Internet of Things

Challenge n. 2: Theoretical exercise

Andrea Caravano, Alberto Cantele

Academic Year 2024–25

Contents

1 Exercise text

A wireless IoT network consists of the following devices:

- A battery-powered, Wi-Fi-enabled **temperature sensor** that measures and transmits temperature data every 5 minutes.
- A battery-powered, Wi-Fi-enabled **valve** that receives temperature readings from the sensor and computes the average temperature every 30 minutes to decide whether to open or close.
- **A Raspberry Pi**, connected to the power grid, which only supports MQTT for communication.

The temperature sensor and valve can communicate using either **MQTT** or **CoAP**, with a specific predefined topic or resource. The topic/resource length is 10 bytes and the payload size is 8 bytes. However, since the Raspberry Pi only supports MQTT, any interaction between the Raspberry Pi and the battery-operated devices must use MQTT. The sensor and valve, however, can communicate directly using CoAP if desired.
Consider the following message sizes (in bytes), which **already include** header and payload size for the COAP resource or MQTT topic used in the system:

CoAP		MQTT	
GET Request	60 B	Subscribe	58 B
GET Response	55 B	Sub Ack	52 B
PUT Request	77 B	Publish	68 B
PUT Response	58 B	Pub Ack	51 B
Empty ACK	14 B	Connect	54 B
		Connect ACK	47 B
		Ping Request	52 B
		Ping Response	48 B

Assuming that:

1. Transmit and Receive cost per bit are:
   $E_{TX} = 50$ **nJ/bit,** $E_{RX} = 58$ **nJ/bit (nanojoule per bit)**
2. The Wi-Fi network is ideal **(no losses)**
3. The processing cost on the valve to compute the average temperature every 30 minutes is $E_c = 2.4$ **mJ (millijoule)**
4. The sensor and valve start in power-off state

Exercise Question 1 (EQ1): Compute the total energy consumed by the two battery-powered devices **over a period of 24 hours** in both cases when using COAP **(a)** and MQTT **(b)**, using each in its **most efficient configuration energy-wise**.

Exercise Question 2 (EQ2): Propose **at least one** solution for decreasing the energy consumption when passing using the Raspberry PI as a broker. **Give a rough estimate of the energy saving** that could be obtained with your solution: recompute the energy under your proposed configuration.

Let's tackle each point singularly.

2 A 24-hour complete cycle, from power-off

2.1 CoAP

2.1.1 The most efficient configuration energy-wise

Of course, in this point of the exercise, we are assuming to follow slavishly all the details of the protocol definition.

The most reasonably energy-efficient configuration that uses CoAP as an application-layer communication protocol is the one mimicking a Publish-Subscribe communication style, through the observe pattern.
For resources that allow observation, a CoAP client just needs to specify an additional option to the GET REQUEST message, registering for reception of all the updates to the interested resource.
The next point to consider is CONFIRMABLE or NON-CONFIRMABLE messages: the first kind provide stronger guarantees, being them shifted to the application layer for CoAP, but does not perform well, energy-wise.
Instead, NON-CONFIRMABLE messages only require an initial request, serving as a registration for the observable object. Then, any successive value will come as a RESPONSE, without requiring an ACK from the Client.
A comparison among the two is shown in the following test environment.

