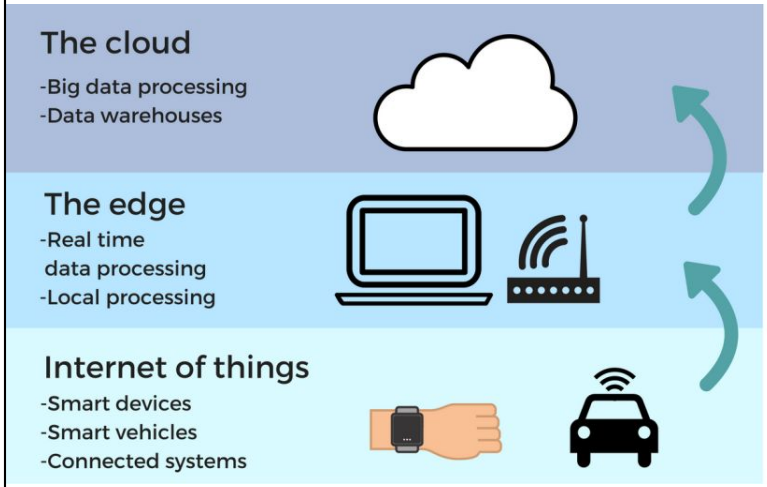
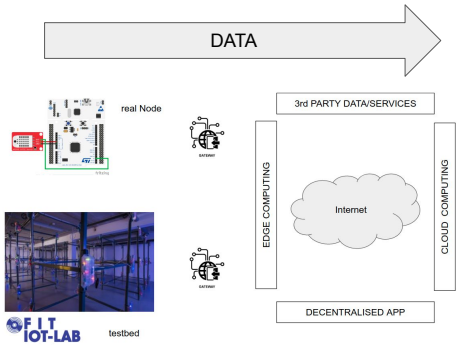
Performance Evaluation for IoT

Test complex scenarios

Andrea Vitaletti

Note: the images are linked to the sources





Your final deployment, in its essence, should look similar to the one depicted in the picture.

- The “real node” is used to prove your ability of integrating the specific sensors envisioned in your application scenario
- The Iot-lab deployment is used to prove you ability to scale-up your solution into an ecosystem of nodes organised in a network and to evaluate their performance

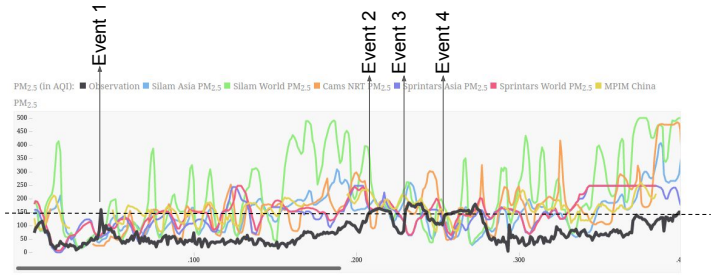
What kind of phenomena?

- Sampling
  - reconstruct a signal (e.g. black PM2.5)
  - observe a value … you already know the dynamics!
- Events (e.g >threshold)

PM2.5 air pollution

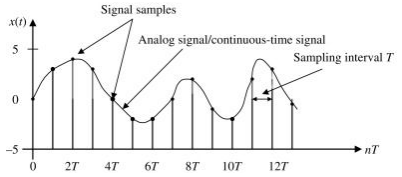
https://aqicn.org/forecast

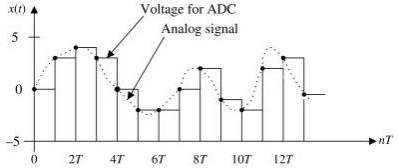
threshold



Reconstruct a signal

Sampling Theorem in few slides

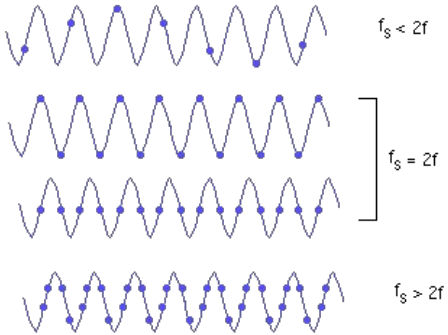


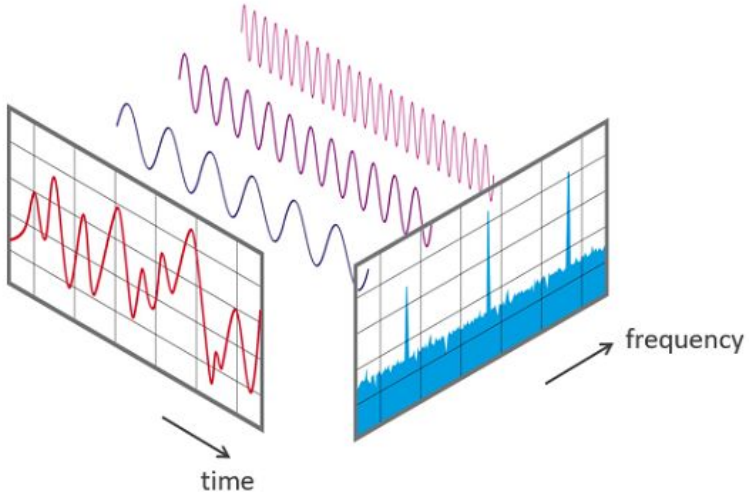


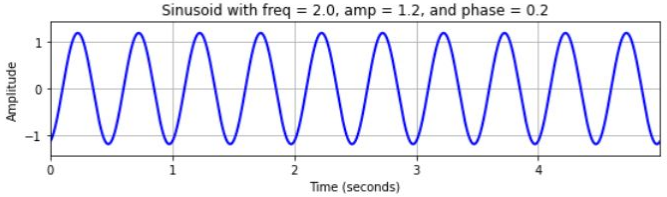
https://www.sciencedirect.com/topics/computer-science/shannon-sampling-theorem

The **sampling theorem** specifies the minimum-**sampling** rate at which a continuous-time signal needs to be uniformly **sampled** so that the original signal can be completely recovered or reconstructed by these samples alone.

