The Hera framework

for

Fault-Tolerant Sensor Fusion

with

Erlang and GRiSP

on an

IoT network

Sébastien Kalbusch
Vincent Verpoten
Peter Van Roy

Université catholique de Louvain

20th ACM SIGPLAN Erlang Workshop

Outline

1. Introduction: IoT and Sensor Fusion
2. Context
    1. Traditional IoT architecture
    2. Low-cost devices for IoT
3. The Hera framework
    1. Purpose
    2. Sensor fusion engine
    3. Software architecture
4. Fault tolerance
5. Examples of sensor fusion with Hera
6. Conclusion and future work

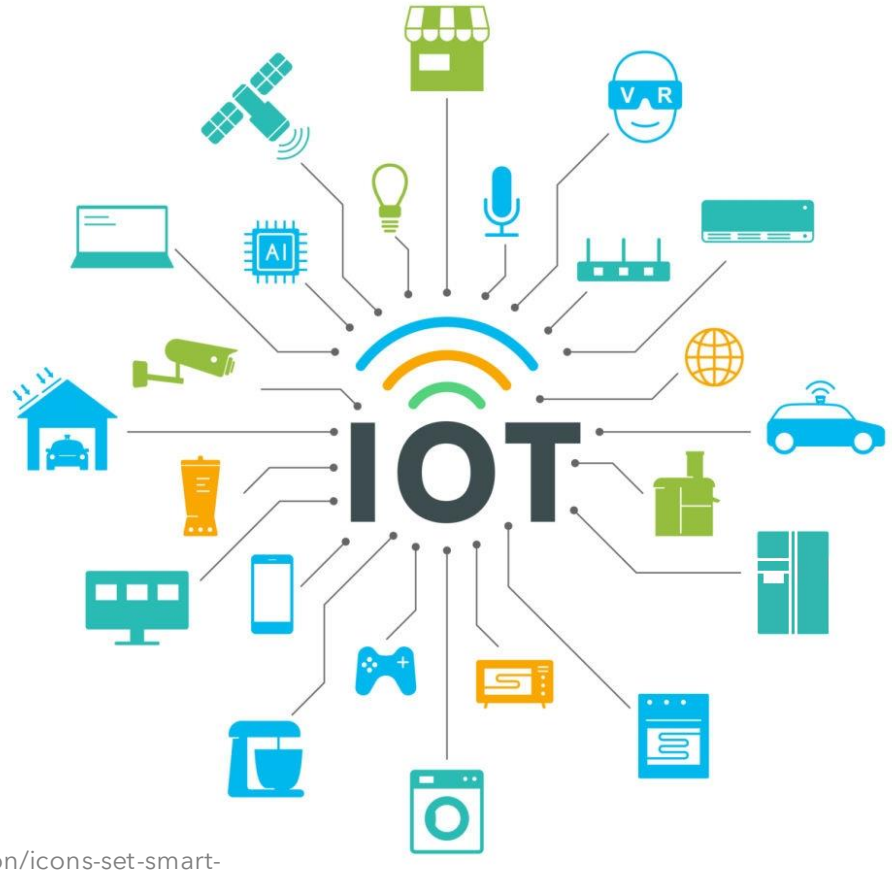
Introduction

IoT

and

Sensor fusion

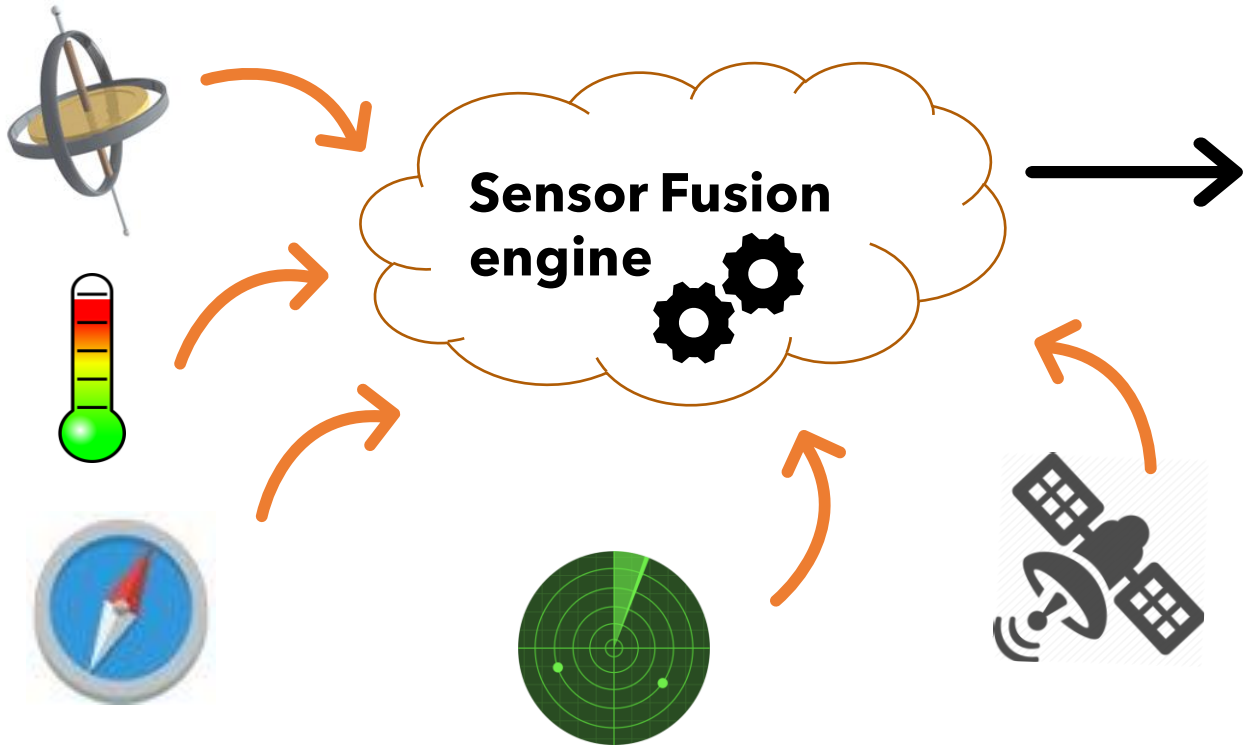
The Internet of Things (IoT)



- Large number
- Low-power
- At the Edge
- Wireless

https://www.gettyimages.ca/detail/illustration/icons-set-smart-appliances-concept-of-future-royalty-free-illustration/1180939305

What is sensor fusion



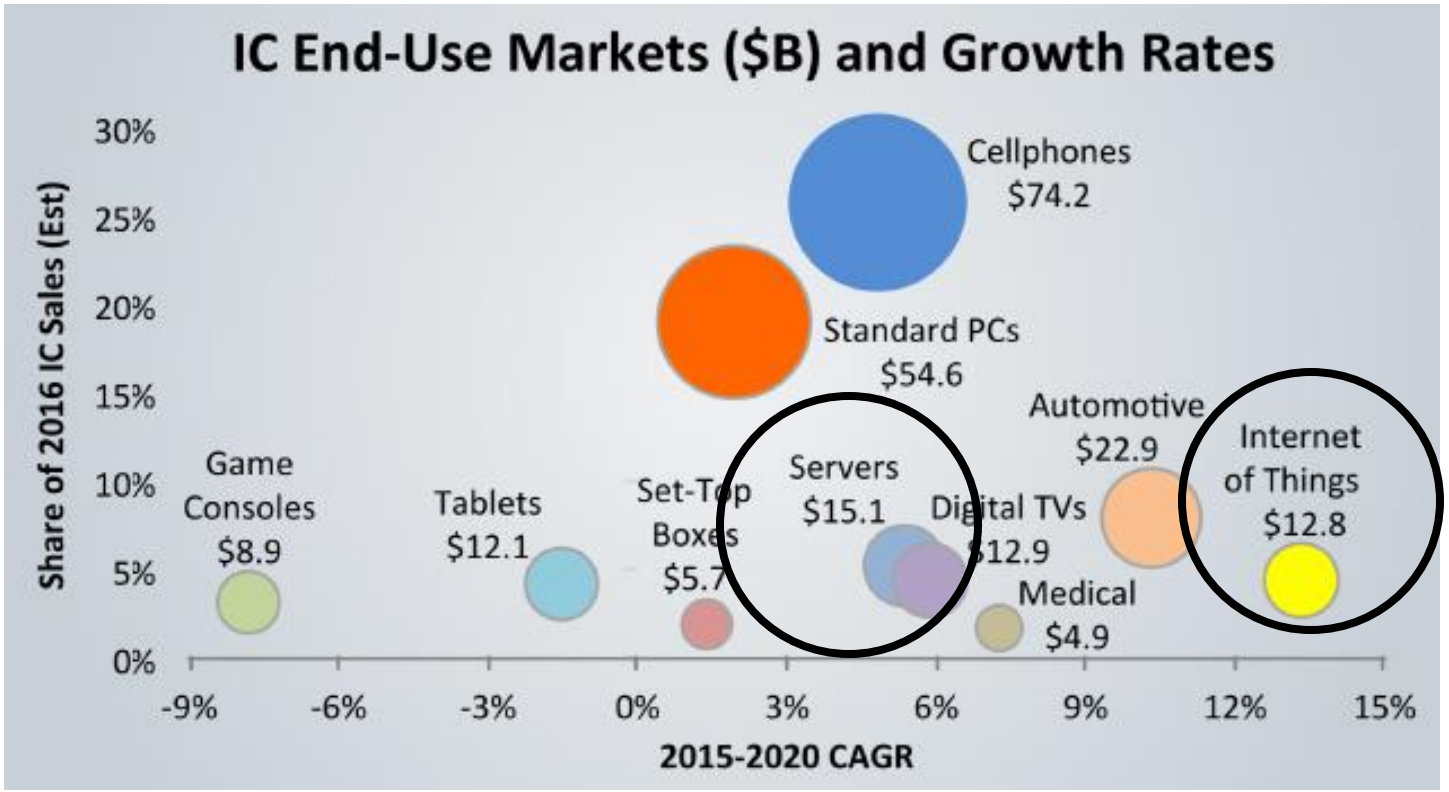
Single view of a situation
- More accurate
- More reliable
- More informative

