

Redes de Sensores

Implementação de uma rede de sensores

Relatório Final

Rafael Ferreira nº2172044

Samuel Heleno nº2172104

Professor Telemo Fernandes

Redes de Sensores 3ºAno, 1º Semestre

Instituto Politécnico de Leiria

9 de Fevereiro de 2021

Conteúdo

Introdução	2
1 Pycom Pysense	3
1.1 Sensores	3
1.1.1 Sensor de Temperatura (SI7006A20)	3
1.1.2 Sensor de Humidade (SI7006A20)	5
1.1.3 Sensor de Luminosidade (LTR-329ALS-02)	5
1.1.4 Sensor Barométrico de Pressão (MPL3115A2)	6
1.1.5 Sensor Acelerómetro (LIS2HH12)	6
1.2 Protocolos de Comunicação	7
1.2.1 Bluetooth Low Energy (BLE)	7
1.2.2 Wi-Fi	7
1.2.3 Message Queuing Telemetry Transport (MQTT)	7
1.2.4 LoRa	7
2 Rede de Sensores	7
2.1 MQTT e BLE	7
2.2 LoRaWAN	8
3 Bibliografia	9
4 Anexos	10

Lista de Figuras

1	Rede de Sensores	2
2	Ler Sensor de Temperatura	3
3	Ler Sensor de Temperatura com biblioteca	4
4	Ler Sensor de Humidade com biblioteca	5
5	Ler Sensor de Luminosidade com biblioteca	5
6	Ler Sensor Barométrico com biblioteca	6
7	Ler Sensor Acelerómetro com biblioteca	6
8	Dashboard Rede de Sensores MQTT e BLE	8
9	Diagrama de blocos Rede de Sensores MQTT e BLE	10
10	Diagrama de blocos Rede de Sensores LoRaWAN	10
11	Diagrama de blocos Rede de Sensores LoRaWAN - Cayenne	10

Introdução

Com o avanço que se tem verificado na tecnologia, cada vez é mais comum a sua utilização em meios domésticos, de forma a proporcionar um maior conforto nas vidas das pessoas.

Um exemplo da sua aplicação é o uso de uma rede de sensores com o intuito de monitorizar e disponibilizar, aos utilizadores, diversos parâmetros ambientais e de consumos considerados de interesse.

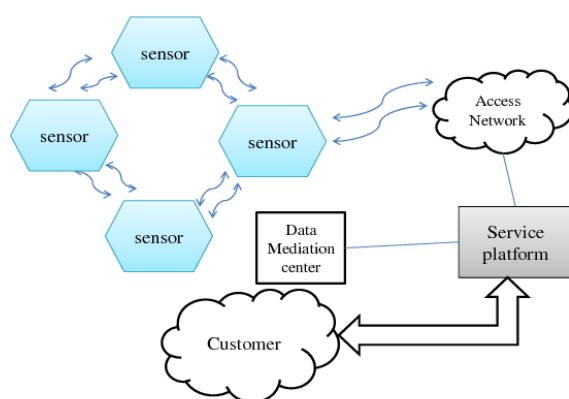


Figura 1: Rede de Sensores

As redes de sensores são um tipo de redes de comunicação, normalmente ad-hoc, que permitem a passagem de informação entre os diversos nós da rede, utilizando um determinado protocolo de rede para o encaminhamento dos dados.

Os nós da rede são compostos por dispositivos activos que recebem e/ou transmitem informações através de um canal de informação.

A comunicação na rede pode ser feita através de ligações com fios, sem fios ou com junção das duas.

1 Pycom Pysense

O PyCom e o PySense são um microcontrolador com multi-network modules e um shield sensor, respetivamente.

O PySense é composto por cinco sensores: humidade, temperatura, acelerómetro, luminosidade e pressão. A informação obtida por estes sensores pode ser obtida pelo LoPY, através do protocolo de comunicação I2C.