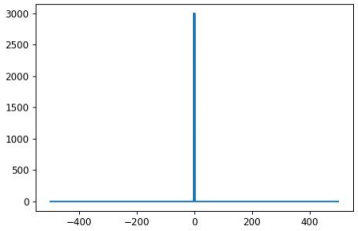
$$f_s > 2f_{max}$$







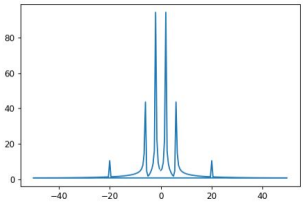
$$x = sin(2\pi freq \cdot t - phase)$$

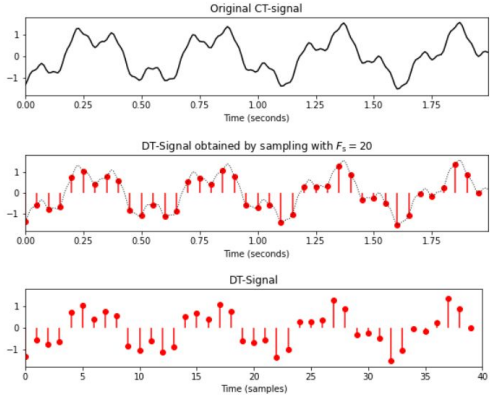


jupyter

https://colab.research.google.com/drive/1nuZx095lzt2d9h42N7yNu13crGg9eS9A?usp=sharing

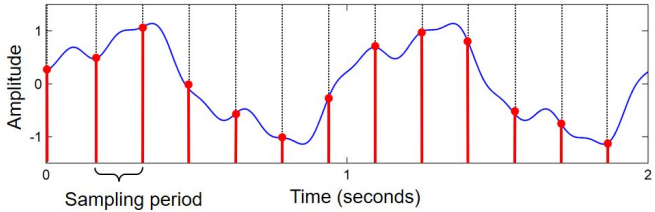
$$x = sin(2\pi \cdot 1.9t - 0.3) + 0.5sin(2\pi \cdot 6.1t - 0.1) + 0.1sin(2\pi \cdot 20t - 0.2)$$

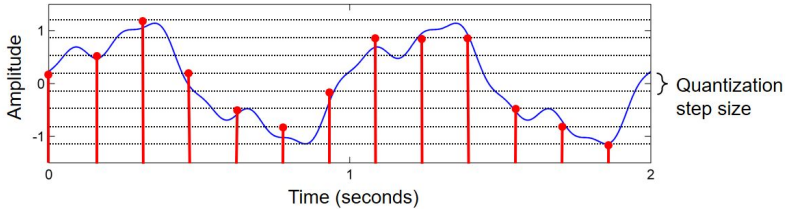


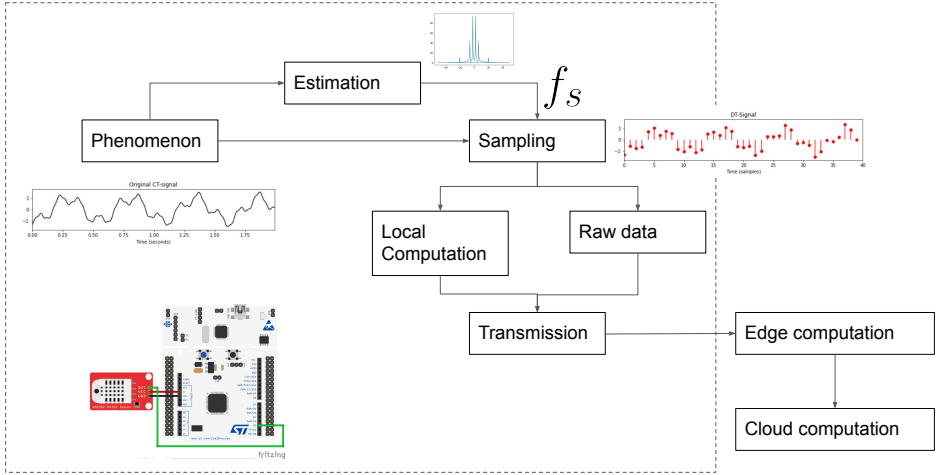


jupyter

Figure 2.13 from [Müller, FMP, Springer 2015]