Radar: https://www.freepng.fr/png-6cga6l/
Satellite: https://www.freepng.fr/png-nw4k8x/
Thermometer: https://www.freepng.fr/png-4z1xb2/
Compas: http://pngimg.com/images/technic/compass
Gyroscope: https://www.transportshaker-wavestone.com/gyropode-gyroroue-hoverboard-de-quoi-parle-t-on/

Context

Traditional IoT architecture

IoT is growing faster than the Cloud

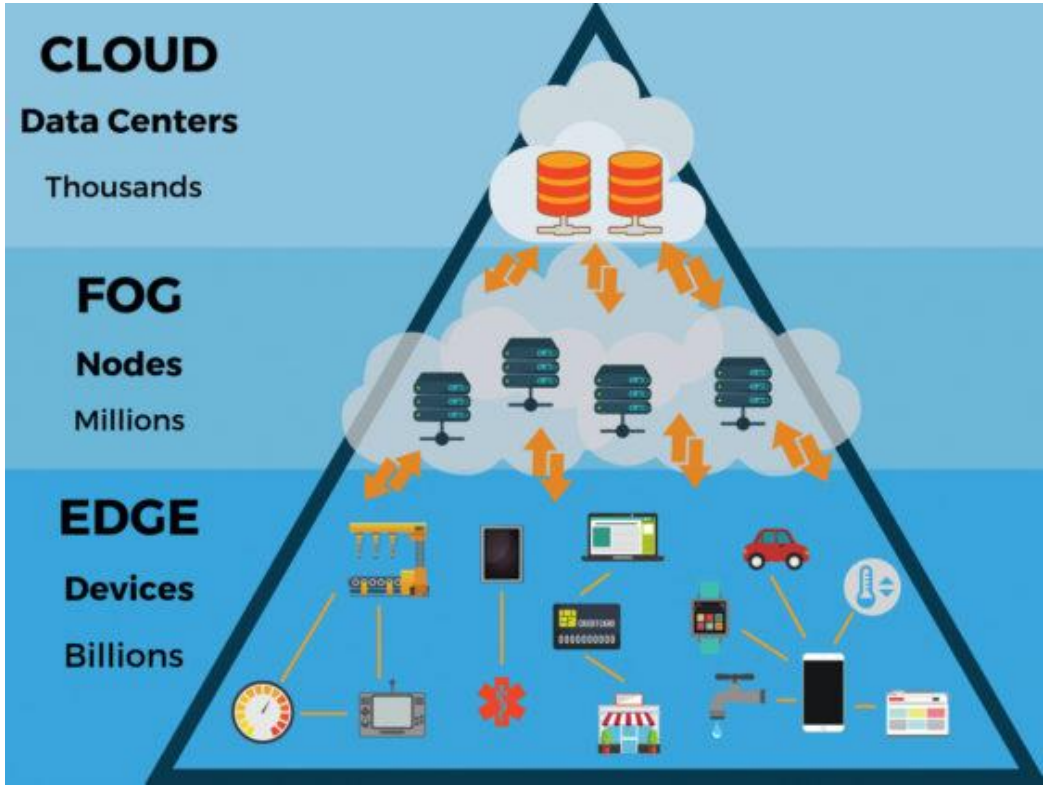


Cloud: 5%
IoT: 13%

Demand for microchips is not in servers

Holger Pirk. 2020. Dark silicon – a currency we do not control.