I2C é um protocolo de comunicação que permite conectar vários slaves a um master ou vice-versa. Este é um protocolo de comunicação série e síncrono. Uma das características do protocolo é que apenas necessita de duas linhas bidireccionais para comunicar: SCL(Serial Clock) e SDA(Serial Data).

1.1 Sensores

Neste sub-capítulo, vamos preceder à leitura dos sensores do PySene. Fizemos uso de várias bibliotecas, disponibilizadas, para efectuar a leitura dos sensores. Estas permitem-nos usufruir de todas as capacidades do sensor de uma forma simples e rápida.

1.1.1 Sensor de Temperatura (SI7006A20)

É um CMOS (semi-condutor) monolítico que integra elementos de sensores de temperatura, um ADC, processamento de sinal, dados de calibração e uma Interface com I2C. Tem uma elevada estabilidade a longo prazo.

Leitura de Sensor de Temperatura

```
1 from machine import I2C
2
3 import time
4
5 # create and init as a master and use non-default PIN assignments (P22=SDA, P21=SCL)
6 i2c = I2C(0, I2C.MASTER, pins=('P22','P21'), baudrate=100000)
7
8 #imprime lista de slave addresses
9 print(i2c.scan())
10
11 #send 2 bytes to slave with address 0x40
12 i2c.writeto(0x40, bytearray([0xF3]))
13
14 #aguarda que a operação ocorra
15 time.sleep(0.1)
16
17 # receive 2 bytes from slave with address 0x40
18 data = i2c.readfrom(0x40, 2)
19
20 datatemp = ((data[0]&0xFF)<<8) + (data[1]&0xFF)
21
22 temp = ((175.72 * datatemp) / 65536.0) - 46.85
23
24 print("Temperatura: %s" %temp)
25
```

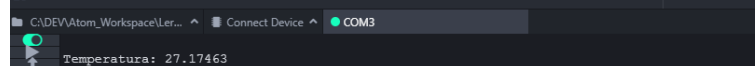


Figura 2: Ler Sensor de Temperatura

Como se pode ver na figura 2, de forma a obter a informação do sensor através do protocolo I2C é necessário fazer o import da biblioteca I2C. Definem-se os pinos das duas linhas bidireccionais (P22=SDA, P21=SCL) e define-se o baudrate com um valor aconselhado.

Define-se o endereço onde são lidos os dados de temperatura, que neste caso será o endereço do sensor 0x40. É gerado um delay para garantir que a escrita é bem-sucedida.

Posteriormente, os dados do sensor são lidos através da função `i2c.readfrom()`, onde o parâmetro de entrada é o endereço do respectivo sensor, e são devidamente acondicionados, de acordo com o datasheet, para a unidade da temperatura ser Celsius.

Leitura de Sensor de Temperatura com biblioteca

```
import time
import pycom
from pysense import Pysense
import machine

from SI7006A20 import SI7006A20

pycom.heartbeat(False)
# pycom.rgbled(0x0A0A08) # white
pycom.rgbled(0x0000DD) # white

py = Pysense()

si = SI7006A20(py)
print("Temperature: " + str(si.temperature())+ " deg C ")

time.sleep(3)
py.setup_sleep(10)
py.go_to_sleep()
```

Figura 3: Ler Sensor de Temperatura com biblioteca

1.1.2 Sensor de Humidade (SI7006A20)

O sensor de humidade é o mesmo sensor utilizado na medição da temperatura, sendo que a sua implementação em python e o processo de recolha de dados é semelhante à do sensor anterior.

Leitura de Sensor de Humidade com biblioteca

```
import time
import pycom
from pysense import Pysense
import machine

from SI7006A20 import SI7006A20

pycom.heartbeat(False)
# pycom.rgbled(0x0A0A0A) # white
pycom.rgbled(0x000000) # white

py = Pysense()

si = SI7006A20(py)
print("Relative Humidity: " + str(si.humidity()) + " %RH")
t_ambient = 24.4
print("Humidity Ambient for " + str(t_ambient) + " deg C is " + str(si.humid_ambient(t_ambient)) + " %RH")

time.sleep(3)
py.setup_sleep(10)
py.go_to_sleep()
```

Figura 4: Ler Sensor de Humidade com biblioteca

1.1.3 Sensor de Luminosidade (LTR-329ALS-02)

Converte a intensidade da luz, num sinal de saída digital com interface direta através de I2C. Fornece uma resposta linear entre uma faixa dinâmica de 0.01 lux a 64k lux e deve ser utilizado em ambientes com alta incidência luminosa.