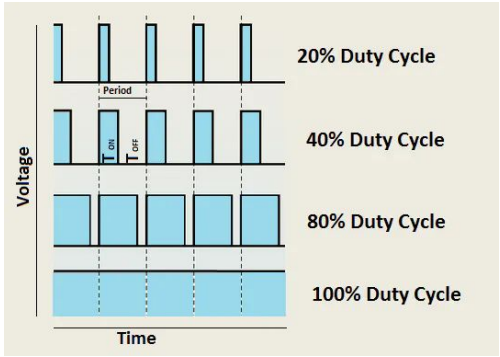


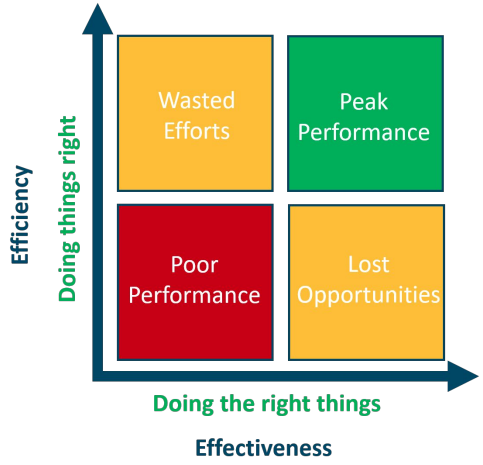


f_s

Samples per second

Duty Cycle → Energy



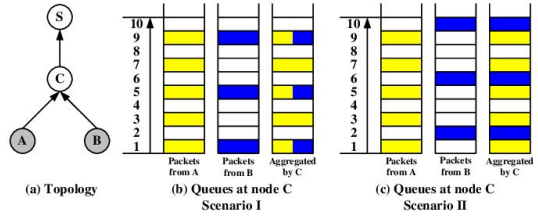


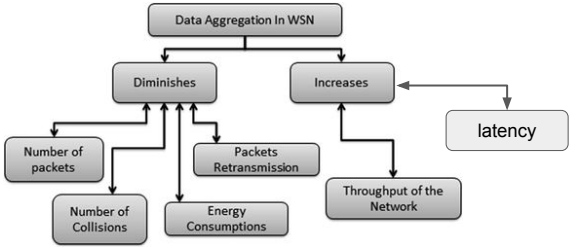
SPEED → CAPACITY → ENERGY → COST

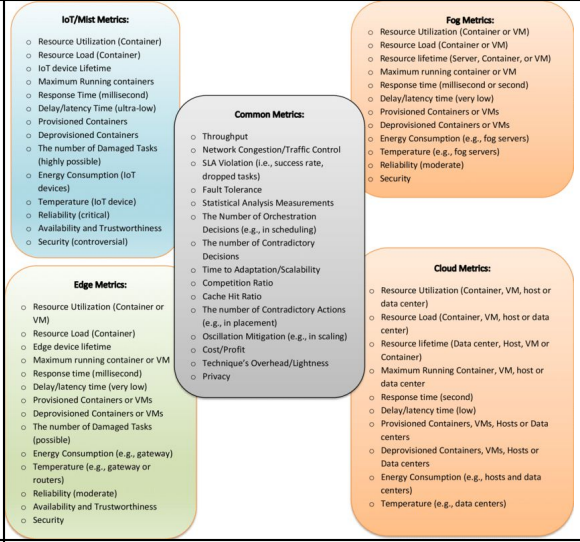
NETWORK LAYER	QoS STRATEGY	IMPROVEMENT in QoS METRIC
Application Layer	Compression	Latency, Available Bandwidth
	Adjust Sensing Rate	Network Lifetime, QoD
	Data Reduction Techniques	Latency, Available Bandwidth
Network Layer	Routing Protocols	Realibility, Latency, Network Lifetime
	Packets Priority	Realibility, Latency, Network Lifetime
Physical Layer	Energy Aware MACs	Network Lifetime
	Selection of low interference channels	Reliability, Network Lifetime
	MAC that Avoid Collisions	Reliability, Network Lifetime
Link Layer	Channel Surfing	Reliability, Network Lifetime
	Modifying Signal Power	Network Lifetime
	Using low interference channel	Reliability, Network Lifetime

https://www.researchgate.net/figure/A-Summary-of-WSN-strategies-that-produce-improvement-in-QoS-metrics_fig1_266143081