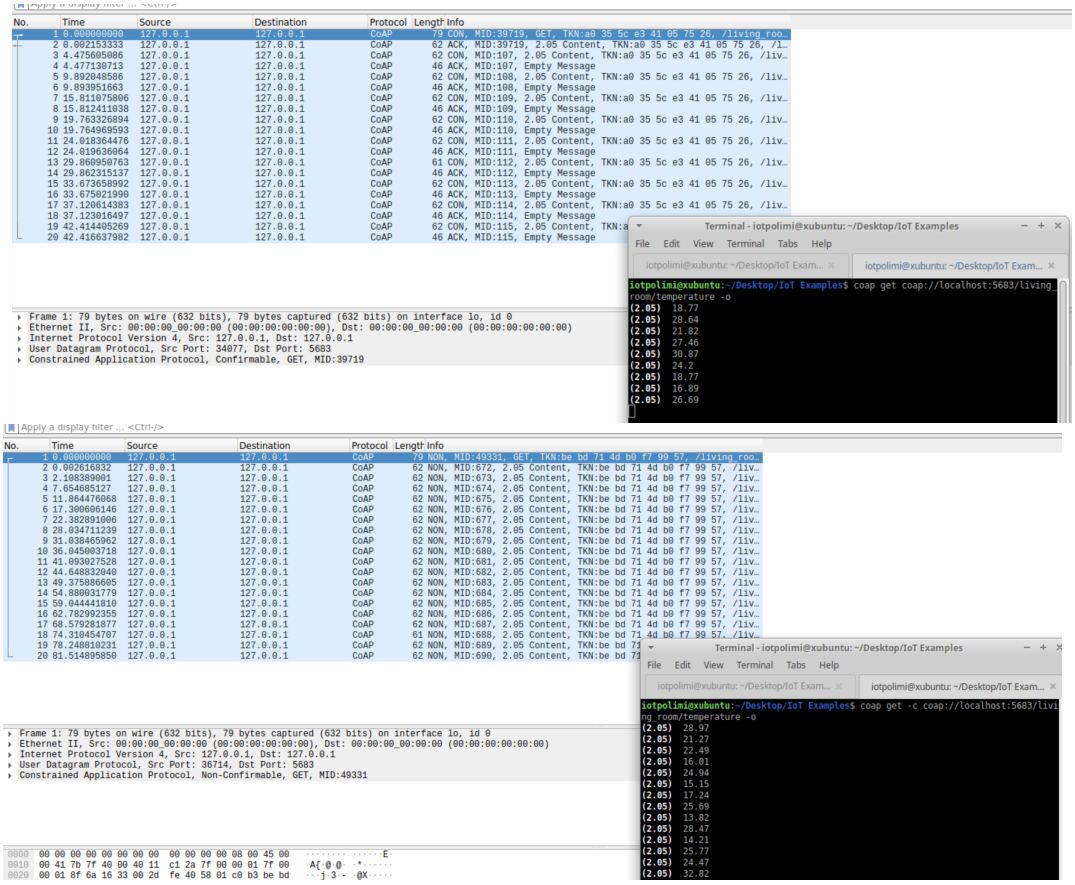
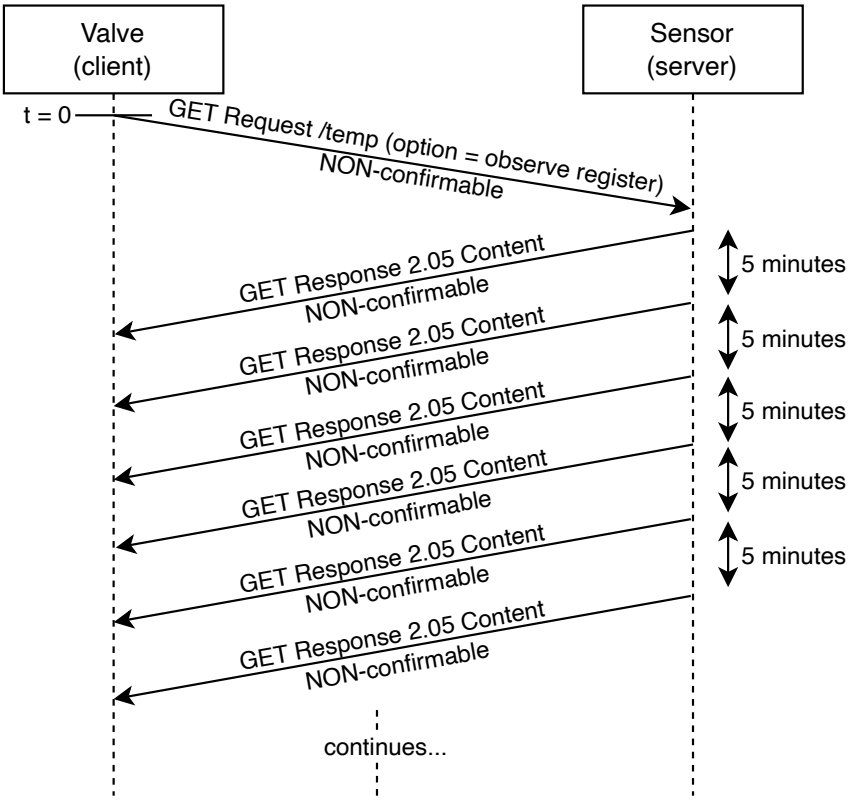


Figure 1: Comparison: CONFIRMABLE messages versus NON-CONFIRMABLE messages

2.1.2 Communication diagram

The resulting communication diagram is shown in the following. Please note that the propagation time, in this case, is not interesting during the calculation steps.



2.1.3 Solution scheme

Data declaration:

Given the message sizes as part of the text,

Energy spent for transmission (E_{TX}) = 50 nJ/bit,

Energy spent for reception (E_{RX}) = 58 nJ/bit,

Energy spent for average computation at the valve (E_c) = 2,4 mJ,

Being $L_{GET\ Request}$ and $L_{GET\ Response}$ the message sizes for the GET Request (including the option to register for observation) and GET Response packets (carrying the temperature data), respectively.

We are able to compute the number of bytes transmitted and received by each battery-operated node and therefore the energy consumed, overall.

2.1.3.1 Sensor

$$L_{SENSOR,TX} = \frac{24 \cdot 60\ minutes}{5\ minutes} \cdot L_{GET\ Response} = 288 \cdot 55 \text{ bytes} = 15840 \text{ bytes}$$

With the fraction representing the frequency of temperature updates sent via a GET Response (every 5 minutes, 12 per hour).

$L_{SENSOR,RX} = 1 \cdot L_{GET\ Request} = 60$ bytes

2.1.3.2 Valve

$L_{VALVE,TX} = 1 \cdot L_{GET\ Request} = 60$ bytes

$$L_{VALVE,RX} = \frac{24 \cdot 60\ minutes}{5\ minutes} \cdot L_{GET\ Response} = 288 \cdot 55 \text{ bytes} = 15840 \text{ bytes}$$

2.1.3.3 Energy computation

Let's define

$$E_{TX} = \frac{50\ nJ}{1\ bit} = \frac{400\ nJ}{1\ byte = 1\ bit \cdot 8}$$

$$E_{RX} = \frac{58\ nJ}{1\ bit} = \frac{464\ nJ}{1\ byte = 1\ bit \cdot 8}$$

$$E_{c\ per\ day} = 2{,}4\ mJ \cdot \frac{24 \cdot 60\ minutes}{30\ minutes} = 115{,}200\ \text{mJ}$$

Being a byte composed by 8 bits

Then, we have that, finally:

$E_{SENSOR} = E_{TX} \cdot L_{SENSOR,TX} + E_{RX} \cdot L_{SENSOR,RX} = 15840\ bytes \cdot 400\ nJ/byte + 60\ bytes \cdot 464\ nJ/byte = 6{,}336\ mJ\ + 27{,}840\ \mu J = 6{,}364\ mJ$

$E_{VALVE} = E_{TX} \cdot L_{VALVE,TX} + E_{RX} \cdot L_{VALVE,RX} + E_{c\ per\ day} = 60\ bytes \cdot 400\ nJ/byte + 15840\ bytes \cdot 464\ nJ/byte + 115{,}200\ mJ = 24{,}000\ \mu J + 7{,}350\ mJ + 115{,}200\ mJ = 122{,}574\ mJ$

$E_{TOTAL} = E_{SENSOR} + E_{VALVE} = 6{,}364\ mJ + 122{,}574\ mJ = 128{,}938\ mJ$

2.2 MQTT

2.2.1 The most efficient configuration energy-wise

Of course, in this point of the exercise, we are assuming to follow slavishly all the details of the protocol definition.
The most relevant energy-consuming element of the protocol can definitely be located in the overhead coming from both reliability at transport and application layer.
The transport protocol (TCP) is not something which can be moved away from the architecture, but the Quality of Service level at the application layer can instead be minimized, therefore minimizing energy consumption.
This behaviour can be achieved by using the QoS level 0.
Note, however, that, independently of the chosen reliability level, a CONNECT message will have its matching CONNECT ACK and each SUBSCRIBE will have its own SUBSCRIBE ACK.
Other kinds of messages require an acknowledgement from the broker, like the UNSUBSCRIBE one, reporting back with the status code for each subscription.
In our case, these last are not interesting.
Let's now simulate the system publishing a few simulated temperature measurements.

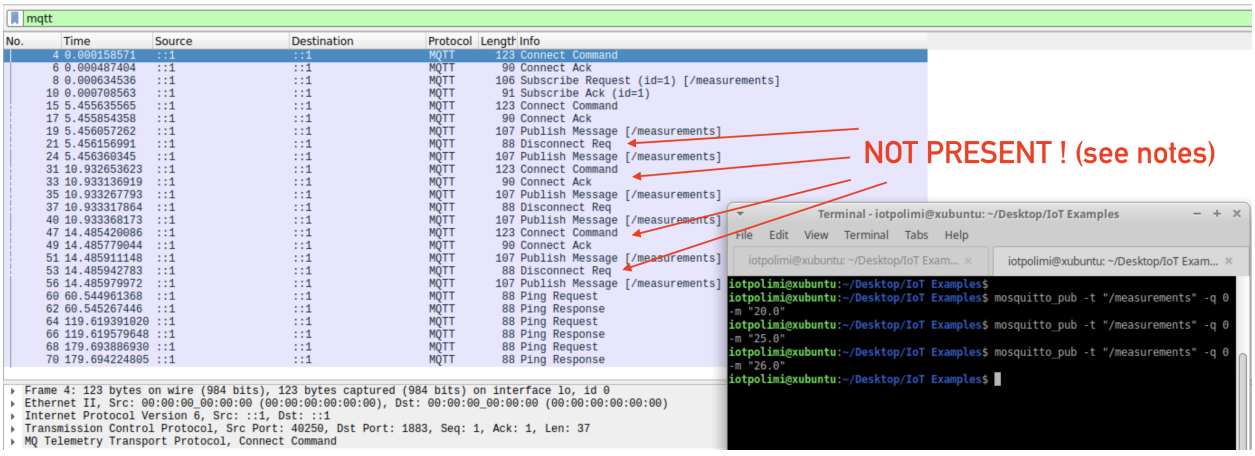


Figure 2: A simulation of the system

In our description, we have still not mentioned the liveness properties of the system: both subscribers and publishers keep an active session with the broker for the whole time? Moreover, are there any liveness timeouts (keep-alives)?

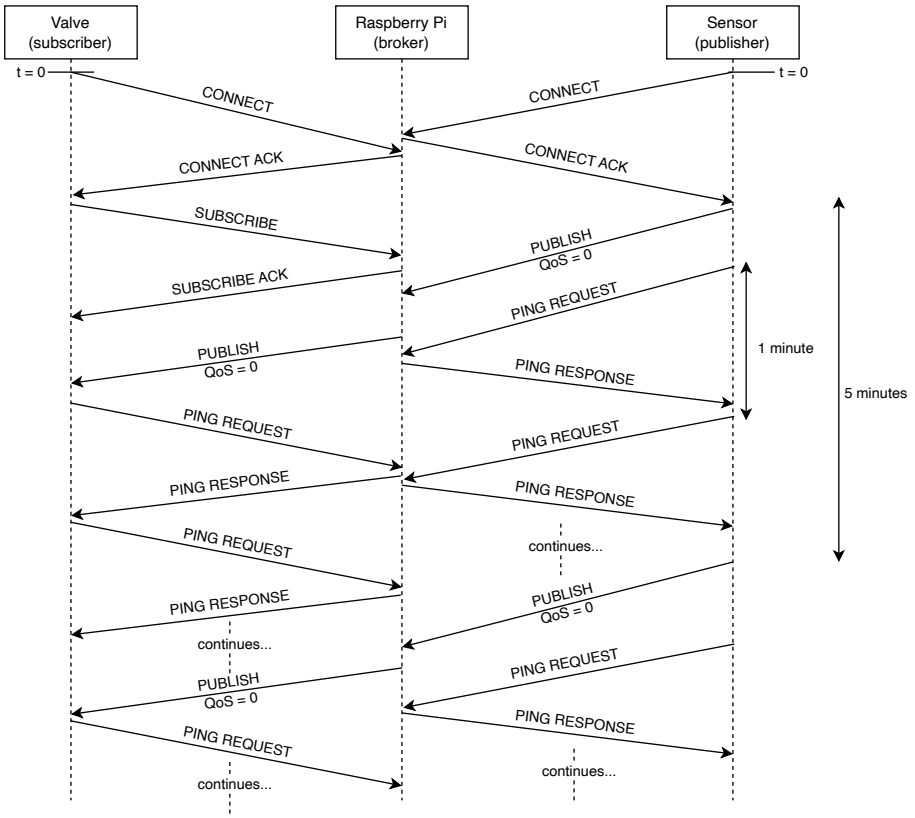
To address the first one, it is important to note that each choice is implementation dependent and no actual protocol requirements impose a certain session duration limit.
However, since we are operating in the field of IoT and no other MQTT messages are expected other than the periodic PUBLISH updates, a reasonable choice is to **keep the session open while the device is on**.
Closing it at each PUBLISH message from the sensor side, in fact, requires, from our exterior understanding of the system, more energy than persisting a state of the session in the internal device memory (3 more messages would be required at every update of the temperature: a DISCONNECT message and a pair of CONNECT and CONNECT ACK being respectfully sent and received).

