python

- (i). Aceder à página da organização e descarregar a última versão para Windows a partir de:

<https://www.python.org/downloads/>

indicando, no processo de instalação, para incluir o *Path* no sistema operativo.

(Neste momento é a versão 3.10)

- (ii). Instalar o **spyder** através de:

```
>> python -m pip install spyder
```

- (iii). No **spyder**, instalar o **pyserial**:

```
In [2]: pip install pyserial
Collecting pyserial
  Using cached pyserial-3.5-py2.py3-none-any.whl (90 kB)
Installing collected packages: pyserial
Successfully installed pyserial-3.5
Note: you may need to restart the kernel to use updated packages.
```

3.2. Script Python para comunicação com SenseCAP: ver 1.0

```
import serial
import serial.rs485
import time
import codecs

ser = serial.Serial(port='COM7', baudrate=9600, bytesize=8, parity='N', stopbits=1,
timeout=None, xonxoff=0, rtscts=0)
#ser.rs485_mode = serial.rs485.RS485Settings()

#ser.is_open
frame = '0E030000000844F3'
ser.write(codecs.decode(frame, 'hex'))
time.sleep(5)
n = ser.in_waiting
print(n)
s = ser.read(n)
print(s)
ser.close()
shex = codecs.encode(s, 'hex')
```

```

print(shex)
orcap =
{'address':None,'function':None,'byte':None,'reg1':None,'reg2':None,'reg3':None,'reg4':None,'reg5':None,'reg6':None,'reg7':None,'reg8':None,'CRC':None}
if len(shex)>0:
    orcap['address']=chr(shex[0:2])

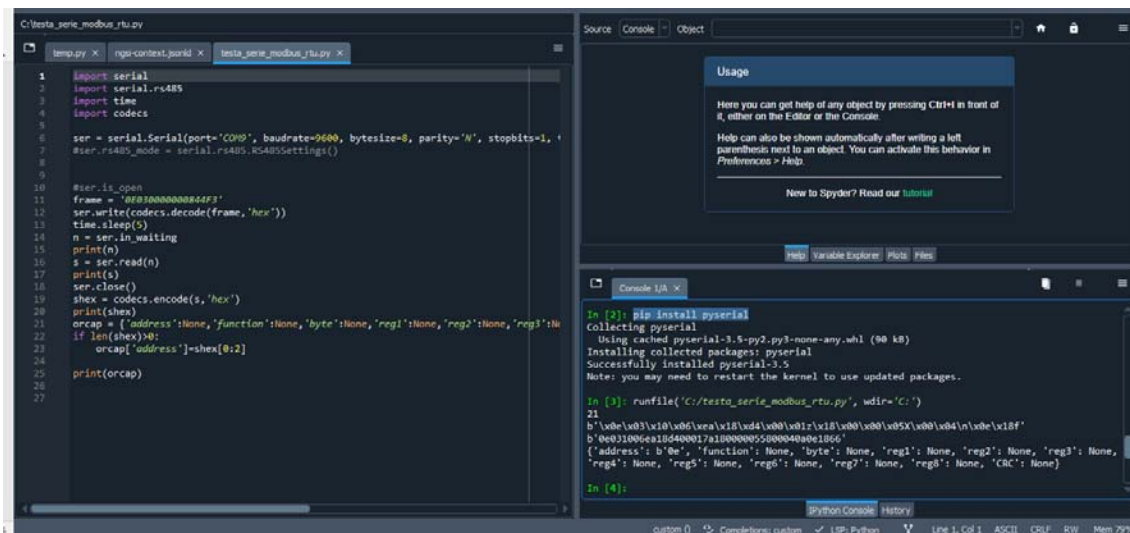
```

Cuja execução resulta em:

```

In [3]: runfile('C:/testa_serie_modbus_rtu.py', wdir='C:')
21
b'\x0e\x03\x10\x06\xea\x18\xd4\x00\x01z\x18\x00\x00\x05X\x00\x04\n\x0e\x18f'
b'\0e031006ea18d400017a180000055800040a0e1866'
{'address': b'\0e', 'function': None, 'byte': None, 'reg1': None, 'reg2': None, 'reg3': None, 'reg4': None, 'reg5': None, 'reg6': None, 'reg7': None, 'reg8': None, 'CRC': None}

```



3.3. Utilizando o MinimalModbus no Python

```
>> pip install -U minimalmodbus
```

```

In [4]: pip install -U minimalmodbus
Collecting minimalmodbus
  Downloading minimalmodbus-2.0.1-py3-none-any.whl (33 kB)
Requirement already satisfied: pyserial>=3.0 in c:\users\jpcoe\appdata\local\programs\python\python310\lib\site-packages (from minimalmodbus) (3.5)
Installing collected packages: minimalmodbus
Successfully installed minimalmodbus-2.0.1
Note: you may need to restart the kernel to use updated packages.

```

```
# -*- coding: utf-8 -*-
```



```

"""
Created on Fri Sep 30 12:43:58 2022

@author: jpcoelho
"""

import minimalmodbus

PORT='COM9'
TEMP_REGISTER = 0
HUM_REGISTER = 1
BAR_REGISTER = 2
LUX_REGISTER = 4

SLAVE_ADDRESS = 14
#Set up instrument
instrument = minimalmodbus.Instrument(PORT,SLAVE_ADDRESS,mode=minimalmodbus.MODE_RTU)

#Make the settings explicit
instrument.serial.baudrate = 9600          # Baud
instrument.serial.bytesize = 8
instrument.serial.parity = minimalmodbus.serial.PARITY_NONE
instrument.serial.stopbits = 1
instrument.serial.timeout = None           # seconds

# Good practice
instrument.close_port_after_each_call = True

instrument.clear_buffers_before_each_transaction = True

# Read temperature as a float
# if you need to read a 16 bit register use instrument.read_register()
temperature = instrument.read_register(TEMP_REGISTER)

# Read the humidity
humidity = instrument.read_register(HUM_REGISTER)

# Read the Atmospheric Pressure
barMSB= instrument.read_register(BAR_REGISTER)
barLSB= instrument.read_register(BAR_REGISTER+1)

# Read the Light Intensity
luxMSB= instrument.read_register(LUX_REGISTER)
luxLSB= instrument.read_register(LUX_REGISTER+1)

#Print the values
print('The Air Temperature is: %.1f °C\r' % (temperature/100.0))
print('The Air Humidity is: %.1f percent\r' % (humidity/100.0))
print('The Atmospheric Pressure is: %.1f Pa\r' % (((barMSB<<16)+barLSB)*0.01))
print('The Light Intensity is: %.1f lux\r' % (((luxMSB<<16)+luxLSB)*0.01))

```

3.4. Publicação e subscrição MQTT usando Python

Começa-se por se instalar o package PAHO através do comando:

```
pip install paho-mqtt
```

O script posto em funcionamento que permite, com sucesso, publicar mensagens em tópicos encontra-se a seguir:

```
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 30 14:51:06 2022

@author: jpcoe
"""

import paho.mqtt.client as mqttclient
import time

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

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

# Colcar Try e Catch
try:
    client.connect(broker_address,port=port)

    client.loop_start()

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

    client.publish("/json/teste","hello MQTT")
    client.loop_stop()
```

```
except:
    print("Impossível conectar ao broker MQTT ")
```

Neste caso, e para referência futura, o mosquito encontra-se a correr num container Docker com o listener ativo assim como a permissão de ligação a dispositivos anónimos. O `mosquitto_sub` é executado para monitorizar o tópico `/json/teste` no sentido de apurar se o script publica a mensagem “hello MQTT” no broker de forma correta.

3.5. Execução periódica em Python

4. Modelo de dados do SenseCAP e entidades de contexto

No Ubuntu (Windows ou Linux) clonar o repositório:

```
https://github.com/FIWARE/tutorials.Understanding-At-Context
```

Executar:

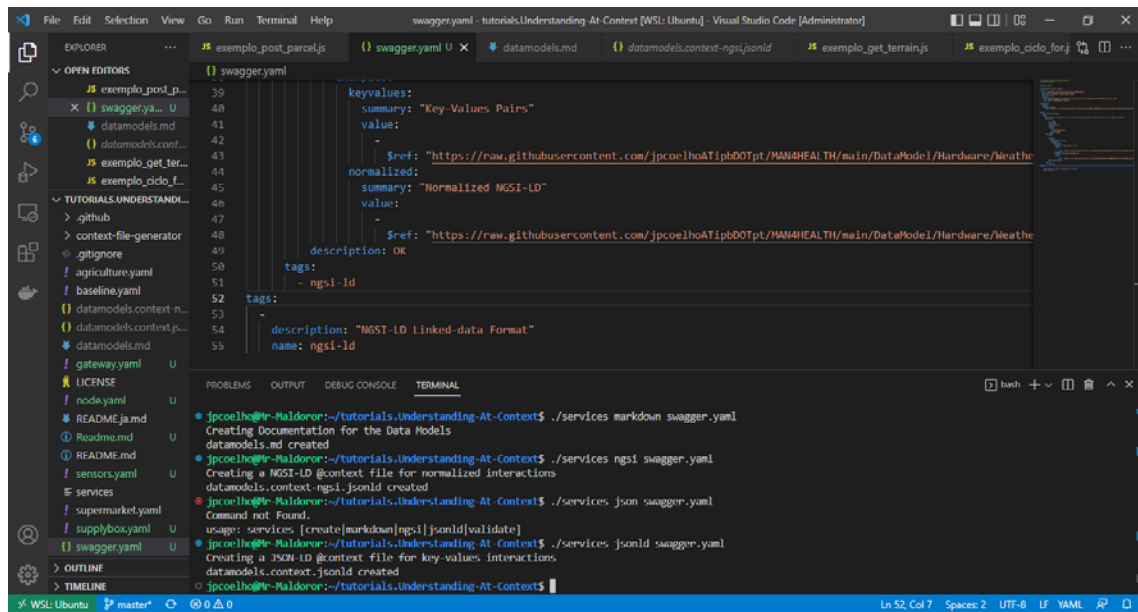
```
$ ./service create
```

```
jqcoelho@Mr-Maldoror: ~/tutorials.Understanding-At-Context
$ Code/bin:/mnt/c/Users/jpcoe/AppData/Roaming/npm:/snap/bin
declare -x PATH="/home/jpcoelho/tutorials.Understanding-At-Context"
declare -x SHELL="/bin/bash"
declare -x SHLVL="2"
declare -x TERM="xterm-256color"
declare -x USER="jqcoelho"
declare -x WSLLENV=""
declare -x WSL_DISTRO_NAME="Ubuntu"
declare -x WSL_INTEROP="/run/WSL/175_interop"
declare -x XDG_DATA_DIRS="/usr/local/share:/usr/share:/var/lib/snapd/desktop"
[+] Building 21.2s (10/10) FINISHED
-> [internal] load build definition from Dockerfile
-> => transferring dockerfile: 319B
-> [internal] load .dockerignore
-> => transferring context: 2B
-> [internal] load metadata for docker.io/library/node:14-slim
-> [auth] library/node:pull token for registry-1.docker.io
-> [internal] load build context
-> => transferring context: 17.94kB
-> [1/4] FROM docker.io/library/node:14-slim@sha256:c402462b7c20eb7e2bd8a4d9a285485332162638721c0311c75741d67ef24059
-> => resolve docker.io/library/node:14-slim@sha256:c402462b7c20eb7e2bd8a4d9a285485332162638721c0311c75741d67ef24059
-> => sha256:ef8e0bb59438e2d6073d9e2d083f81fb18eeaf4d51f9a11b9118899ded600 4.18kB / 4.18kB
-> => sha256:235001128bf2573ae9ddc4139e91c473a14e745e815723f8221cf49c942b07f 36.04MB / 36.04MB
-> => sha256:c402462b7c20eb7e2bd8a4d9a285485332162638721c0311c75741d67ef24059 276B / 276B
-> => sha256:c97da91ca064f0ca50b99fa08776fc5355ca8aed13ceb995193bd9065482901 1.37kB / 1.37kB
-> => sha256:bb29c1d8e73781295131bfdbfecaebc12e847a2169396a7825dbf8aaa2ccac 7.03kB / 7.03kB
-> => sha256:4b7b4a870e2f6766de51b0473f97a237d3d4df201b9df40318ca9060370b1 27.13MB / 27.13MB
-> => sha256:cf214a5df08c31f6021b7bd93a82392977af7b7d7967bb70a8928521e415 2.76MB / 2.76MB
-> => sha256:b082cb3b596f51b556ee6155ab02be7344a8ca0091e2e11ca025282d7e95dcd7 455B / 455B
-> => extracting sha256:4b7b4a870e2f6766de51b0473f97a237d3d4df201b9df40318ca9060370b1
-> => extracting sha256:e05e9bb59438e2d68a7309e2d083f81fb18eeaf4d51f9a11b9118899ded600
-> => extracting sha256:235001128bf2573ae9ddc4139e91c473a14e745e815723f8221cf49c942b07f
-> => extracting sha256:cf214a5df08c31f6021b7bd93a82392977af7b7d7967bb70a8928521e415
-> => extracting sha256:b082cb3b596f51b556ee6155ab02be7344a8ca0091e2e11ca025282d7e95dcd7
-> [2/4] COPY ./usr/src/app
-> [3/4] WORKDIR /usr/src/app
-> [4/4] RUN npm install --production && rm -rf /root/.npm/cache/* && chmod +x entrypoint.sh
-> exporting to image
-> exporting layers
-> writing image sha256:14ddcacfe78a440d4b195a1bac0a36b4f3480604f15345865fe4845b60a1df
-> naming to docker.io/library/contextgen
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
jqcoelho@Mr-Maldoror:~/tutorials.Understanding-At-Context$
```

Criar o ficheiro `swagger.yaml` juntamente com todas as suas dependências. Detalhes no github do projeto em:

<https://github.com/jpcoelhoATipbDOTpt/MAN4HEALTH/tree/main/DataModel/Hardware/WeatherStation>

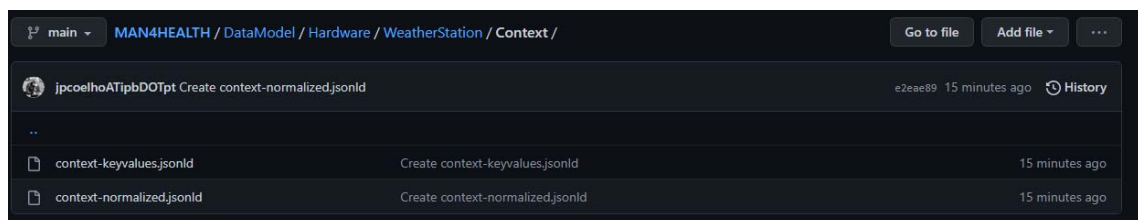
Clonar o `swagger` para o diretório do `understanding-At-context`:



E executar:

```
$ ./services/markdown/swagger.yaml  
  
$ ./services/ngsi/swagger.yaml  
  
$ ./services/json/swagger.yaml
```

Colocar esses ficheiros no arquivo `Context` no `github`



4.1.1. Criar o Service no FIWARE

```
curl --location --request POST 'http://localhost:4041/iot/services' \
--header 'Fiware-Service: man4health' \
--header 'Fiware-ServicePath: /terrain/parcel/device/hardware' \
--header 'Content-Type: application/json' \
--data-raw '{
  "services": [
    {
      "apikey": "Meteo",
      "cbroker": "http://orion:1026",
      "entity_type": "WeatherStation",
      "resource": "/iot/json",
      "attributes": [
        {
          "object_id": "AT",
          "type": "Property",
          "name": "airTemperature",
          "metadata": {
            "unitCode": {
              "type": "Text",
              "value": "CEL"
            }
          }
        },
        {
          "object_id": "AM",
          "type": "Property",
          "name": "airMoisture",
          "metadata": {
            "unitCode": {
              "type": "Text",
              "value": "P1"
            }
          }
        },
        {
          "object_id": "AP",
          "type": "Property",
          "name": "atmosphericPressure",
          "metadata": {
            "unitCode": {
              "type": "Text",
              "value": "PAL"
            }
          }
        },
        {
          "object_id": "SI",
          "type": "Property",
          "name": "solarIlluminance",
          "metadata": {
            "unitCode": {
              "type": "Text",
              "value": "LUX"
            }
          }
        },
        {
          "object_id": "SR",
          "type": "Property",
          "name": "solarIrradiance",
          "metadata": {
            "unitCode": {
              "type": "Text",

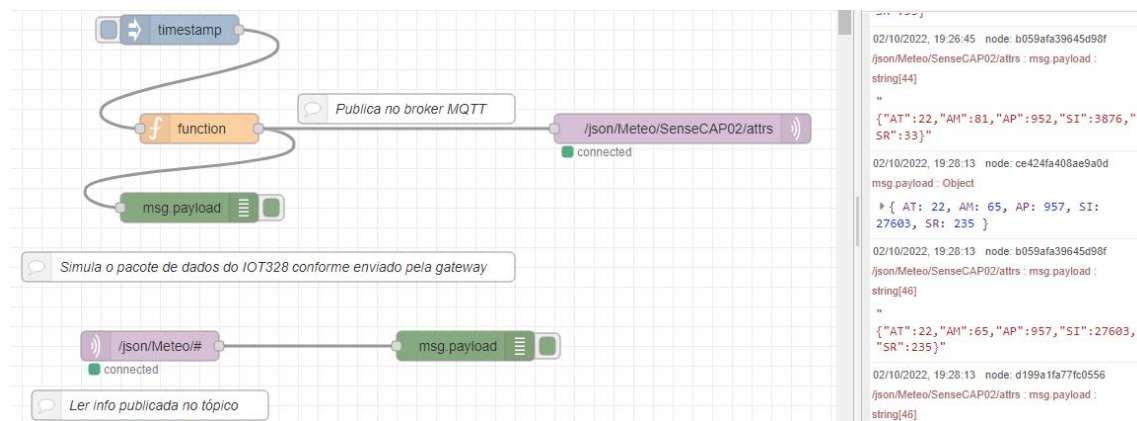
```

```

        "value": "D54"
      }
    }
  },
  "static_attributes": [
    {
      "name": "refTerrain",
      "type": "Relationship",
      "value": "urn:ngsi-ld:Terrain:MRDLSUC01"
    }
  ]
}
]
}
'