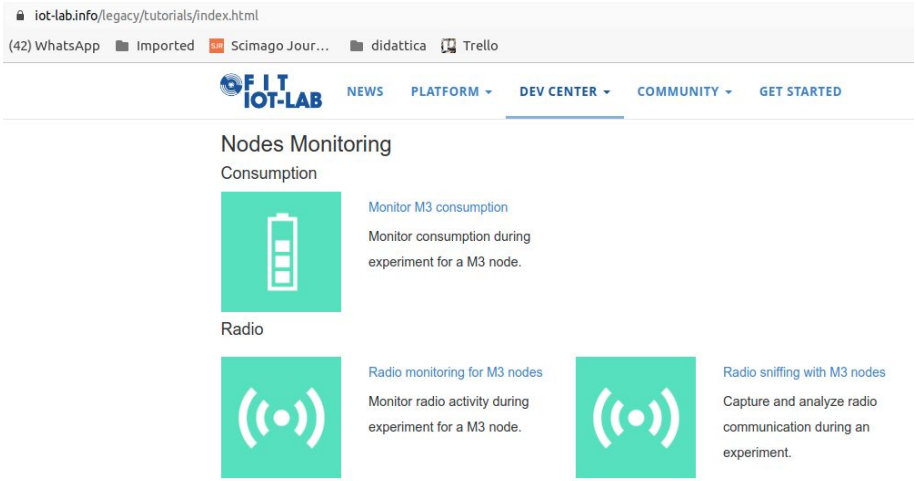
Aggregation

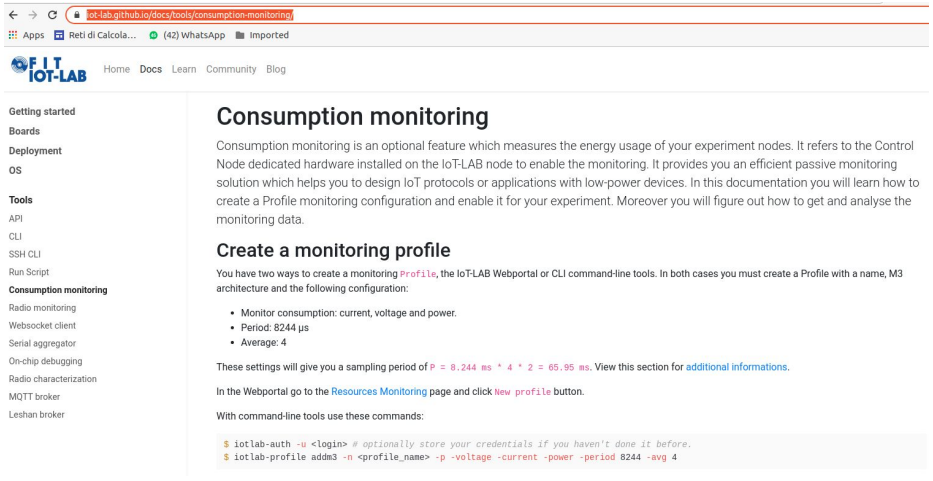




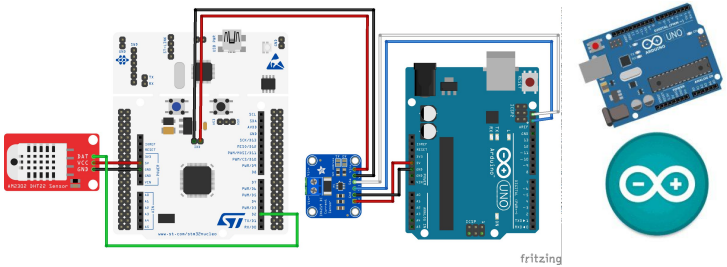


https://www.researchgate.net/figure/A-taxonomy-of-real-world-performance-metrics-for-evaluating-IoT-Mist-Edge-Fog-and-Cloud_fig2_343578510

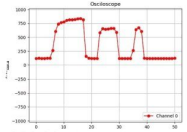




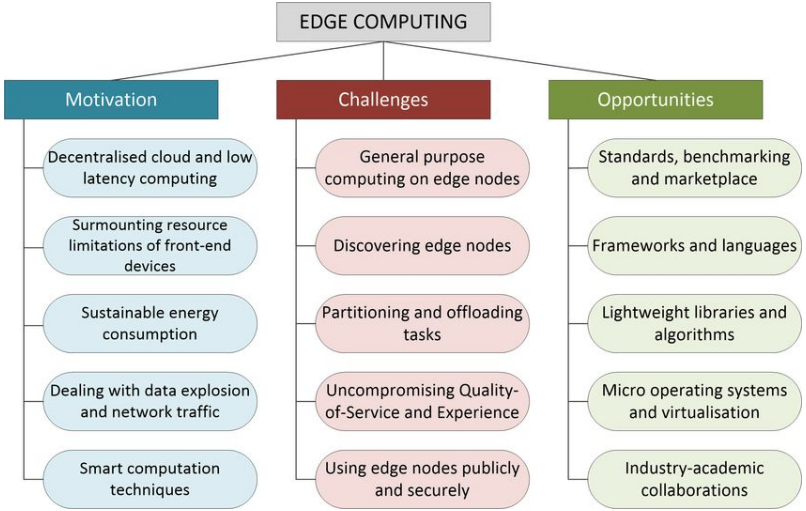
https://github.com/RIOT-OS/RIOT/tree/master/tests/periph_pm



Arduino Oscilloscope

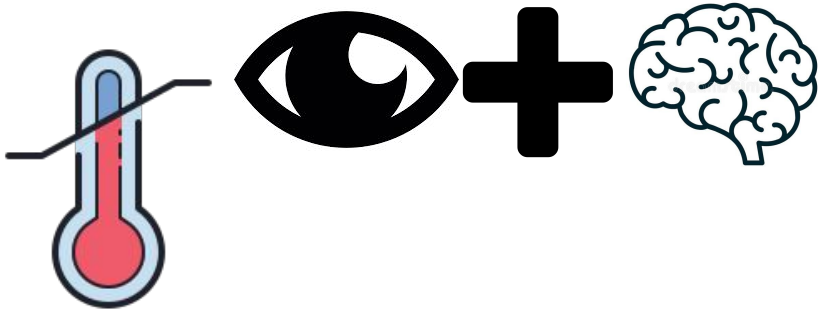


https://github.com/ichatz/riotos-apps/tree/main/temperature_humidity

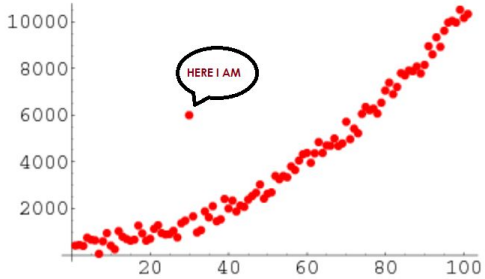


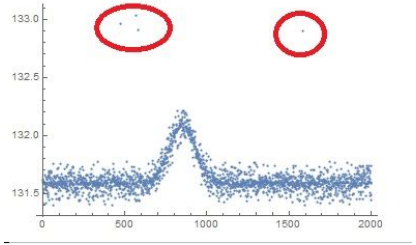
Events

Something interesting happens!