No processing power is needed, instead, while data transmission/reception is not ongoing.

A special exception are liveness timeouts, that instead requires radio operation and are, in fact, added to the final energy term: they are implemented through keep-alives, being composed of the pair of PING REQUEST and PING RESPONSE messages.
How frequent is the pair of PING REQUEST and PING RESPONSE messages? Again, the protocol does not impose a strict upper or lower bound to the frequency of keep-alives, which are instead left to the implementation.
Since, for our laboratory activities, MOSQUITTO is the **reference implementation, the assumed default for liveness timeouts is 60 seconds (1 minute)**.
A very reasonable optimization choice would require the removal of liveness constraints completely, acting on the specific implementation parameters, rather than the protocol itself.
The first optimization proposal, in fact, builds exactly on this intuition.

2.2.2 Communication diagram

The resulting communication diagram is shown in the following. Please note that the propagation time, in this case, is not interesting during the calculation steps.



2.2.3 Solution scheme

Data declaration:

Given the message sizes as part of the text,

Energy spent for transmission (E_{TX}) = 50 nJ/bit,

Energy spent for reception (E_{RX}) = 58 nJ/bit,

Energy spent for average computation at the valve (E_c) = 2,4 mJ,

Being L_{CONN}, $L_{CONNACK}$, L_{SUB}, L_{SUBACK}, L_{PUB}, $L_{PINGREQ}$, $L_{PINGRESP}$ the message sizes for the CONNECT, CONNECT ACK, SUBSCRIBE, SUBSCRIBE ACK, PUBLISH, PING REQUEST, PING RESPONSE message types, respectively.

We are able to compute the number of bytes transmitted and received by each battery-operated node and therefore the energy consumed, overall.

2.2.3.1 Sensor

$$\begin{aligned} L_{SENSOR,TX} &= 1 \cdot L_{CONN} + \frac{24 \cdot 60\ minutes}{5\ minutes} \cdot L_{PUB} + \frac{24 \cdot 60\ minutes}{1\ minute} \cdot L_{PINGREQ} \\ &= 54 \text{ bytes} + 288 \cdot 68 \text{ bytes} + 1440 \cdot 52 \text{ bytes} = 94518 \text{ bytes} \end{aligned}$$

With the fraction representing the frequency of temperature updates sent via a PUBLISH message (every 5 minutes, 12 per hour) and frequency of the PING REQUEST message (liveness timeout).

$$\begin{aligned} L_{SENSOR,RX} &= 1 \cdot L_{CONNACK} + \frac{24 \cdot 60\ minutes}{1\ minute} \cdot L_{PINGRESP} \\ &= 47 \text{ bytes} + 1440 \cdot 48 \text{ bytes} = 69167 \text{ bytes} \end{aligned}$$

2.2.3.2 Valve

$$\begin{aligned} L_{VALVE,TX} &= 1 \cdot L_{CONN} + 1 \cdot L_{SUB} + \frac{24 \cdot 60\ minutes}{1\ minute} \cdot L_{PINGREQ} \\ &= 54 \text{ bytes} + 58 \text{ bytes} + 1440 \cdot 52 \text{ bytes} = 74992 \text{ bytes} \end{aligned}$$

$$\begin{aligned} L_{VALVE,RX} &= 1 \cdot L_{CONNACK} + 1 \cdot L_{SUBACK} + \frac{24 \cdot 60\ minutes}{5\ minutes} \cdot L_{PUB} + \frac{24 \cdot 60\ minutes}{1\ minute} \cdot L_{PINGRESP} \\ &= 47 \text{ bytes} + 52 \text{ bytes} + 288 \cdot 68 \text{ bytes} + 1440 \cdot 48 \text{ bytes} = 88803 \text{ bytes} \end{aligned}$$

2.2.3.3 Energy computation

Let's define

$$E_{TX} = \frac{50\ nJ}{1\ bit} = \frac{400\ nJ}{1\ byte = 1\ bit \cdot 8}$$

$$E_{RX} = \frac{58\ nJ}{1\ bit} = \frac{464\ nJ}{1\ byte = 1\ bit \cdot 8}$$

$$E_{c\ per\ day} = 2{,}4\ mJ \cdot \frac{24 \cdot 60\ minutes}{30\ minutes} = 115{,}2 \text{ mJ}$$