```

Publicar dados com Node-Red



```

[{"id":"329b4ac7b82e5d2b","type":"tab","label":"Flow
8","disabled":false,"info":"","{
  "id":"b424794711212d9d","type":"inject","z":"329b4ac7b82
e5d2b","name":"","props":[{"p":"payload"}, {"p":"topic","vt":"str"}],
  "repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payloadType":"date","x":180,"y":60,"wires":
  [[{"id":"de83dc47b61cdfc3"}]],
  "id":"de83dc47b61cdfc3","type":"function","z":"329b4ac7b82e5d2
b","name":"","func":"var gateway_payload={};\nvar airMoisture =
Math.round(100*(0.5+0.4*Math.random()));\nvar airTemperature =
Math.round(22+0.4*Math.random());\nvar atmosphericPressure =
Math.round(950+10*Math.random());\nvar solarIlluminance =
Math.round(30000*Math.random());\nvar solarIrradiance = Math.round(solarIlluminance *
0.0085);\n\nmsg.payload={\n  \\"AT\\": airTemperature,\n  \\"AM\\": airMoisture,\n
  \\"AP\\": atmosphericPressure,\n  \\"SI\\": solarIlluminance,\n  \\"SR\\":
solarIrradiance\n};\n\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":200,"y":160,"wir
es":[[{"id":"ce424fa408ae9a0d","type":"debug","z":"329b4ac7b82e5d2b","name":"","active":true,"tosidebar":true,"console":false,"tostatus
":false,"complete":"payload","targetType":"msg","statusVal":"","statusType":"auto","x":1
90,"y":240,"wires":[]}], [{"id":"5b3cc6e2d2a16f7c","type":"comment","z":"329b4ac7b82e5d2b",
"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\nl sensor de temperatura do ar\nl sensor de humidade do ar\nl sensor de radiação
solar\nl indicação de
bateria", "x":270, "y":300, "wires":[]}, {"id":"5f73a686ec0e8c66", "type":"mqtt
out", "z":"329b4ac7b82e5d2b", "name":"", "topic":"/json/Meteo/SenseCAP02/attrs", "qos":"2", "
retain":"","respTopic":"","contentType":"","userProps":"","correl":"","expiry":"","broke
r":"a7f68a32698ddb0f", "x":690, "y":160, "wires":[]}, {"id":"5348075958ac53cc", "type":"mqtt
in", "z":"329b4ac7b82e5d2b", "name":"", "topic":"/json/Meteo/#", "qos":"2", "datatype":"auto"
, "broker":"b6aedbb0641104f5", "nl":false, "rap":true, "rh":0, "x":150, "y":380, "wires":[["d19
9a1fa77fc0556"]]}, {"id":"d199a1fa77fc0556", "type":"debug", "z":"329b4ac7b82e5d2b", "name":
"", "active":true, "tosidebar":true, "console":false, "tostatus":false, "complete":"false", "s
tatusVal":"","statusType":"auto", "x":470, "y":380, "wires":[]}, {"id":"7616d8ad11511adf", "t
ype":"comment", "z":"329b4ac7b82e5d2b", "name":"Ler info publicada no
tópico", "info":"","x":150, "y":440, "wires":[]}, {"id":"208afa832fd4e54b", "type":"comment",
"z":"329b4ac7b82e5d2b", "name":"Publica no broker
MQTT", "info":"","x":410, "y":140, "wires":[]}, {"id":"a7f68a32698ddb0f", "type":"mqtt-
broker", "name":"LigacaoFIWARE", "broker":"localhost", "port":"1883", "clientId":"NODE_RED_C
lient", "usetls":false, "protocolVersion":"4", "keepalive":"60", "cleansession":true, "birthT
opic":"","birthQos":"0", "birthPayload":"","birthMsg":{"}, "closeTopic":"","closeQos":"0", "
closePayload":"","closeMsg":{"}, "willTopic":"","willQos":"0", "willPayload":"","willMsg":{"
}, "sessionExpiry":""}, {"id":"b6aedbb0641104f5", "type":"mqtt-
broker", "name":"","broker":"localhost", "port":"1883", "clientId":"","usetls":false, "proto
colVersion":"4", "keepalive":"60", "cleansession":true, "birthTopic":"","birthQos":"0", "bir
thPayload":"","birthMsg":{"}, "closeTopic":"","closeQos":"0", "closePayload":"","closeMsg":
{"}, "willTopic":"","willQos":"0", "willPayload":"","willMsg":{"}, "sessionExpiry":""}]

```

Depois de executar o código no Python:

```

# -*- coding: utf-8 -*-
"""
Created on Fri Sep 30 12:43:58 2022

@author: jpcoelho
"""

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

def on_connect(client, userdata, 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 = "Meteo"

```

```

deviceid = "SenseCAP01"
protocol = "json"
topic = "/" + protocol + "/" + apikey + "/" + deviceid + "/attrs"

# Configuração Modbus
PORT='COM10'
TEMP_REGISTER = 0
HUM_REGISTER = 1
BAR_REGISTER = 2
LUX_REGISTER = 4

SLAVE_ADDRESS = 14

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

# Parametros do instrumento
instrument.serial.baudrate = 9600          # Baud
instrument.serial.bytesize = 8
instrument.serial.parity = minimalmodbus.serial.PARITY_NONE
instrument.serial.stopbits = 1
instrument.serial.timeout = None           # seconds

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

# Le temperatura
airTemperature = (instrument.read_register(TEMP_REGISTER)/100.0)
# Le Humidade
airMoisture = (instrument.read_register(HUM_REGISTER)/100.0)
# Le Pressão Atmosférica
barMSB= instrument.read_register(BAR_REGISTER)
barLSB= instrument.read_register(BAR_REGISTER+1)
atmosphericPressure= (((barMSB<<16)+barLSB)*0.01)
# Le intensidade de radiação solar
luxMSB= instrument.read_register(LUX_REGISTER)
luxLSB= instrument.read_register(LUX_REGISTER+1)
solarIlluminance = ((luxMSB<<16)+barLSB)*0.01
# Mostra valores (debug apenas)
print('The Air Temperature is: %.1f °C\r' % airTemperature)
print('The Air Humidity is: %.1f percent\r' % airMoisture)
print('The Atmospheric Pressure is: %.1f Pa\r' % atmosphericPressure)
print('The Light Intensity is: %.1f lux\r' % solarIlluminance)

# Criar payload json

payload = {"AT": airTemperature,
           "AM": airMoisture,
           "AP": atmosphericPressure,
           "SI": solarIlluminance,
           "SR": solarIlluminance }

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)

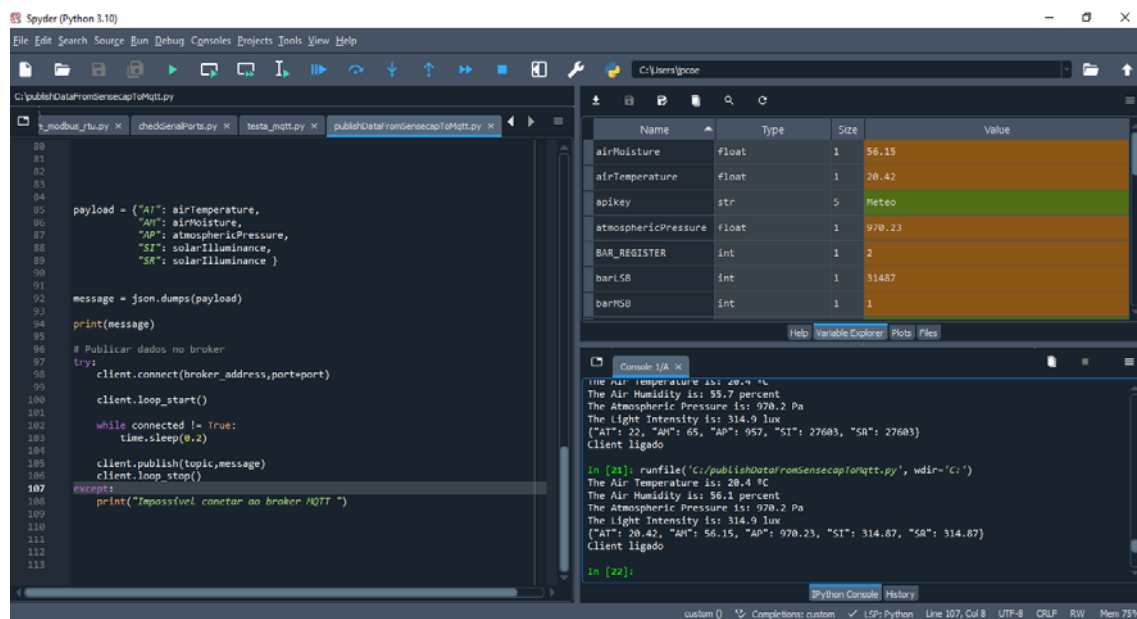
```



```

        client.loop_stop()
    except:
        print("Impossível conectar ao broker MQTT ")