Outliers: interesting events or noise?





A short journey of outlier detection

Quick overview of several methods for finding outliers

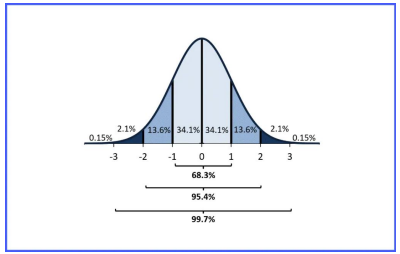
Ryota Bannai Jan 12, 2019 · 5 min read ★

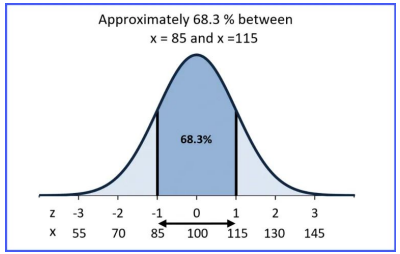
Let's focus on Z-score

$$z = \frac{x - \mu}{\rho}$$

where
- x is raw data
- μ is just mean of x
- ρ is standard deviation of x.

OUTLIER: x whose |Z-Score|>threshold





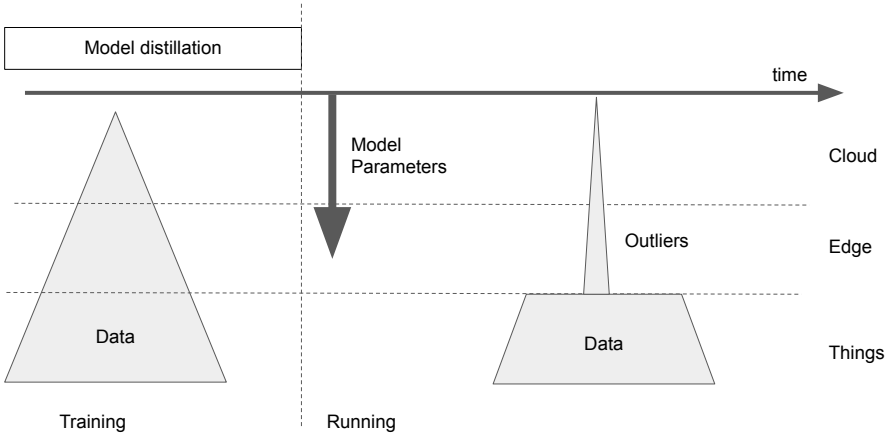
Source: https://www.drdawnwright.com/empirical-rule-and-z-score-probability/

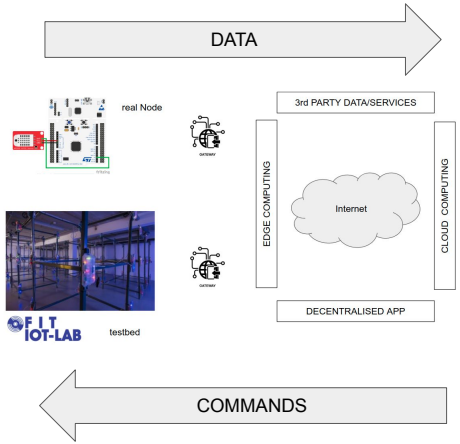
```
import numpy as np
x=np.array([1,4,7,2,5,7,7,8,4,6,8,30])
z=(x-x.mean(axis=0))/x.std()
#
for l in (z < z.mean() - 3*z.std(),
          z > z.mean() + 3*z.std()):
    if np.any(z[l]):
        print(z[l])
```

outlier_w_zscore.py hosted with ❤ by GitHub view raw

$$z = \frac{x - \mu}{\rho}$$

Where are we going to compute μ and ρ?





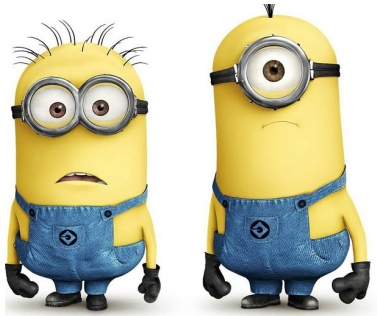
Your final deployment, in its essence, should look similar to the one depicted in the picture.