Cloud and Fog computing



https://iot.electronicsforu.com/content/tech-trends/edge-and-fog-computing-practical-uses/

- Complex infrastructure
- Cost
- Latency
- Unreliable connection

We want to live at the Edge!
Erlang is a very good choice

- **Concurrency**
- **Scalability**
- **Fault tolerance**

Context

Low-cost devices for IoT

Low-cost platforms for IoT

- Close to hardware (datasheet, soldering)
- Low-level language (C)

Complex and error-prone

- Distributed computing?
- Fault tolerance?

Raspberry Pi 4

Arduino Uno

The GRiSP platform

- **Low-cost** embedded system
- Boots into an **Erlang** VM
- **Pmod** connectors and drivers "plug and play"

Pmod NAV

Pmod MAXSONAR

- Fault tolerance?
- Sensor fusion model?

We need a sensor fusion framework

The Hera framework

Purpose

The Hera framework

> Hera provides an **efficient** and **user-friendly** framework to enable **sensor fusion** experimentation by a **large public**.

The Hera framework

What does Hera do?

- Provides a sensor fusion engine
- Takes care of the hardware (GRiSP's drivers)
- Handles failures

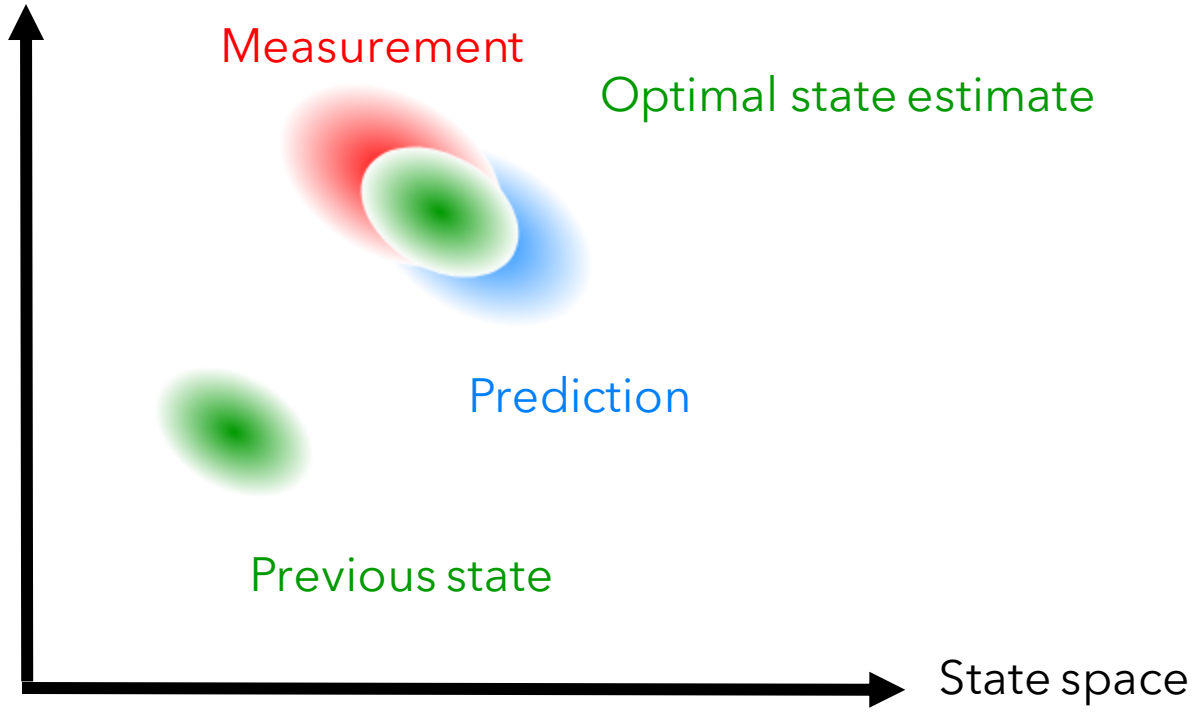
What does the user need to do?