```

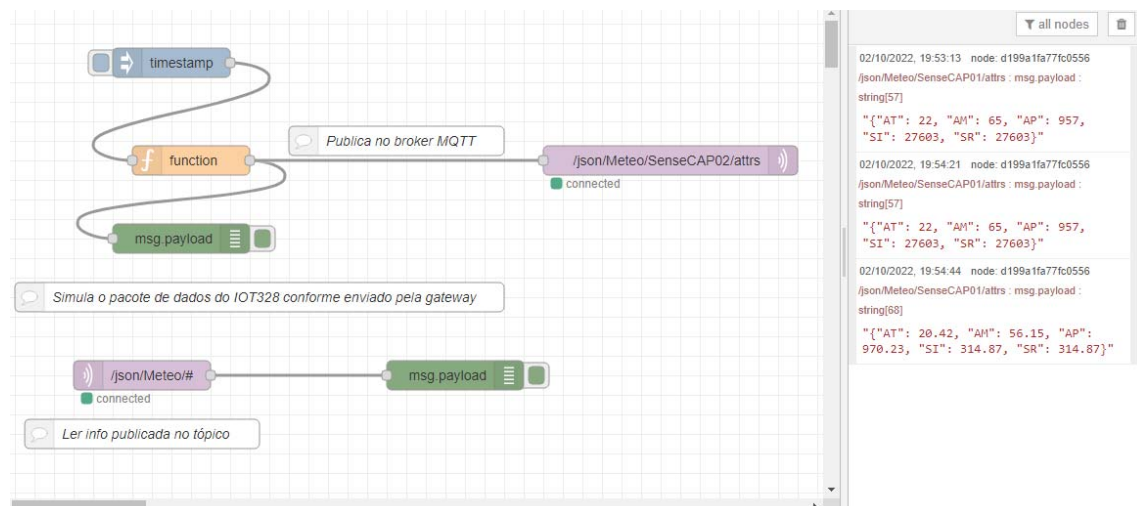


```

In [21]: runfile('C:/publishDataFromSensecapToMqtt.py', wdir='C:')
The Air Temperature is: 20.4 °C
The Air Humidity is: 56.1 percent
The Atmospheric Pressure is: 970.2 Pa
The Light Intensity is: 314.9 lux
{"AT": 20.42, "AM": 56.15, "AP": 970.23, "SI": 314.87, "SR": 314.87}
Client ligado

```

Ler o valor publicado no broker MQTT



Fui ao postman e executei:

```

curl --location --request GET 'http://localhost:1026/ngsi-ld/v1/entities/urn:ngsi-ld:WeatherStation:SenseCAP01' \

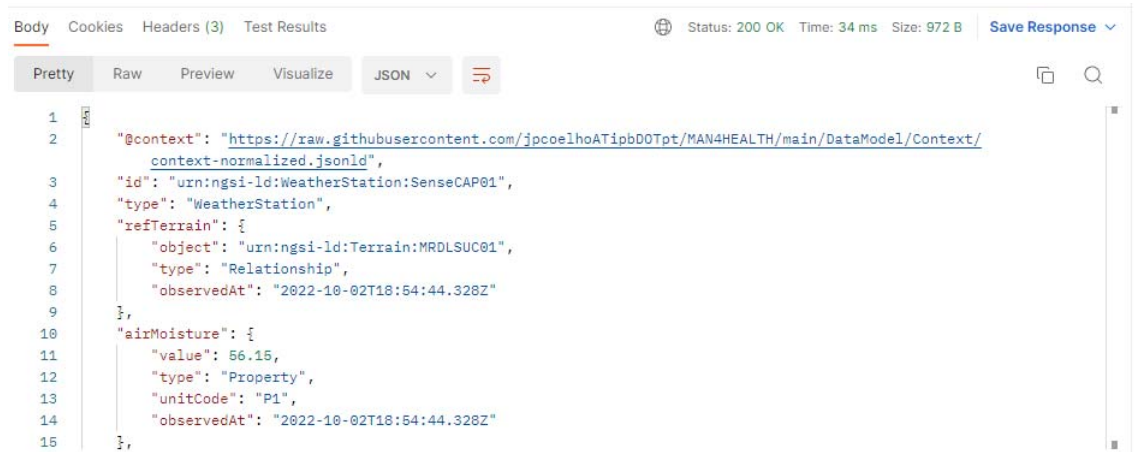
```

```
--header 'Fiware-Service: man4health' \

--header 'Accept: application/ld+json' \

--header                                     'Link:
<https://raw.githubusercontent.com/jpcoelhoATipbDOTpt/MAN4HEALTH/main/DataModel/Context/
context-normalized.jsonld>' \

--header 'Fiware-ServicePath: /terrain/parcel/device/hardware'
```



The screenshot shows a REST client interface with a status bar indicating 'Status: 200 OK', 'Time: 34 ms', and 'Size: 972 B'. The response is displayed in 'Pretty' format as a JSON object. The JSON structure includes a context link, an ID, a type, a reference to a terrain object, and several property objects with values and timestamps.

```
1  {
2    "@context": "https://raw.githubusercontent.com/jpcoelhoATipbDOTpt/MAN4HEALTH/main/DataModel/Context/
3    context-normalized.jsonld",
4    "id": "urn:ngsi-ld:WeatherStation:SenseCAP01",
5    "type": "WeatherStation",
6    "refTerrain": {
7      "object": "urn:ngsi-ld:Terrain:MRDLSUC01",
8      "type": "Relationship",
9      "observedAt": "2022-10-02T18:54:44.328Z"
10   },
11   "airMoisture": {
12     "value": 56.15,
13     "type": "Property",
14     "unitCode": "P1",
15     "observedAt": "2022-10-02T18:54:44.328Z"
16   },
17   "airTemperature": {
18     "value": 20.42,
19     "type": "Property",
20     "unitCode": "P1",
21     "observedAt": "2022-10-02T18:54:44.328Z"
22   },
23   "atmosphericPressure": {
24     "value": 970.23,
25     "type": "Property",
26     "unitCode": "PAL",
27     "observedAt": "2022-10-02T18:54:44.328Z"
28   },
29   "solarIlluminance": {
30     "value": 314.87,
31     "type": "Property",
32     "unitCode": "LUX",
33     "observedAt": "2022-10-02T18:54:44.328Z"
34   }
35 }
```

```
{
  "@context":
"https://raw.githubusercontent.com/jpcoelhoATipbDOTpt/MAN4HEALTH/main/DataModel/Context/context-normalized.jsonld",
  "id": "urn:ngsi-ld:WeatherStation:SenseCAP01",
  "type": "WeatherStation",
  "refTerrain": {
    "object": "urn:ngsi-ld:Terrain:MRDLSUC01",
    "type": "Relationship",
    "observedAt": "2022-10-02T18:54:44.328Z"
  },
  "airMoisture": {
    "value": 56.15,
    "type": "Property",
    "unitCode": "P1",
    "observedAt": "2022-10-02T18:54:44.328Z"
  },
  "airTemperature": {
    "value": 20.42,
    "type": "Property",
    "unitCode": "P1",
    "observedAt": "2022-10-02T18:54:44.328Z"
  },
  "atmosphericPressure": {
    "value": 970.23,
    "type": "Property",
    "unitCode": "PAL",
    "observedAt": "2022-10-02T18:54:44.328Z"
  },
  "solarIlluminance": {
    "value": 314.87,
    "type": "Property",
    "unitCode": "LUX",
    "observedAt": "2022-10-02T18:54:44.328Z"
  }
},
```

```
"solarIrradiance": {
  "value": 314.87,
  "type": "Property",
  "unitCode": "D54",
  "observedAt": "2022-10-02T18:54:44.328Z"
}
```