Being a byte composed by 8 bits

Then, we have that, finally:

$E_{SENSOR} = E_{TX} \cdot L_{SENSOR,TX} + E_{RX} \cdot L_{SENSOR,RX} = 94518\ bytes \cdot 400\ nJ/byte + 69167\ bytes \cdot 464\ nJ/byte = 37{,}807\ mJ\ + 32{,}093\ mJ = 69{,}900\ mJ$

$E_{VALVE} = E_{TX} \cdot L_{VALVE,TX} + E_{RX} \cdot L_{VALVE,RX} + E_{c\ per\ day} = 74992\ bytes \cdot 400\ nJ/byte + 88803\ bytes \cdot 464\ nJ/byte + 115{,}200\ mJ = 30{,}000\ mJ + 41{,}205\ mJ + 115{,}200\ mJ = 186{,}404\ mJ$

$E_{TOTAL} = E_{SENSOR} + E_{VALVE} = 69{,}900\ mJ + 186{,}404\ mJ = 256{,}304\ mJ$

3 Improvements proposals and possible revisions

The main point of leverage for optimizations focuses on message exchange and temperature averaging computation, being, of course, the most energy-relevant elements for the system.
The frequency and kind of messages exchanged among the nodes are the main point to tackle, alongside computational capabilities, when reviewing the overall structure of the communication.

3.1 MQTT

The following proposals aim at minimizing the number of messages exchanged between different battery-operated nodes. They will also include some restructurization of the network schema to fully explore the range of possibilities to still solve the problem, but more efficiently.

3.1.1 Deletion of the liveness constraints

As mentioned in the beginning, there is no specific upper or lower bound to the liveness constraints dictated by the protocol (that implements it through the pair of PING REQUEST and PING RESPONSE messages), but it's mainly left to the specific implementation.
As we are operating in an IoT scenario, the complete removal of the liveness constraint will definitely provide a huge benefit to the overall energy consumption.
Let's explore this possibility:

3.1.1.1 Sensor

$$L_{SENSOR,TX} = 1 \cdot L_{CONN} + \frac{24 \cdot 60\ minutes}{5\ minutes} \cdot L_{PUB} = 54 \text{ bytes} + 288 \cdot 68 \text{ bytes} = 19638 \text{ bytes}$$

$$L_{SENSOR,RX} = 1 \cdot L_{CONNACK} = 47 \text{ bytes}$$

3.1.1.2 Valve

$$L_{VALVE,TX} = 1 \cdot L_{CONN} + 1 \cdot L_{SUB} = 54 \text{ bytes} + 58 \text{ bytes} = 112 \text{ bytes}$$

$$\begin{aligned} L_{VALVE,RX} &= 1 \cdot L_{CONNACK} + 1 \cdot L_{SUBACK} + \frac{24 \cdot 60\ minutes}{5\ minutes} \cdot L_{PUB} \\ &= 47 \text{ bytes} + 52 \text{ bytes} + 288 \cdot 68 \text{ bytes} = 19683 \text{ bytes} \end{aligned}$$

3.1.1.3 Energy computation

$E_{SENSOR} = E_{TX} \cdot L_{SENSOR,TX} + E_{RX} \cdot L_{SENSOR,RX} = 19638\ bytes \cdot 400\ nJ/byte + 47\ bytes \cdot 464\ nJ/byte = 7{,}855\ mJ\ + 21{,}808\ \mu J = 7{,}877\ mJ$

$E_{VALVE} = E_{TX} \cdot L_{VALVE,TX} + E_{RX} \cdot L_{VALVE,RX} + E_{c\ per\ day} = 112\ bytes \cdot 400\ nJ/byte + 19683\ bytes \cdot 464\ nJ/byte + 115{,}200\ mJ = 44{,}800\ \mu J + 9{,}132\ mJ + 115{,}200\ mJ = 124{,}378\ mJ$

$E_{TOTAL} = E_{SENSOR} + E_{VALVE} = 7{,}877\ mJ + 124{,}378\ mJ = 132{,}255\ mJ$

Final values	
Entity	Value [mJ]
E_{SENSOR}	7,877
E_{VALVE}	124,378
E_{TOT}	132,255

As expected, calculations confirm the strong relevance of the computational effort made by the valve, energy-wise.
In the following points, we will explore other relaxations of the various constraints of the system.

3.1.2 bis: Time boundaries reduction and deletion of the liveness constraints

In the following, we also consider reduced time boundaries, with the clear assumption that the system can undertake a slower duty cycle, from both of the battery operated devices.
In particular, the new frequency for temperature capture and publishing, from the sensor, is of 15 minutes. The frequency of averaging on the valve, instead, is shifted to 45 minutes.
Let's explore this possibility:

$$E_{c\ per\ day} = 2{,}4\ mJ \cdot \frac{24 \cdot 60\ minutes}{45\ minutes} = 76{,}800\ \text{mJ}$$

