Project Winter 2025

Evaluation:

a) implementation in a full, smart implementation (25% τελικός, 25% εργαστήριο, 50% project)
b) basic implementation (50% τελικός, 25% εργαστήριο, 25% project)

Note: (a) the implementation can be done on any system (by simulating a random number generator instead of temperature), (b) using open-source, such as libraries, is permitted.

Goal:

As more and more embedded systems operate on batteries and try to save energy, intelligent management of the large volume of data from their sensors requires new techniques for storing it in the device's limited flash memory and sending this data to the cloud. For this reason, the goal of this project is to manage data from incoming sensor values (e.g. temperature sensors), locally in the embedded system.

Guidelines:

Assume 3 sources of sensor data, with incoming arrival of data with different rates (due to network congestion, intermittent faults): sourceA: 2 temp values every 1 sec (+/- 0.5sec), sourceB: 3 temp values every 3 sec (+/- 0.5sec), sourceC: 1 temp values every 2 sec (+/- 1sec),

One thread (or three?) is responsible to generate the incoming traffic and fill queues qA, qB, qC of size 10 values (or 100).

Processing thread (processing internally):

i. periodically collect and generate a bucket of data from all incoming queues (input queue maybe full or may have empty space), for instance every 20 sec.

ii. collect data when one queue is full.

The bucket assembled with collected data, will contain 10 positions for sourceA, 10 for B and 10 for C.

Sending thread:

a. compress bucket and send to cloud (instead of cloud you may assume the uart) where uncompress them

b. detect anomalies and send average and max (of each queue)

c. display in the OLED a notification in case of emergency (e.g. temperature>threshold_red)

Report (i) the latency to compress and send, (ii) the efficiency of compression (percentage saved), (iii) the effective queues occupancy

<u>Useful links:</u>

Lightweight databases for IoT: https://github.com/armink/FlashDB

https://www.devcoons.com/a-simple-generic-c-queue-for-static-or-dynamic-memory-usage/

https://www.freertos.org/mqtt/index.html

https://blog.reds.ch/?p=1671

https://www.techiedelight.com/circular-queue-implementation-c/

https://github.com/topics/lock-free-queue?o=asc&s=updated

https://github.com/topics/circular-queue?l=c%2B%2B

https://github.com/topics/circular-queue?l=c