5. Deploy no Raspberry Pi

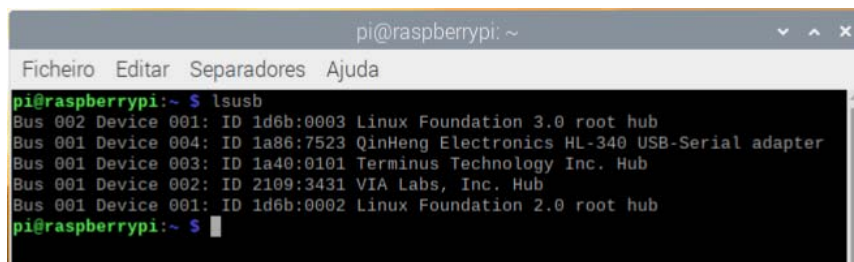
5.1. Usando o Raspberry Pi (Modelo 4)

```
import serial
import serial.rs485
import time
import codecs

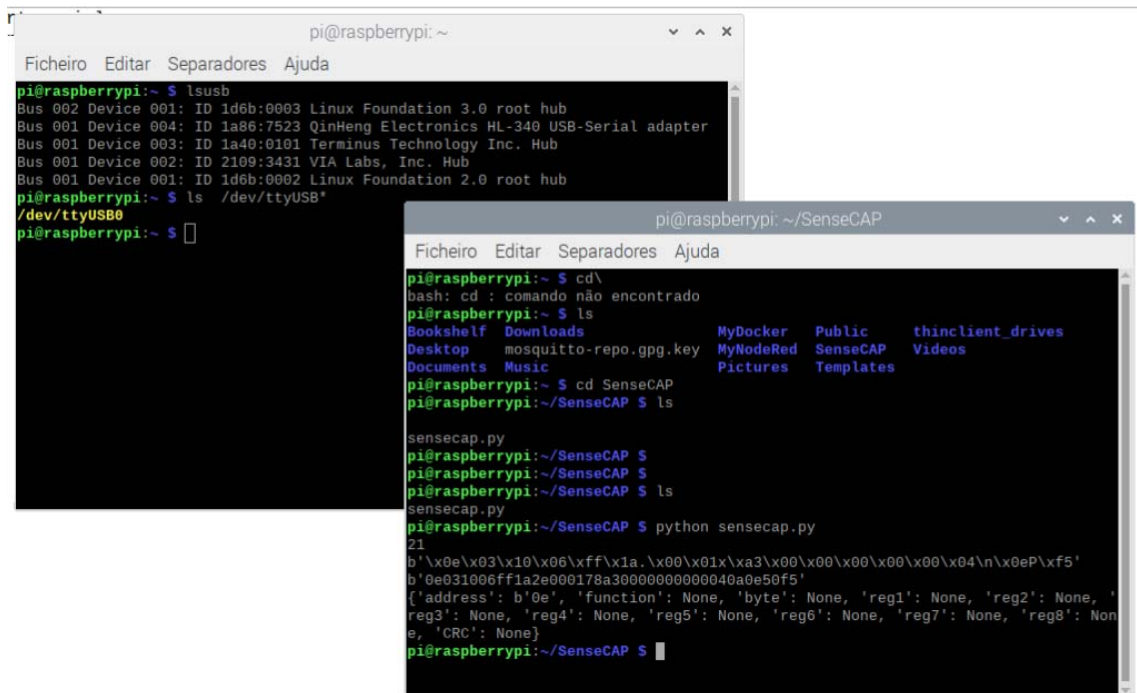
ser = serial.Serial(port='/dev/ttyUSB0', baudrate=9600, bytesize=8, parity='N',
stopbits=1, timeout=None, xonxoff=0, rtscts=0)
#ser.rs485_mode = serial.rs485.RS485Settings()

#ser.is_open
frame = '0E030000000844F3'
ser.write(codecs.decode(frame, 'hex'))
time.sleep(5)
n = ser.in_waiting
print(n)
s = ser.read(n)
print(s)
ser.close()
shex = codecs.encode(s, 'hex')
print(shex)
orcap = {
  'address':None, 'function':None, 'byte':None, 'reg1':None, 'reg2':None, 'reg3':None,
  'reg4':None, 'reg5':None, 'reg6':None, 'reg7':None, 'reg8':None, 'CRC':None}
if len(shex)>0:
    orcap['address']=shex[0:2]

print(orcap)
```



```
pi@raspberrypi: ~
Ficheiro  Editar  Separadores  Ajuda
pi@raspberrypi:~$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 1a86:7523 QinHeng Electronics HL-340 USB-Serial adapter
Bus 001 Device 003: ID 1a40:0101 Terminus Technology Inc. Hub
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
pi@raspberrypi:~$
```



```
pi@raspberrypi:~  
Ficheiro Editar Separadores Ajuda  
pi@raspberrypi:~$ lsusb  
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub  
Bus 001 Device 004: ID 1a86:7523 QinHeng Electronics HL-340 USB-Serial adapter  
Bus 001 Device 003: ID 1a40:0101 Terminus Technology Inc. Hub  
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub  
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub  
pi@raspberrypi:~$ ls /dev/ttyUSB*  
/dev/ttyUSB0  
pi@raspberrypi:~$  
  
pi@raspberrypi:~/SenseCAP  
Ficheiro Editar Separadores Ajuda  
pi@raspberrypi:~$ cd\  
bash: cd : comando não encontrado  
pi@raspberrypi:~$ ls  
Bookshelf Downloads MyDocker Public thinclient_drives  
Desktop mosquito-repo.gpg.key MyNodeRed SenseCAP Videos  
Documents Music Pictures Templates  
pi@raspberrypi:~$ cd SenseCAP  
pi@raspberrypi:~/SenseCAP$ ls  
  
sensecap.py  
pi@raspberrypi:~/SenseCAP$  
pi@raspberrypi:~/SenseCAP$  
pi@raspberrypi:~/SenseCAP$ ls  
sensecap.py  
pi@raspberrypi:~/SenseCAP$ python sensecap.py  
21  
b'\x0e\x03\x10\x06\xff\x1a.\x00\x01\x0a3\x00\x00\x00\x00\x040a0e50f5'  
b'\x0e031006ff1a2e000178a30000000000040a0e50f5'  
{'address': b'\x0e', 'function': None, 'byte': None, 'reg1': None, 'reg2': None, 'reg3': None, 'reg4': None, 'reg5': None, 'reg6': None, 'reg7': None, 'reg8': None, 'CRC': None}  
pi@raspberrypi:~/SenseCAP$
```

```
import serial  
import serial.rs485  
import time  
import codecs  
  
ser = serial.Serial(port='/dev/ttyUSB0', baudrate=9600, bytesize=8,  
parity='N', stopbits=1, timeout=None, xonxoff=0, rtscts=0)  
#ser.rs485_mode = serial.rs485.RS485Settings()  
  
#ser.is_open  
frame = '0E030000000844F3'  
ser.write(codecs.decode(frame, 'hex'))  
time.sleep(5)  
n = ser.in_waiting  
print(n)  
s = ser.read(n)  
print(s)  
ser.close()  
shex = codecs.encode(s, 'hex')  
print(shex)  
orcap =  
{ 'address': None, 'function': None, 'byte': None, 'reg1': None, 'reg2': None, 'reg3': None,  
  'reg4': None, 'reg5': None, 'reg6': None, 'reg7': None, 'reg8': None, 'CRC': None}  
if len(shex)>0:  
    orcap['address']=shex[0:2]  
  
print(orcap)
```

5.2. Raspberry Pi modelo 3 B+

Nesta seção, a versão 3 B+ da plataforma Raspberry Pi será utilizada para realizar a instalação da stack que permite a publicação dos dados meteorológicos da estação SenseCAP no broker MQTT.

O primeiro passo consistiu na instalação do sistema operativo. Neste caso foi utilizada a versão de 64 bits do Raspberry Pi OS. Primeiro, aceder ao site da Raspberry Pi e descarregar o Imager.

<https://www.raspberrypi.com/software/>

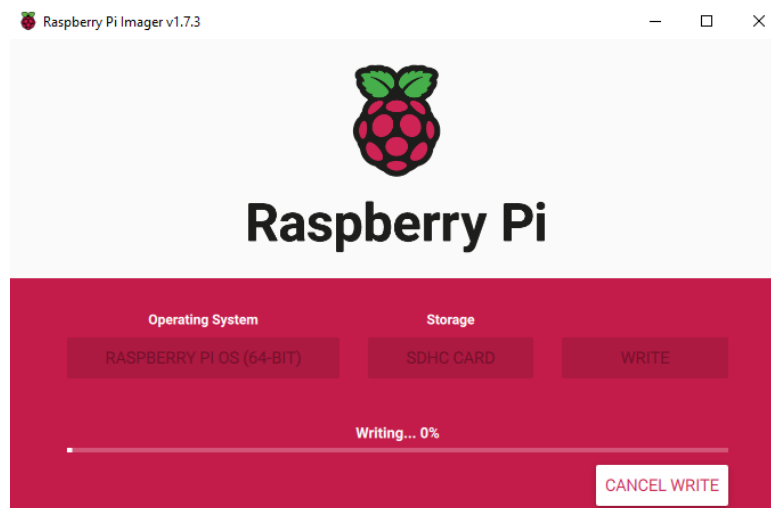


Figura 21 - Imager do Raspberry Pi e escolha do sistema operativo a ser criado no cartão SD.