3.1.2.1 Sensor

$$L_{SENSOR,TX} = 1 \cdot L_{CONN} + \frac{24 \cdot 60\ minutes}{15\ minutes} \cdot L_{PUB} = 54\ \text{bytes} + 96 \cdot 68\ \text{bytes} = 6582\ \text{bytes}$$

$$L_{SENSOR,RX} = 1 \cdot L_{CONNACK} = 47\ \text{bytes}$$

3.1.2.2 Valve

$$L_{VALVE,TX} = 1 \cdot L_{CONN} + 1 \cdot L_{SUB} = 54\ \text{bytes} + 58\ \text{bytes} = 112\ \text{bytes}$$

$$\begin{aligned} L_{VALVE,RX} &= 1 \cdot L_{CONNACK} + 1 \cdot L_{SUBACK} + \frac{24 \cdot 60\ minutes}{15\ minutes} \cdot L_{PUB} \\ &= 47\ \text{bytes} + 52\ \text{bytes} + 96 \cdot 68\ \text{bytes} = 6627\ \text{bytes} \end{aligned}$$

3.1.2.3 Energy computation

$E_{SENSOR} = E_{TX} \cdot L_{SENSOR,TX} + E_{RX} \cdot L_{SENSOR,RX} = 6582\ bytes \cdot 400\ nJ/byte + 47\ bytes \cdot 464\ nJ/byte = 2{,}633\ mJ\ + 21{,}808\ \mu J = 2{,}655\ mJ$

$E_{VALVE} = E_{TX} \cdot L_{VALVE,TX} + E_{RX} \cdot L_{VALVE,RX} + E_{c\ per\ day} = 112\ bytes\ \cdot 400\ nJ/byte + 6627\ bytes \cdot 464\ nJ/byte + 76{,}800\ mJ = 44{,}800\ \mu J + 3{,}074\ mJ + 76{,}800\ mJ = 79{,}920\ mJ$

$E_{TOTAL} = E_{SENSOR} + E_{VALVE} = 2{,}655\ mJ + 79{,}920\ mJ = 82{,}574\ mJ$

Final values	
Entity	Value [mJ]
E_{SENSOR}	2,655
E_{VALVE}	79,920
E_{TOT}	82,574

3.1.3 A versioning system

The further elaborate on the previous proposal, the next step is to further leverage the Raspberry Pi to shift the computational effort from the valve.
At each measurement, the broker acts as the subscriber to the temperature measurements and moves the subscription of the valve to a new topic, that serves as a versioning system.
In its internal memory, furthermore, it will store the current state and the previous average values and compare them.
In fact, the Raspberry Pi will be the node responsible for the average computation (symbolically representing a meaningful workload, of course).
If a significative change is detected, the newly-computed value is published to the new topic, to let the valve access it.
The energy term for the average computation is, in fact, definitely non-negligible in the standard case and can benefit from the optimization.

In the following, **a meaningful change is assumed to be present on one update over two**.
This translates one meaningful change per hour.

3.1.3.1 Sensor

$$L_{SENSOR,TX} = 1 \cdot L_{CONN} + \frac{24 \cdot 60\ minutes}{5\ minutes} \cdot L_{PUB} + \frac{24 \cdot 60\ minutes}{1\ minute} \cdot L_{PINGREQ}$$
$$= 54 \text{ bytes} + 288 \cdot 68 \text{ bytes} + 1440 \cdot 52 \text{ bytes} = 94518 \text{ bytes}$$
$$L_{SENSOR,RX} = 1 \cdot L_{CONNACK} + \frac{24 \cdot 60\ minutes}{1\ minute} \cdot L_{PINGRESP}$$
$$= 47 \text{ bytes} + 1440 \cdot 48 \text{ bytes} = 69167 \text{ bytes}$$

3.1.3.2 Valve

$$L_{VALVE,TX} = 1 \cdot L_{CONN} + 1 \cdot L_{SUB} + \frac{24 \cdot 60\ minutes}{1\ minute} \cdot L_{PINGREQ}$$
$$= 54 \text{ bytes} + 58 \text{ bytes} + 1440 \cdot 52 \text{ bytes} = 74992 \text{ bytes}$$
$$L_{VALVE,RX} = 1 \cdot L_{CONNACK} + 1 \cdot L_{SUBACK} + \frac{24 \cdot 60\ minutes}{30\ minutes} \cdot L_{PUB} \cdot \frac{1}{2} + \frac{24 \cdot 60\ minutes}{1\ minute} \cdot L_{PINGRESP}$$
$$= 47 \text{ bytes} + 52 \text{ bytes} + 48 \cdot 68 \cdot \frac{1}{2} \text{ bytes} + 1440 \cdot 48 \text{ bytes} = 70851 \text{ bytes}$$

3.1.3.3 Energy computation

Let's define

$E_{c\ per\ day} = 0$ J (shifted to the Raspberry Pi)

$E_{SENSOR} = E_{TX} \cdot L_{SENSOR,TX} + E_{RX} \cdot L_{SENSOR,RX} = 94518\ bytes \cdot 400\ nJ/byte + 69167\ bytes \cdot 464\ nJ/byte = 37{,}807\ mJ\ + 32{,}093\ mJ = 69{,}900\ mJ$

$E_{VALVE} = E_{TX} \cdot L_{VALVE,TX} + E_{RX} \cdot L_{VALVE,RX} = 74992\ bytes \cdot 400\ nJ/byte + 70851\ bytes \cdot 464\ nJ/byte = 30{,}000\ mJ + 32{,}872\ mJ = 62{,}872\ mJ$

$E_{TOTAL} = E_{SENSOR} + E_{VALVE} = 69{,}900\ mJ + 62{,}872\ mJ = 132{,}772\ mJ$

Final values	
Entity	Value [mJ]
E_{SENSOR}	69,900
E_{VALVE}	62,872
E_{TOT}	132,772

3.1.4 bis: Time boundaries reduction and a versioning system

In the following, as shown for the previous case, we also consider reduced time boundaries, with the clear assumption that the system can undertake a slower duty cycle, from both of the battery operated devices. In particular, the new frequency for temperature capture and publishing, from the sensor, is of 15 minutes. The frequency of averaging on the valve, instead, is shifted to 45 minutes.