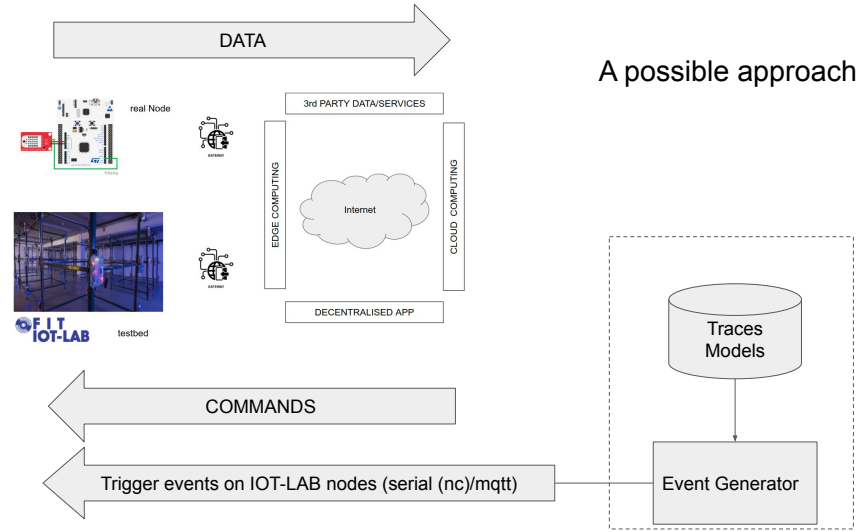
- The “real node” is used to prove you ability of integrating the specific sensors envisioned in your application scenario
- The Iot-lab deployment is used to prove you ability to scale-up your solution into an ecosystem of nodes organised in a network and to evaluate their performance

Problem

Since we cannot integrate arbitrary sensors (e.g. stress sensor) in the Iot-Lab deployment, and in any case we do not have the ability of generating specific events to trigger interesting monitoring activities (e.s. apply a force to a sensor)

how can we prove the efficiency/effectiveness of the proposed solution in “realistic” application scenarios?

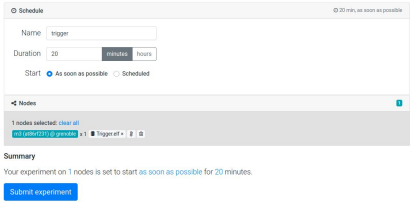




A possible approach

A firmware skeleton

```
BOARD=iotlab-m3 make all
```




```c
#include <stdio.h>
#include <string.h>
#include "shell.h"

int trigger(int argc, char **argv)
{
    printf("triggered %s\n", argv[1]);
    (void)argc;
    (void)argv;
    return 0;
}

static const shell_command_t commands[] = {
    { "trigger", "trigger an event", trigger },
    { NULL, NULL, NULL }
};

int main(void)
{
    puts("Trigger events");
    char line_buf[SHELL_DEFAULT_BUFSIZE];
    shell_run(commands, line_buf, SHELL_DEFAULT_BUFSIZE);
    return 0;
}
```

/home/andrea/Documents/University/teaching/IoT/iot-lab/RIOT/examples/triggers

Experiment trigger #256238

User **vitalett**
Submitted **2021-03-29 22:23:29**
Started **2021-03-29 22:23:30**
Duration 0 minutes (0%) of **20 minutes**
Nodes **1**
State Running

Stop | Download | Actions on selected nodes

Nodes	UID	Firmware	Monitoring	Deployment	Actions
m3-101.grenoble.iot-lab.info	9181	Trigger.elf		Success	

```
main(): This is RIOT! (Version: 2021.01)
Trigger events
> trigger temperature
trigger temperature
triggered trigger:temperature
>
```

Remember to close the terminal

Nodes	UID	Firmware	Monitoring	Deployment	Actions
m3-101.grenoble.iot-lab.info	9181	Trigger.elf		Success	

Nodes	UID	Firmware	Monitoring	Deployment	Actions
m3-101.grenoble.iot-lab.info	9181	Trigger.elf		Success	

```
andrea@andrea-HP-EliteBook-830-G5:~$ ssh vitalett@grenoble.iot-lab.info
Linux grenoble 4.19.0-13-amd64 #1 SMP Debian 4.19.160-2 (2020-11-28) x86_64
Welcome FIT IoT-LAB users
```

```
vitalett@grenoble:~$ nc m3-101 20000
trigger humidity
trigger humidity
triggered trigger:humidity
>
```

```
andrea@andrea-HP-EliteBook-830-G5:~$ ssh -L 20000:m3-101:20000 vitalett@grenoble.iot-lab.info
Linux grenoble 4.19.0-13-amd64 #1 SMP Debian 4.19.160-2 (2020-11-28) x86_64
Welcome FIT IoT-LAB users
```

```
andrea@andrea-HP-EliteBook-830-G5:~$ nc localhost 20000
trigger ciao
trigger ciao
triggered trigger:ciao
> ^C
andrea@andrea-HP-EliteBook-830-G5:~$
```

```
#!/bin/bash
for i in 1 2 3 4 5
do
   echo "Trigger $i times"
done
```

https://unix.stackexchange.com/questions/332163/netcat-send-text-to-echo-service-read-reply-then-exit

iot-lab.info/legacy/tutorials/serial-aggregator/index.html

2) WhatsApp | Imported | Scimago Jour... | didattica | Trello

F I T IOT-LAB | NEWS | PLATFORM | DEV CENTER | COMMUNITY | GET STARTED | ACTIVITY | Access the testbed

Nodes Serial Link Aggregation

Difficulty: Medium

Duration: 30 minutes

***Prerequisites**: Submit an experiment with M3 nodes using the web portal / Experiment CLI Client*

***Description**: This document shows how to use the `serial_aggregator` tool on the ssh frontend. You will learn how to quickly get the output of all your experiment nodes and how to send messages to some of them.*

Contents [hide]

Description

Each node serial link can be accessed via a tcp socket to the node url on port 20000. Here is an example using net cat on node M3 with Id 10 on Grenoble site involved in your running experiment :

```
user@grenoble-$ nc m3-10 20000
```

When running experiments with many nodes, you may want to access the serial link of all the nodes of your experiment at once. The presented tool `serial_aggregator` should relieve the pain of having multiple terminals each connected to one node.