Leitura de Sensor de Luminosidade com biblioteca

```
import time
import pycom
from pysense import Pysense
import machine

from LTR329ALS01 import LTR329ALS01

pycom.heartbeat(False)
# pycom.rgbled(0x0A0A0A) # white
pycom.rgbled(0x000000) # white

py = Pysense()

lt = LTR329ALS01(py)
print("Light (channel Blue lux, channel Red lux): " + str(lt.light()))

time.sleep(3)
py.setup_sleep(10)
py.go_to_sleep()
```

Figura 5: Ler Sensor de Luminosidade com biblioteca

1.1.4 Sensor Barométrico de Pressão (MPL3115A2)

Mede a pressão atmosférica e possui uma ampla faixa de operação de 20 kPa até 110kPa, uma faixa que cobre todas as elevações da superfície da terra. Os dados de pressão e temperatura são convertidos através de um ADC de alta resolução para fornecer saídas de pressão em Pascal(Pa).

Leitura de Sensor Barométrico de Pressão com biblioteca

```
import time
import pycom
from pysense import Pysense
import machine

from MPL3115A2 import MPL3115A2, ALTITUDE, PRESSURE

pycom.heartbeat(False)
# pycom.rgbled(0x00A0A0) # white
pycom.rgbled(0x000000) # white

py = Pysense()

mpp = MPL3115A2(py, mode=PRESSURE) # Returns pressure in Pa. Mode may also be set to ALTITUDE, returning a value in meters
print("Pressure: " + str(mpp.pressure()))

time.sleep(3)
py.setup_sleep(10)
py.go_to_sleep()
```

Figura 6: Ler Sensor Barométrico com biblioteca

1.1.5 Sensor Acelerômetro (LIS2HH12)

O LIS2HH12 é um acelerômetro linear de três eixos x,y e z, possui um alto desempenho, opera a ultra baixa potência entre temperaturas de -45 a +85 °C e utiliza algoritmia do tipo FIFO.

Possui escalas completas de 2/4/8g de aceleração e é capaz de medir acelerações de saída com taxas de dados de na gama de 10 Hz a 800 Hz de frequência.

Leitura de Sensor Acelerômetro com biblioteca

```
import time
import pycom
from pysense import Pysense
import machine

from LIS2HH12 import LIS2HH12

pycom.heartbeat(False)
# pycom.rgbled(0x00A0A0) # white
pycom.rgbled(0x000000) # white

py = Pysense()

li = LIS2HH12(py)
print("Acceleration: " + str(li.acceleration()))

time.sleep(3)
py.setup_sleep(10)
py.go_to_sleep()
```

Figura 7: Ler Sensor Acelerômetro com biblioteca

1.2 Protocolos de Comunicação

O LoPy possui diversos protocolos de comunicação que permitem comunicações sem fios a elevadas frequências, típicas duma rede de sensores. A vantagem deste tipo de comunicação é a utilização das bandas livres (ISM). Apesar de terem um tempo de utilização limitado, não consiste num problema porque o tempo de comunicação numa rede deste tipo é mínimo. Os protocolos utilizados, neste projeto, são:

1.2.1 Bluetooth Low Energy (BLE)

Protocolo útil para comunicações de curta distância e de baixo débito. Oferece baixo consumo e uma elevada segurança.

1.2.2 Wi-Fi

Protocolo ideal para comunicações de longa distância. Permite transmissão de alto débito mas apresenta uma baixa segurança de comunicação.