3.1.4.1 Sensor

$$L_{SENSOR,TX} = 1 \cdot L_{CONN} + \frac{24 \cdot 60\ minutes}{15\ minutes} \cdot L_{PUB} + \frac{24 \cdot 60\ minutes}{1\ minute} \cdot L_{PINGREQ}$$
$$= 54 \text{ bytes} + 96 \cdot 68 \text{ bytes} + 1440 \cdot 52 \text{ bytes} = 81462 \text{ bytes}$$
$$L_{SENSOR,RX} = 1 \cdot L_{CONNACK} + \frac{24 \cdot 60\ minutes}{1\ minute} \cdot L_{PINGRESP}$$
$$= 47 \text{ bytes} + 1440 \cdot 48 \text{ bytes} = 69167 \text{ bytes}$$

3.1.4.2 Valve

$$L_{VALVE,TX} = 1 \cdot L_{CONN} + 1 \cdot L_{SUB} + \frac{24 \cdot 60\ minutes}{1\ minute} \cdot L_{PINGREQ}$$
$$= 54 \text{ bytes} + 58 \text{ bytes} + 1440 \cdot 52 \text{ bytes} = 74992 \text{ bytes}$$
$$L_{VALVE,RX} = 1 \cdot L_{CONNACK} + 1 \cdot L_{SUBACK} + \frac{24 \cdot 60\ minutes}{45\ minutes} \cdot L_{PUB} \cdot \frac{1}{2} + \frac{24 \cdot 60\ minutes}{1\ minute} \cdot L_{PINGRESP}$$
$$= 47 \text{ bytes} + 52 \text{ bytes} + 32 \cdot 68 \cdot \frac{1}{2} \text{ bytes} + 1440 \cdot 48 \text{ bytes} = 70307 \text{ bytes}$$

3.1.4.3 Energy computation

Let's define

$E_{c\ per\ day} = 0$ J (shifted to the Raspberry Pi)

$E_{SENSOR} = E_{TX} \cdot L_{SENSOR,TX} + E_{RX} \cdot L_{SENSOR,RX} = 81462\ bytes \cdot 400\ nJ/byte + 69167\ bytes \cdot 464\ nJ/byte = 32{,}585\ mJ\ + 32{,}093\ mJ = 64{,}678\ mJ$

$E_{VALVE} = E_{TX} \cdot L_{VALVE,TX} + E_{RX} \cdot L_{VALVE,RX} = 74992\ bytes \cdot 400\ nJ/byte + 70307\ bytes \cdot 464\ nJ/byte = 30{,}000\ mJ + 32{,}619\ mJ = 62{,}619\ mJ$

$E_{TOTAL} = E_{SENSOR} + E_{VALVE} = 64{,}678\ mJ + 62{,}619\ mJ = 127{,}298\ mJ$

Final values	
Entity	Value [mJ]
E_{SENSOR}	64,678
E_{VALVE}	62,619
E_{TOT}	127,298

3.1.5 A complete merge

In the following, we propose a merge of all the best techniques shown in the previous points.
This translates to a versioning system, with reduced time boundaries and negligible liveness constraints.

3.1.5.1 Sensor

$$L_{SENSOR,TX} = 1 \cdot L_{CONN} + \frac{24 \cdot 60\ minutes}{15\ minutes} \cdot L_{PUB} = 54 \text{ bytes} + 96 \cdot 68 \text{ bytes} = 6582 \text{ bytes}$$
$$L_{SENSOR,RX} = 1 \cdot L_{CONNACK} = 47 \text{ bytes}$$

3.1.5.2 Valve

$$L_{VALVE,TX} = 1 \cdot L_{CONN} + 1 \cdot L_{SUB} = 54 \text{ bytes} + 58 \text{ bytes} = 112 \text{ bytes}$$
$$L_{VALVE,RX} = 1 \cdot L_{CONNACK} + 1 \cdot L_{SUBACK} + \frac{24 \cdot 60\ minutes}{45\ minutes} \cdot L_{PUB} \cdot \frac{1}{2}$$
$$= 47 \text{ bytes} + 52 \text{ bytes} + 32 \cdot 68 \cdot \frac{1}{2} \text{ bytes} = 1187 \text{ bytes}$$

3.1.5.3 Energy computation

Let's define

$E_{c\ per\ day} = 0$ J (shifted to the Raspberry Pi)

$E_{SENSOR} = E_{TX} \cdot L_{SENSOR,TX} + E_{RX} \cdot L_{SENSOR,RX} = 6582\ bytes \cdot 400\ nJ/byte + 47\ bytes \cdot 464\ nJ/byte = 2{,}633\ mJ\ + 21{,}808\ \mu J = 2{,}655\ mJ$

$E_{VALVE} = E_{TX} \cdot L_{VALVE,TX} + E_{RX} \cdot L_{VALVE,RX} = 112\ bytes \cdot 400\ nJ/byte + 1187\ bytes \cdot 464\ nJ/byte = 44{,}800\ \mu J + 550{,}768\ \mu J = 595{,}568\ \mu J$

$E_{TOTAL} = E_{SENSOR} + E_{VALVE} = 2{,}655\ mJ + 595{,}568\ \mu J = 3{,}250\ mJ$

Final values	
Entity	Value [mJ]
E_{SENSOR}	2,655
E_{VALVE}	0,596
E_{TOT}	3,250

As we clearly determined from the first proposal, the presence of liveness constraints turn out to be the heaviest, energy-wise.