Depois de instalado o sistema operativo, e localmente no Raspberry Pi, ativar o acesso por SSH e instalar o remote desktop através da seguinte sequência de comandos executados no terminal:

```
$ sudo apt update  
$ sudo apt upgrade  
$ sudo apt install xrdp
```

Mais detalhes sobre este processo de instalação podem ser consultados a partir do site:

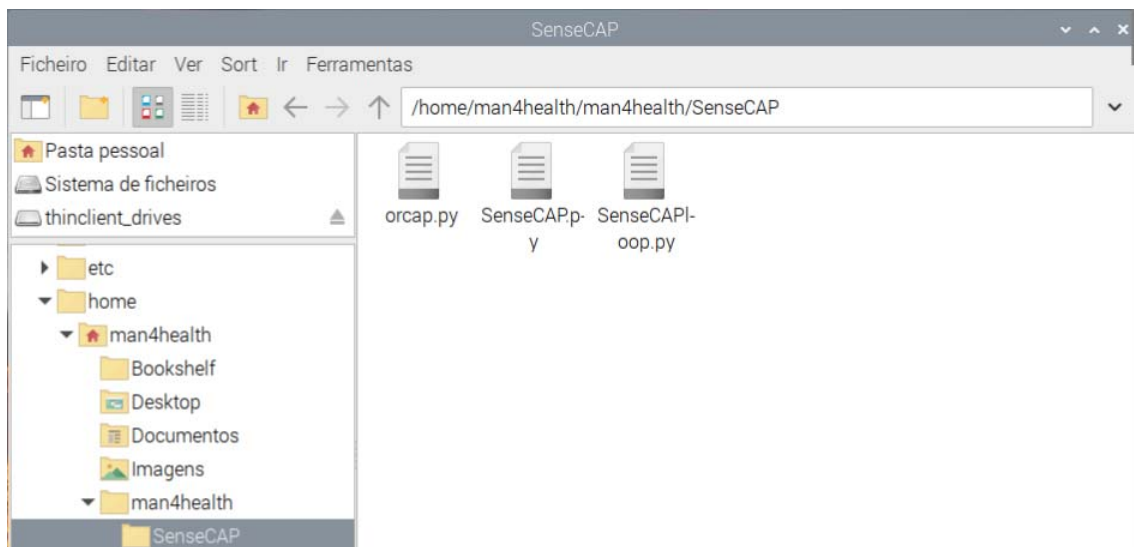
<https://pimylifeup.com/raspberry-pi-remote-desktop/>

Os scripts python que se seguem foram executados tendo em consideração que a versão instalada é a 3.9.2.



Figura 22 - Comando para avaliar a versão do python instalado no Raspberry Pi.

Criar a seguinte árvore de diretórios:



Dentro da pasta SenseCAP criar um arquivo com nome SenseCAP.py:



Figura 23 - Arquivo SenseCAP.py que deve ser criado no interior do diretório.

Com o seguinte conteúdo:

```
#!/usr/bin/python3

"""
Created on Fri Sep 30 12:43:58 2022

@author: jpcoelho
"""

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

def on_connect(client, userdata, 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"
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 = "Meteo"
deviceid = "SenseCAP01"
protocol = "json"
topic = "/" + protocol + "/" + apikey + "/" + deviceid + "/attrs"

# Configuração Modbus
PORT='/dev/ttyUSB0'
TEMP_REGISTER = 0
HUM_REGISTER = 1
BAR_REGISTER = 2
LUX_REGISTER = 4

SLAVE_ADDRESS = 14

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

# Parametros do instrumento
instrument.serial.baudrate = 9600          # Baud
instrument.serial.bytesize = 8
instrument.serial.parity = minimalmodbus.serial.PARITY_NONE
instrument.serial.stopbits = 1
instrument.serial.timeout = None           # seconds

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

# Le temperatura
airTemperature = (instrument.read_register(TEMP_REGISTER)/100.0)
# Le Humidade
airMoisture = (instrument.read_register(HUM_REGISTER)/100.0)
# Le Pressão Atmosférica
barMSB= instrument.read_register(BAR_REGISTER)
barLSB= instrument.read_register(BAR_REGISTER+1)
atmosphericPressure= round(((barMSB<16)+barLSB))*0.01
# Le intensidade de radiação solar
luxMSB= instrument.read_register(LUX_REGISTER)
luxLSB= instrument.read_register(LUX_REGISTER+1)
solarIlluminance = round(((luxMSB<16)+barLSB)*0.01,2)
solarIrradiance = round(solarIlluminance * 0.008197,2)
# Mostra valores (debug apenas)
print('The Air Temperature is: %.1f °C\r' % airTemperature)
print('The Air Humidity is: %.1f percent\r' % airMoisture)
print('The Atmospheric Pressure is: %.1f Pa\r' % atmosphericPressure)
print('The Light Intensity is: %.1f lux\r' % solarIlluminance)
print('The Solar Irradiance is: %.1f W/m2\r' % solarIrradiance)

# Criar payload json

payload = {"AT": airTemperature,
           "AM": airMoisture,
           "AP": atmosphericPressure,
           "SI": solarIlluminance,
           "SR": solarIrradiance }

```

```

message = json.dumps(payload)

print(message)

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

    client.loop_start()

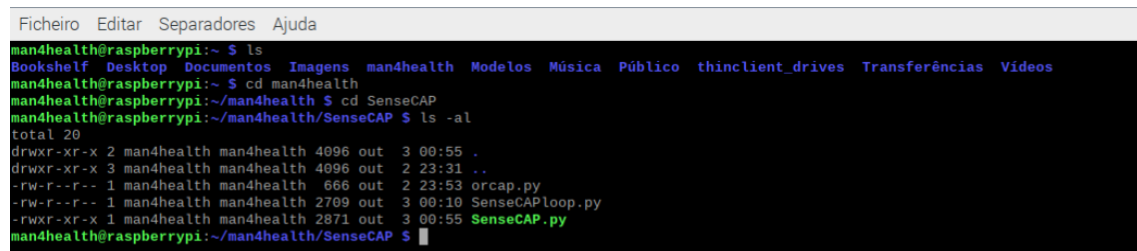
    while connected != True:
        time.sleep(0.2)

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

```

tornar o script executável com:

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



The screenshot shows a terminal window with a menu bar (Ficheiro, Editar, Separadores, Ajuda) and a title bar (man4health@raspberrypi:~). The terminal output shows the user navigating through directories and listing files with permissions:

```

man4health@raspberrypi:~ $ ls
Bookshelf Desktop Documentos Imagens man4health Modelos Música Público thinclient_drives Transferências Videos
man4health@raspberrypi:~ $ cd man4health
man4health@raspberrypi:~/man4health $ cd SenseCAP
man4health@raspberrypi:~/man4health/SenseCAP $ ls -al
total 20
drwxr-xr-x 2 man4health man4health 4096 out  3 00:55 .
drwxr-xr-x 3 man4health man4health 4096 out  2 23:31 ..
-rw-r--r-- 1 man4health man4health  666 out  2 23:53 orcap.py
-rw-r--r-- 1 man4health man4health 2709 out  3 00:10 SenseCAPloop.py
-rwxr-xr-x 1 man4health man4health 2871 out  3 00:55 SenseCAP.py
man4health@raspberrypi:~/man4health/SenseCAP $

```

5.2.1. Executar o script periodicamente

<https://www.geeksforgeeks.org/python-script-that-is-executed-every-5-minutes/>

```
$ crontab -e
```

```
*/1 * * * * /home/man4health/man4health/SenseCAP/SenseCAP.py
```

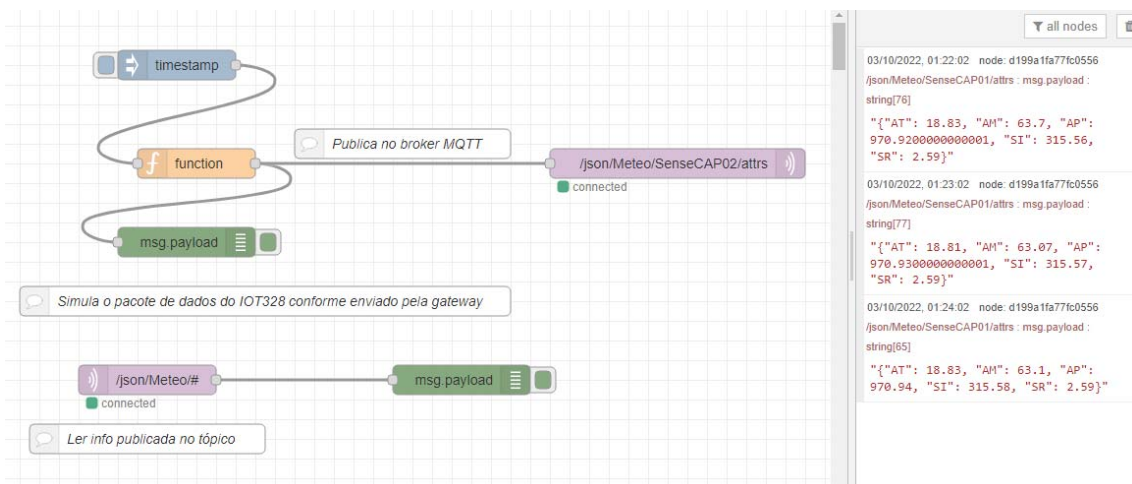


```

Ficheiro  Editar  Separadores  Ajuda
GNU nano 5.4 /tmp/crontab.1WC5yq/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
*/1 * * * * /home/man4health/man4health/SenseCAP/SenseCAP.py