1.2.3 Message Queuing Telemetry Transport (MQTT)

Protocolo útil para comunicações de longa e média distância. Indicado para comunicações de baixo débito, a partir do Wi-Fi. A comunicação é feita a partir de um broker, que permite bidirecionalidade. A sua utilização em redes de sensores é muito aconselhável.

1.2.4 LoRa

Protocolo útil para comunicações de média e longa distância, com baixo débito. Permite fazer o balanço entre o débito e o alcance, através do spreadfactor. Pela sua versatilidade, por ser economicamente muito acessível e oferecer um baixo consumo energético, este protocolo é o mais utilizado em redes de sensores.

2 Rede de Sensores

De forma a implementar uma rede de sensores para uma habitação, que monitorize diversos parâmetros ambientais e consumos considerados de interesse foram utilizadas diversas técnicas. Estes parâmetros são disponibilizados, para consulta do utilizador, numa interface de web.

2.1 MQTT e BLE

Neste caso foi implementado uma rede de sensores, demonstrada na figura 9, que permite medir diversos parâmetros e posteriormente, disponibiliza-los num servidor MQTT (Adafruit IO) através do broker "adafruit.com". Esta medição pode ser controlada a partir do Adafruit IO, o que significa que a comunicação é bi-direcional.

Através da figura 8, é possível verificar os feeds presentes no dashboard do Adafruit IO, onde serão publicados os valores medidos pelos sensores. Quando o interruptor, presente no dashboard, está 'ON' as medições são efetuadas periodicamente e publicadas. Esta periodicidade deve-se à gestão de utilização da comunicação, tanto para poupar a bateria do microcontrolador como para cumprir os limites de utilização da banda livre (ISM). Caso o interruptor esteja 'OFF', as medições não são efetuadas e os feeds são limpos. E o LoPy é colocado no estado idle, de forma a poupar a bateria. O interruptor pode ser comutado de três maneiras distintas: através do hardware (botão do LoPy), através de um dispositivo móvel (BLE) ou através do Adafruit IO, como mencionada anteriormente. A comutação do interruptor através do Adafruit IO é possível devido à subscrição do feed, a partir do LoPy, que permite receber o seu estado.

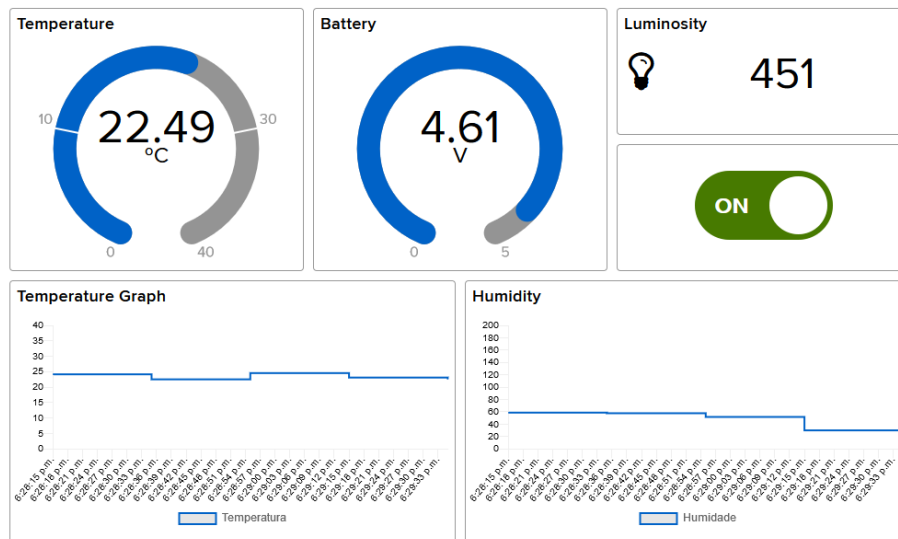


Figura 8: Dashboard Rede de Sensores MQTT e BLE

2.2 LoRaWAN

Neste caso foi implementado uma rede de sensores, demonstrada na figura 10, que permite medir diversos parâmetros e, posteriormente disponibiliza-los num servidor Adafruit IO através de uma rede LoRaWAN, utilizando o protocolo Cayenne LPP.