3.2 CoAP

Being the CoAP protocol based on UDP, the adoption of CoAP leads to a reasonably lower result, in itself.
However, a couple of shifts in computation and time boundaries could still be applied, like shown in the following.

3.2.1 Time boundaries reduction

In the following, we consider reduced time boundaries, with the clear assumption that the system can undertake a slower duty cycle, from both of the battery operated devices.
In particular, the new frequency for temperature capture and publishing, from the sensor, is of 15 minutes. The frequency of averaging on the valve, instead, is shifted to 45 minutes.

3.2.1.1 Sensor

$$L_{SENSOR,TX} = \frac{24 \cdot 60\ minutes}{15\ minutes} \cdot L_{GET\ Response} = 96 \cdot 55 \text{ bytes} = 5280 \text{ bytes}$$

$$L_{SENSOR,RX} = 1 \cdot L_{GET\ Request} = 60 \text{ bytes}$$

3.2.1.2 Valve

$$L_{VALVE,TX} = 1 \cdot L_{GET\ Request} = 60 \text{ bytes}$$

$$L_{VALVE,RX} = \frac{24 \cdot 60\ minutes}{15\ minutes} \cdot L_{GET\ Response} = 96 \cdot 55 \text{ bytes} = 5280 \text{ bytes}$$

3.2.1.3 Energy computation

Let's define

$$E_{c\ per\ day} = 2{,}4\ mJ \cdot \frac{24 \cdot 60\ minutes}{45\ minutes} = 76{,}800 \text{ mJ}$$

$E_{SENSOR} = E_{TX} \cdot L_{SENSOR,TX} + E_{RX} \cdot L_{SENSOR,RX} = 5280\ bytes \cdot 400\ nJ/byte + 60\ bytes \cdot 464\ nJ/byte = 2{,}112\ mJ\ + 27{,}840\ \mu J = 2{,}140\ mJ$

$E_{VALVE} = E_{TX} \cdot L_{VALVE,TX} + E_{RX} \cdot L_{VALVE,RX} + E_{c\ per\ day} = 60\ bytes\ \cdot 400\ nJ/byte + 5280\ bytes\ \cdot 464\ nJ/byte + 76{,}800\ mJ = 24{,}000\ \mu J + 2{,}450\ mJ + 76{,}800\ mJ = 79{,}274\ mJ$

$E_{TOTAL} = E_{SENSOR} + E_{VALVE} = 2{,}140\ mJ + 79{,}274\ mJ = 81{,}414\ mJ$

Final values	
Entity	Value [mJ]
E_{SENSOR}	2,140
E_{VALVE}	79,274
E_{TOT}	81,414

3.2.2 Shift the workload to the sensor

With the assumption that the energy consumed by the temperature sensor to perform the average computation is the same one of the valve, it is possible to exploit the direct knowledge at the sensor side.
The temperature sensor, therefore, computes itself the standard average every 30 minutes, informing the valve of the update in its operation mode, directly.
This, translates to a single GET RESPONSE per measurement and average cycle.

3.2.2.1 Sensor

$$L_{SENSOR,TX} = \frac{24 \cdot 60\ minutes}{30\ minutes} \cdot L_{GET\ Response} = 48 \cdot 55 \text{ bytes} = 2640 \text{ bytes}$$

$$L_{SENSOR,RX} = 1 \cdot L_{GET\ Request} = 60 \text{ bytes}$$

3.2.2.2 Valve

$$L_{VALVE,TX} = 1 \cdot L_{GET\ Request} = 60 \text{ bytes}$$

$$L_{VALVE,RX} = \frac{24 \cdot 60\ minutes}{30\ minutes} \cdot L_{GET\ Response} = 48 \cdot 55 \text{ bytes} = 2640 \text{ bytes}$$

3.2.2.3 Energy computation

Let's define

$$E_{c\ per\ day} = 2{,}4\ mJ \cdot \frac{24 \cdot 60\ minutes}{30\ minutes} = 115{,}200 \text{ mJ}$$

$E_{SENSOR} = E_{TX} \cdot L_{SENSOR,TX} + E_{RX} \cdot L_{SENSOR,RX} = 2640\ bytes \cdot 400\ nJ/byte + 60\ bytes \cdot 464\ nJ/byte + 115{,}200\ mJ = 1{,}056\ mJ\ + 27{,}840\ \mu J + 115{,}200\ mJ = 116{,}284\ mJ$

$E_{VALVE} = E_{TX} \cdot L_{VALVE,TX} + E_{RX} \cdot L_{VALVE,RX} + E_{c\ per\ day} = 60\ bytes \cdot 400\ nJ/byte + 2640\ bytes \cdot 464\ nJ/byte = 24{,}000\ \mu J + 1{,}225\ mJ = 1{,}249\ mJ$

$E_{TOTAL} = E_{SENSOR} + E_{VALVE} = 116{,}284\ mJ + 1{,}249\ mJ = 117{,}533\ mJ$

Final values	
Entity	Value [mJ]
E_{SENSOR}	116,284
E_{VALVE}	1,249
E_{TOT}	117,533

As confirmed by our previous intuitions, one of the most energy-consuming element is definitely located in the average computation, that can't be shifted to the broker in CoAP, like instead was possible in MQTT, leveraging the broker node.