You will have all the nodes output in one terminal, prefixed by a timestamp and the node identifier.

```
$ harter@grenoble:~$ serial_aggregator
1417086154.415922:Aggregator started
1417086156.031700;m3-11;     iotlab_uid() == 9971
1417086156.039846;m3-11;   platform_uid() == 9971
1417086156.039986;m3-11;node_from_uid(id) == 3c00B
1417086156.040103;m3-11;          node_str == m3-11
1417086157.998551;m3-10;     iotlab_uid() == 8772
1417086157.999130;m3-10;   platform_uid() == 8772
1417086157.999130;m3-10;node_from_uid(id) == 0:000
1417086157.999130;m3-10;Unknown node
1417086162.639063;Stopping                  # Stopping using 'Ctrl+C'
```

riot-os.org/api/group__drivers__saul.html

Apps | (42) WhatsApp | Imported | Scimago Jour... | didattica | Trello

RIOT — The friendly Operating System for the Internet of Things | Main Page | Related Pages | Modules | Data Structures

[S]ensor [A]ctuator [U]ber [L]ayer

Drivers

Generic sensor/actuator abstraction layer for RIOT. More...

Detailed Description

Generic sensor/actuator abstraction layer for RIOT.

SAUL is a generic actuator/sensor interface in RIOT. Its purpose is to enable unified intera

Each device driver implementing this interface has to expose a set of predefined functions

Each device has further to expose a name and its type. This information can be used for a

The SAUL module enables further the automated initialization of preconfigured actuators/s

SAUL drivers may rely on being called from a thread, and often block for short amounts of

Todo:

So far, the interface only supports simple read and set operations. It probably needs to

See also

SAUL registry

/home/andrea/Documents/University/teaching/IoT/iot-lab/RIOT/examples/saul

https://github.com/RIOT-OS/RIOT/tree/master/examples/saul

Experiment test #256239

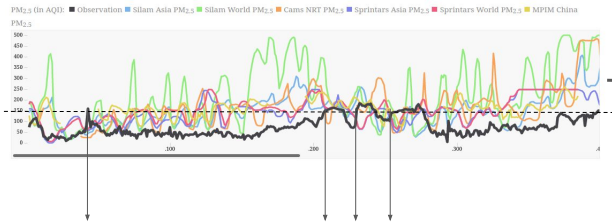
User **vitalett**
Submitted **2021-03-29 22:42:24**
Started **2021-03-29 22:42:26**
Duration 1 minute (5%) of **20 minutes**
Nodes **1**
State **Running**

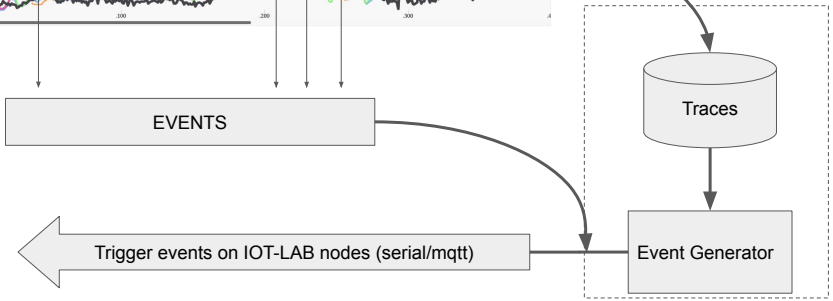
Stop | Download | Actions on selected nodes

Nodes	UID	Firmware	Monitoring	Deployment	Actions
m3-96.grenoble.iot-lab.info	b468	saul_example.elf		Success	

```
#6      SENSE_TEMP      lps331ap
#7      SENSE_ACCEL     lsm303dlhc
#8      SENSE_MAG       lsm303dlhc
> saul
saul
ID      Class           Name
#0      ACT_SWITCH      LED(red)
#1      ACT_SWITCH      LED(green)
#2      ACT_SWITCH      LED(orange)
#3      SENSE_LIGHT     isl29020
#4      SENSE_GYRO      l3g4200d
#5      SENSE_PRESS     lps331ap
#6      SENSE_TEMP      lps331ap
#7      SENSE_ACCEL     lsm303dlhc
#8      SENSE_MAG       lsm303dlhc
> saul read 6
saul read 6
Reading from #6 (lps331ap|SENSE_TEMP)
Data:           34.24 °C
>
```