```

5.2.2. Flow no Node-Red para observar as publicações:



```

[{"id":"329b4ac7b82e5d2b","type":"tab","label":"Flow
8","disabled":false,"info":"","props":[{"p":"payload"},{"p":"topic","vt":"str"}],"repeat":"","crontab
":"","once":false,"onceDelay":0.1,"topic":"","payloadType":"date","x":180,"y":60,"wires"
:[["de83dc47b61cdfc3"]]}, {"id":"de83dc47b61cdfc3","type":"function","z":"329b4ac7b82e5d2
b","name":"","func":"var gateway_payload={};\nvar airMoisture =
Math.round(100*(0.5+0.4*Math.random()));\nvar airTemperature =
Math.round(22+0.4*Math.random());\nvar atmosphericPressure =
Math.round(950+10*Math.random());\nvar solarIlluminance =
Math.round(30000*Math.random());\nvar solarIrradiance = Math.round(solarIlluminance *
0.0085);\n\nmsg.payload={\n  \"AT\": airTemperature,\n  \"AM\": airMoisture,\n
  \"AP\": atmosphericPressure,\n  \"SI\": solarIlluminance,\n  \"SR\":
solarIrradiance\n};\n\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":200,"y":160,"wir
es":[[["ce424fa408ae9a0d","5f73a686ec0e8c66"]]}, {"id":"ce424fa408ae9a0d","type":"debug","
z":"329b4ac7b82e5d2b","name":"","active":true,"tosidebar":true,"console":false,"tostatus
":false,"complete":"payload","targetType":"msg","statusVal":"","statusType":"auto","x":1
90,"y":240,"wires":[]]}, {"id":"5b3cc6e2d2a16f7c","type":"comment","z":"329b4ac7b82e5d2b",
"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":270,"y":300,"wires":[]},{ "id":"5f73a686ec0e8c66","type":"mqtt
out","z":"329b4ac7b82e5d2b","name":"","topic":"/json/Meteo/SenseCAP02/attrs","qos":"2","
retain":"","respTopic":"","contentType":"","userProps":"","correl":"","expiry":"","broke
r":"a7f68a32698ddb0f","x":690,"y":160,"wires":[]},{ "id":"5348075958ac53cc","type":"mqtt
in","z":"329b4ac7b82e5d2b","name":"","topic":"/json/Meteo/#","qos":"2","datatype":"auto"
,"broker":"a7f68a32698ddb0f","nl":false,"rap":true,"rh":0,"x":150,"y":380,"wires":[["d19
9a1fa77fc0556"]]}, {"id":"d199a1fa77fc0556","type":"debug","z":"329b4ac7b82e5d2b","name":
":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","s
tatusVal":"","statusType":"auto","x":470,"y":380,"wires":[]},{ "id":"7616d8ad11511adf","t
ype":"comment","z":"329b4ac7b82e5d2b","name":"Ler info publicada no
tópico","info":"","x":150,"y":440,"wires":[]},{ "id":"208afa832fd4e54b","type":"comment",
"z":"329b4ac7b82e5d2b","name":"Publica no broker
MQTT","info":"","x":410,"y":140,"wires":[]},{ "id":"a7f68a32698ddb0f","type":"mqtt-
broker","name":"man4health","broker":"193.136.195.25","port":"1883","clientId":"NODE_RED
_Client","usetls":false,"protocolVersion":"4","keepalive":"60","cleansession":true,"birt
hTopic":"","birthQos":"0","birthPayload":"","birthMsg":{},"closeTopic":"","closeQos":"0"
,"closePayload":"","closeMsg":{},"willTopic":"","willQos":"0","willPayload":"","willMsg"
:{}, "sessionExpiry":""}]}
```

5.2.3. Executar o script pelo startup

<https://www.instructables.com/Raspberry-Pi-Launch-Python-script-on-startup/>

5.2.4. Executar em Loop

SenseCAPloop.py

```
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 30 12:43:58 2022

@author: jpcoelho
"""

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

def on_connect(client, userdata, 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"
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 = "Meteo"
deviceid = "SenseCAP01"
protocol = "json"
topic = "/" + protocol + "/" + apikey + "/" + deviceid + "/attrs"

# Configuração Modbus
```

```

PORT='/dev/ttyUSB0'
TEMP_REGISTER = 0
HUM_REGISTER = 1
BAR_REGISTER = 2
LUX_REGISTER = 4

SLAVE_ADDRESS = 14

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

# Parametros do instrumento
instrument.serial.baudrate = 9600          # Baud
instrument.serial.bytesize = 8
instrument.serial.parity = minimalmodbus.serial.PARITY_NONE
instrument.serial.stopbits = 1
instrument.serial.timeout = None           # seconds

# Fecha porto
instrument.close_port_after_each_call = True
instrument.clear_buffers_before_each_transaction = True
while(True):
    # Le temperatura
    airTemperature = (instrument.read_register(TEMP_REGISTER)/100.0)
    # Le Humidade
    airMoisture = (instrument.read_register(HUM_REGISTER)/100.0)
    # Le Pressão Atmosférica
    barMSB= instrument.read_register(BAR_REGISTER)
    barLSB= instrument.read_register(BAR_REGISTER+1)
    atmosphericPressure= (((barMSB<<16)+barLSB)*0.01)
    # Le intensidade de radiação solar
    luxMSB= instrument.read_register(LUX_REGISTER)
    luxLSB= instrument.read_register(LUX_REGISTER+1)
    solarIlluminance = ((luxMSB<<16)+barLSB)*0.01
    # Mostra valores (debug apenas)
    print('The Air Temperature is: %.1f °C\r' % airTemperature)
    print('The Air Humidity is: %.1f percent\r' % airMoisture)
    print('The Atmospheric Pressure is: %.1f Pa\r' % atmosphericPressure)
    print('The Light Intensity is: %.1f lux\r' % solarIlluminance)

    # Criar payload json

    payload = {"AT": airTemperature,
               "AM": airMoisture,
               "AP": atmosphericPressure,
               "SI": solarIlluminance,
               "SR": solarIlluminance }

    message = json.dumps(payload)

    print(message)

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

        client.loop_start()

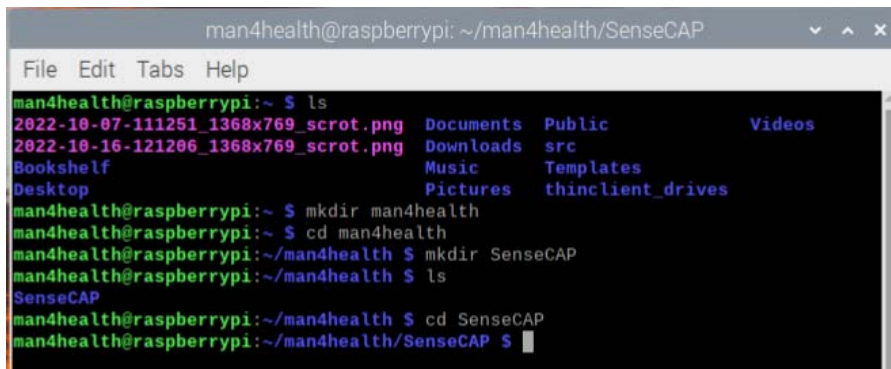
        while connected != True:
            time.sleep(0.2)

        client.publish(topic,message)
        client.loop_stop()
    except:
        print("Impossível conetar ao broker MQTT ")
        time.sleep(2)

```

5.3. Raspberry Pi modelo 1 B+

Depois da instalação do ambiente de acesso remoto e dos drivers para a placa WiFi (ver documento/capítulo correspondente), vai-se instalar o script Python para publicação dos dados. Começa-se por se criar a árvore de diretórios

A terminal window titled 'man4health@raspberrypi: ~/man4health/SenseCAP' showing the following commands and output:

```
man4health@raspberrypi:~ $ ls
2022-10-07-111251_1368x769_scrot.png  Documents  Public      Videos
2022-10-16-121206_1368x769_scrot.png  Downloads  src
Bookshelf                             Music      Templates
Desktop                               Pictures   thinclient_drives

man4health@raspberrypi:~ $ mkdir man4health
man4health@raspberrypi:~ $ cd man4health
man4health@raspberrypi:~/man4health $ mkdir SenseCAP
man4health@raspberrypi:~/man4health $ ls
SenseCAP
man4health@raspberrypi:~/man4health $ cd SenseCAP
man4health@raspberrypi:~/man4health/SenseCAP $
```

Figura 24 - Estrutura de diretórios onde será disponibilizado o script python.