Esta rede consiste num LoPy que se conecta a um gateway LoRaWAN, através de comunicação LoRa. Este gateway, estabelece ligação a um servidor TTN utilizando conexões IP padrão.

O TTN (The Things Network) permite-nos criar uma aplicação, onde associamos o nosso LoPy usando o device EUI (Extend Unique Identifier). Consequentemente, é gerada uma key que nos permitem ligar à rede usando OTAA (Over the Air Activation).

De modo a publicar os valores obtidos pelos sensores no Adafruit IO, foi utilizada a integração IFTT (If This Then That). O IFTT permite criar um Applet, que serve para conectar duas ou mais aplicações ou dispositivos. Neste caso, foram criados Applets utilizando o Webhooks e o Adafruit. Como a key do Webhooks foi associada ao TTN, cada vez que os dados são recebidos pela aplicação, é gerado um trigger que induz na publicação de um valor, presente no payload, num feed do Adafruit IO.

Uma outra forma de publicar os dados num servidor, seria através da integração MyDevices configurada com o ProcessID e usando uma Access Key (default), para ser usada no downlink. A informação seria publicada num dashboard do Cayenne MyDevices, através do Cayenne LPP.

Também foi utilizada a integração HTTP, que permite uma comunicação bidirecional, ou seja, permite enviar dados para um endpoint, bem como receber dados. Ao adicionar esta integração, os dados fornecidos à aplicação vão ser enviados para o url associado (uplink url). A integração HTTP gera um downlink url, que permite enviar os dados de volta para o LoPy utilizando o curl. Porém, o downlink só sucede na próxima vez que o nó da rede mandar informação.

3 Bibliografia

<https://docs.pycom.io>
<https://github.com/pycom/pycom-libraries>
<https://io.adafruit.com/>
<https://io.adafruit.com/api/docs/>
<https://cayenne.mydevices.com/>
<https://console.thethingsnetwork.org/>
<https://ifttt.com/home>
Slides das Aulas Teóricas

4 Anexos

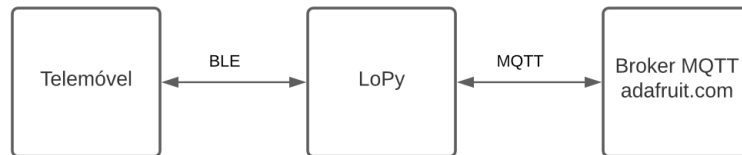


Figura 9: Diagrama de blocos Rede de Sensores MQTT e BLE

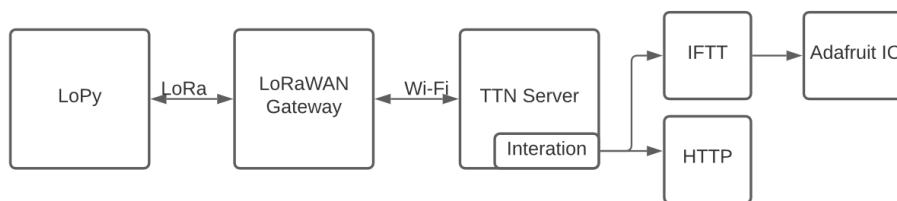


Figura 10: Diagrama de blocos Rede de Sensores LoRaWAN

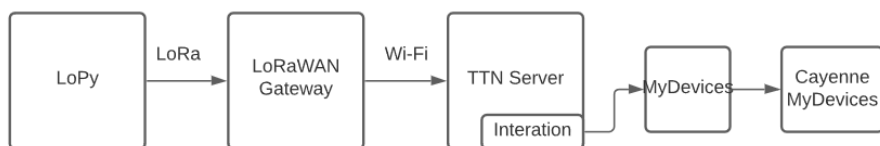


Figura 11: Diagrama de blocos Rede de Sensores LoRaWAN - Cayenne