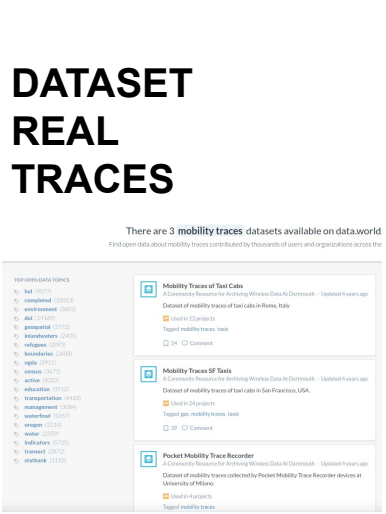
An example https://aqicn.org/forecast



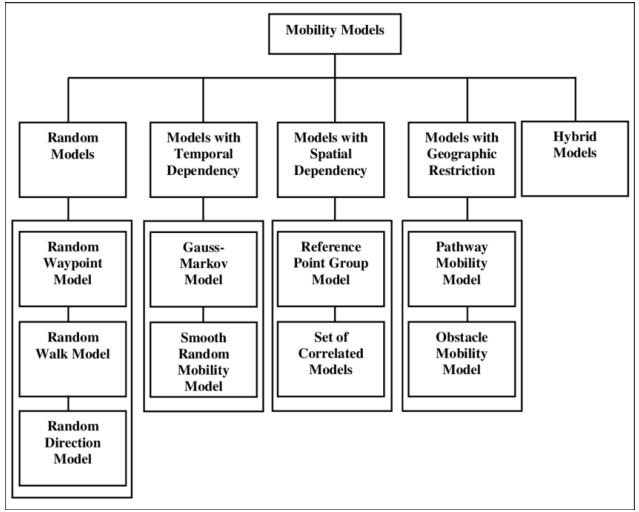


Assume you have to monitor PM2.5 air pollution and you are interested in events generated when observations are above 150

DATASET REAL TRACES

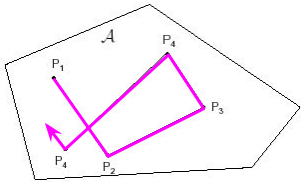


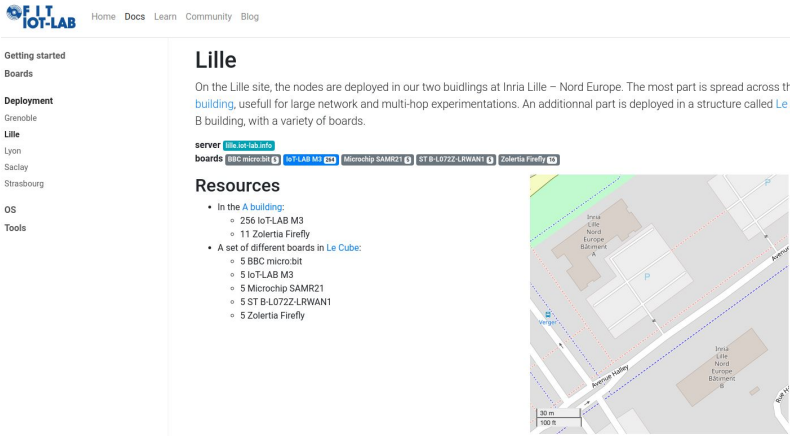
MODELS SYNTHETIC DATA

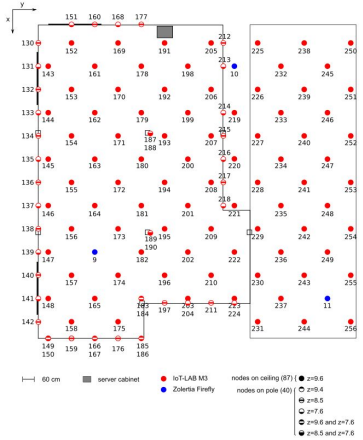


Random Way Point model:

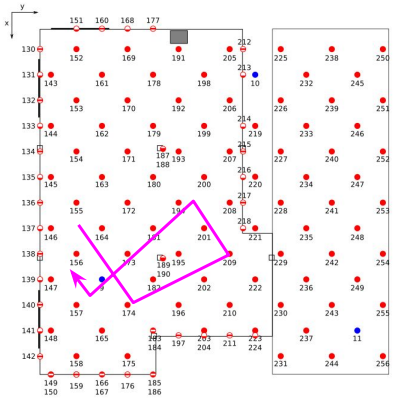
- Each node moves along a zigzag line from one waypoint P_i to the next P_{i+1}.
- The waypoints are uniformly distributed over the given convex area, e.g. unit disk.
- At the start of each leg a random velocity is drawn from the velocity distribution. (in the basic case the velocity is constant 1)
- Optionally, the nodes may have so-called "thinking times" when they reach each waypoint before continuing on the next leg, where durations are independent and identically distributed random variables.





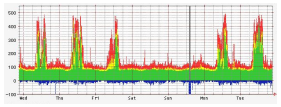


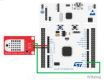
- Ground floor: m3-[1-45], firefly-[1-3]
- First floor: m3-[46-111], firefly-[4-6]
- Second floor: m3-[112-129], firefly-[7-8]
- Dedicated room (2nd floor): m3-[130-256], firefly[9-11]

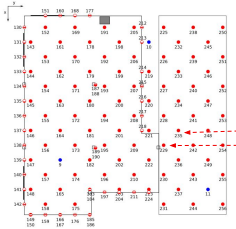


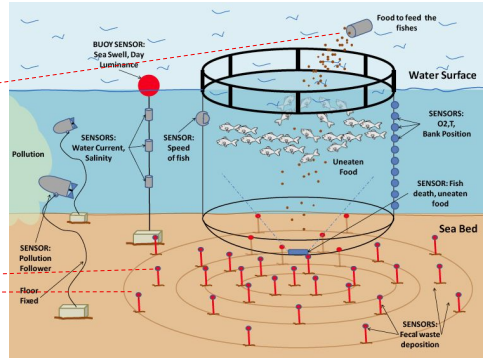
Remember … make reasonable assumptions and always support your claims with quantitative evidences!

REMEMBER









ASSUMPTION: Simulation of underwater links with wireless … I'm perfectly aware this is a simplistic assumption because … but …. Then I consider it as an upper bound on performance!