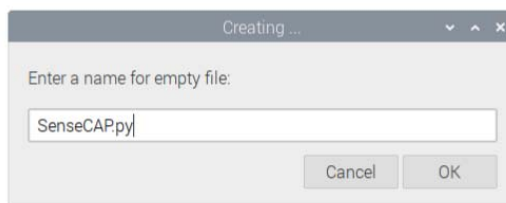
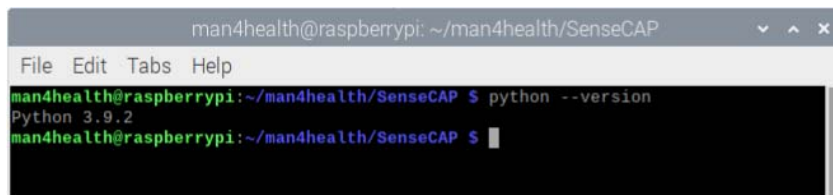


Figura 25 - Criação do ficheiro SenseCAP.py

A terminal window titled 'man4health@raspberrypi: ~/man4health/SenseCAP' showing the command 'python --version' and its output:

```
man4health@raspberrypi:~/man4health/SenseCAP $ python --version
Python 3.9.2
man4health@raspberrypi:~/man4health/SenseCAP $
```

Figura 26 - Versão do Python instalado.

```
$ sudo pip install pyserial
$ sudo pip3 install -U minimalmodbus
$ sudo pip install paho-mqtt
```

```
man4health@raspberrypi: ~/man4health/SenseCAP
File Edit Tabs Help
man4health@raspberrypi:~/man4health/SenseCAP $ sudo pip install pyserial
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: pyserial in /usr/lib/python3/dist-packages (3.5b0)
man4health@raspberrypi:~/man4health/SenseCAP $ sudo pip3 install -U minimalmodbus
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting minimalmodbus
  Downloading https://www.piwheels.org/simple/minimalmodbus/minimalmodbus-2.0.1-py3-none-any.whl (33 kB)
Requirement already satisfied: pyserial>=3.0 in /usr/lib/python3/dist-packages (from minimalmodbus) (3.5b0)
Installing collected packages: minimalmodbus
Successfully installed minimalmodbus-2.0.1
man4health@raspberrypi:~/man4health/SenseCAP $
```

```
man4health@raspberrypi: ~
File Edit Tabs Help
man4health@raspberrypi:~ $ sudo pip install paho-mqtt
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting paho-mqtt
  Downloading https://www.piwheels.org/simple/paho-mqtt/paho_mqtt-1.6.1-py3-none-any.whl (75 kB)
  | 75 kB 193 kB/s
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.6.1
man4health@raspberrypi:~ $
```

```
man4health@raspberrypi: ~/man4health/SenseCAP
File Edit Tabs Help
s
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting minimalmodbus
  Downloading https://www.piwheels.org/simple/minimalmodbus/minimalmodbus-2.0.1-py3-none-any.whl (33 kB)
Requirement already satisfied: pyserial>=3.0 in /usr/lib/python3/dist-packages (from minimalmodbus) (3.5b0)
Installing collected packages: minimalmodbus
Successfully installed minimalmodbus-2.0.1
man4health@raspberrypi:~/man4health/SenseCAP $ python SenseCAP1shot.py
Traceback (most recent call last):
  File "/home/man4health/man4health/SenseCAP/SenseCAP1shot.py", line 8, in <module>
    import paho.mqtt.client as mqttclient
ModuleNotFoundError: No module named 'paho'
man4health@raspberrypi:~/man4health/SenseCAP $ python SenseCAP1shot.py
The Air Temperature is: 19.5 °C
The Air Humidity is: 71.2 percent
The Atmospheric Pressure is: 963.8 Pa
The Light Intensity is: 308.4 lux
The Solar Irradiance is: 2.5 W/m2
["AT": 19.49, "AM": 71.23, "AP": 963.75, "SI": 308.39, "SR": 2.53]
Impossível conectar ao broker MQTT
man4health@raspberrypi:~/man4health/SenseCAP $
```

```
The Light Intensity is: 308.4 lux
The Solar Irradiance is: 2.5 W/m2
["AT": 19.49, "AM": 71.23, "AP": 963.75, "SI": 308.39, "SR": 2.53]
Impossível conectar ao broker MQTT
man4health@raspberrypi:~/man4health/SenseCAP $
```

tornar o script executável com:

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

```
man4health@raspberrypi: ~/man4health/SenseCAP
File Edit Tabs Help
man4health@raspberrypi:~/man4health/SenseCAP $ sudo chmod +x SenseCAP.py
man4health@raspberrypi:~/man4health/SenseCAP $ ls
SenseCAP.py
man4health@raspberrypi:~/man4health/SenseCAP $ ls -al
total 12
drwxr-xr-x 2 man4health man4health 4096 Oct 16 13:10 .
drwxr-xr-x 3 man4health man4health 4096 Oct 16 13:04 ..
-rwxr-xr-x 1 man4health man4health 2871 Oct 16 13:10 SenseCAP.py
man4health@raspberrypi:~/man4health/SenseCAP $
```

Figura 27 - Resultado após alteração de modo do ficheiro SenseCAP.py

Adicionar tarefa de execução ao scheduler:

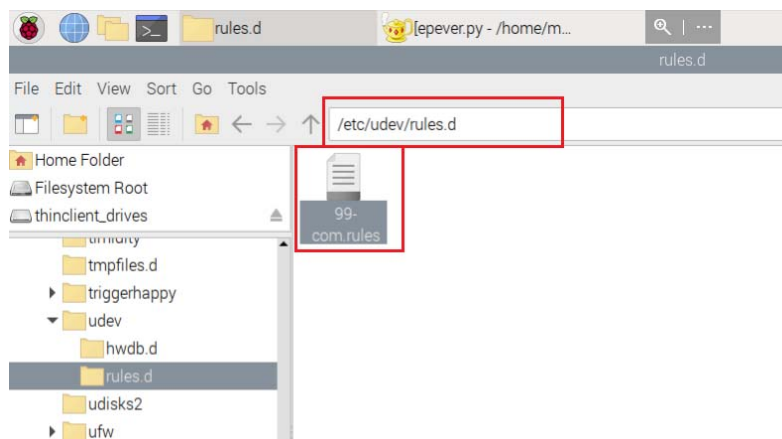
```
$ crontab -e
```

```
Ficheiro Editar Separadores Ajuda
GNU nano 5.4 /tmp/crontab.1WC5yq/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
*/1 * * * * /home/man4health/man4health/SenseCAP/SenseCAP.py
```

```
*/1 * * * * /home/man4health/man4health/SenseCAP/SenseCAP.py
```

5.4. Binding do nome da porta série ao hardware

Abrir o ficheiro **99-com.rules** que se encontra em **/etc/udev/rules.d**



Adicionar, no final do ficheiro, as seguintes ações:

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

Guardar e fazer reboot.

5.4.1. Novo script Python usando hardware binding port name

```
#!/usr/bin/python3  
  
"""  
Created on Fri Sep 30 12:43:58 2022  
  
@author: jpcoelho  
"""  
  
import minimalmodbus  
import paho.mqtt.client as mqttclient  
import time  
import json  
  
def on_connect(client, userdata, 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 = "mrmdoror.hopto.org"  
  
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 = "Meteo"  
deviceid = "SenseCAP01"  
protocol = "json"  
topic = "/" + protocol + "/" + apikey + "/" + deviceid + "/attrs"  
  
# Configuração Modbus  
PORT='/dev/SenseCAPdongle'  
TEMP_REGISTER = 0  
HUM_REGISTER = 1  
BAR_REGISTER = 2  
LUX_REGISTER = 4  
  
SLAVE_ADDRESS = 14  
  
# Configura instrumento  
instrument =  
minimalmodbus.Instrument(PORT,SLAVE_ADDRESS,mode=minimalmodbus.MODE_RTU)  
  
# Parametros do instrumento
```

```

instrument.serial.baudrate = 9600          # Baud
instrument.serial.bytesize = 8
instrument.serial.parity   = minimalmodbus.serial.PARITY_NONE
instrument.serial.stopbits = 1
instrument.serial.timeout  = None          # seconds

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

# Le temperatura
airTemperature = (instrument.read_register(TEMP_REGISTER)/100.0)
# Le Humidade
airMoisture = (instrument.read_register(HUM_REGISTER)/100.0)
# Le Pressão Atmosférica
barMSB= instrument.read_register(BAR_REGISTER)
barLSB= instrument.read_register(BAR_REGISTER+1)
atmosphericPressure= round(((barMSB<<16)+barLSB))*0.01
# Le intensidade de radiação solar
luxMSB= instrument.read_register(LUX_REGISTER)
luxLSB= instrument.read_register(LUX_REGISTER+1)
solarIlluminance = round(((luxMSB<<16)+barLSB)*0.01,2)
solarIrradiance = round(solarIlluminance * 0.008197,2)
# Mostra valores (debug apenas)
print('The Air Temperature is: %.1f °C\r' % airTemperature)
print('The Air Humidity is: %.1f percent\r' % airMoisture)
print('The Atmospheric Pressure is: %.1f Pa\r' % atmosphericPressure)
print('The Light Intensity is: %.1f lux\r' % solarIlluminance)
print('The Solar Irradiance is: %.1f W/m2\r' % solarIrradiance)

# Criar payload json
payload = {"AT": airTemperature,
           "AM": airMoisture,
           "AP": atmosphericPressure,
           "SI": solarIlluminance,
           "SR": solarIrradiance }

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 ")

```

6. Credenciais utilizadas no projeto

6.1. Login:

Gateway: root

Raspberry Pi: man4health

6.2. Password:

mAn4hEaLtH

7. Conversão entre irradiância solar e iluminância

<https://www.extrica.com/article/21667>

1000 W/m² → 122 klux (outdoor sunlight)

122 000 lux ----- 1000 W/m²

K lux ----- SR

$SR = 1000 * K / 122\ 000$