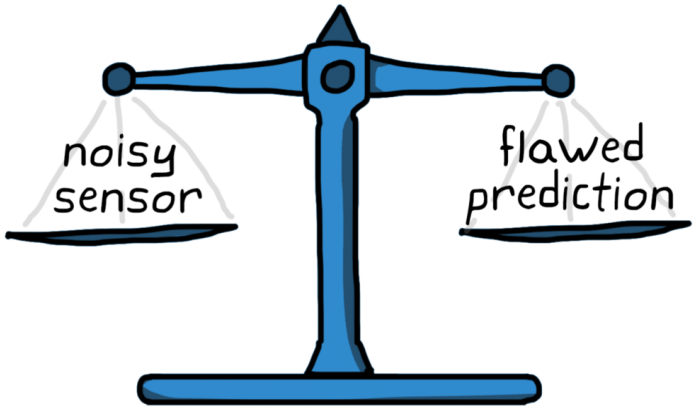
- Provide the sensor fusion parameters
- Bootstrap the cluster of nodes

The Hera framework

Sensor fusion engine

Sensor fusion engine: The Kalman filter





https://engineeringmedia.com/controlblog/the-kalman-filter

Sensor fusion engine: The Kalman filter

- **Well-established** technique in Wireless Sensor Networks
- For the physical model, we just need **differentiable equations**
- **Many variations** for optimization
- Allows **asynchronous** measurements

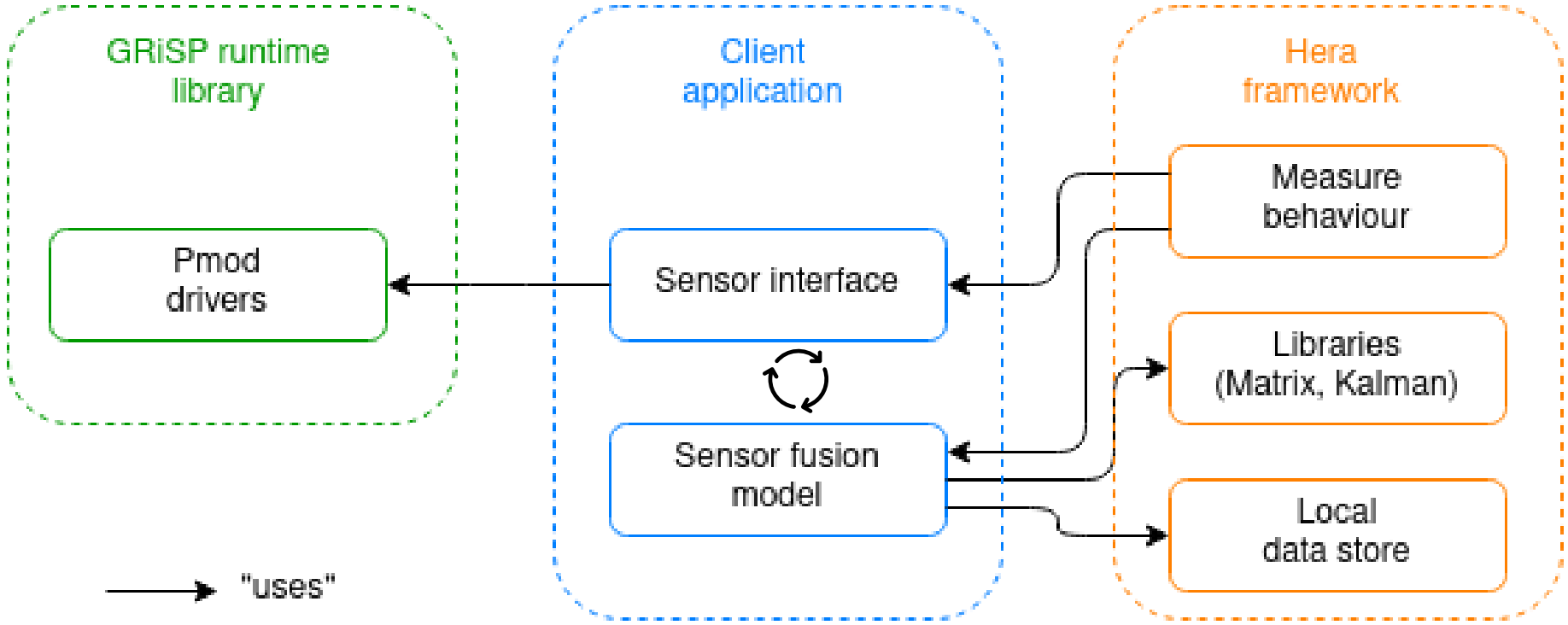
- Defining the physical model is **not trivial**, but many examples in the literature

The Hera framework

Software architecture

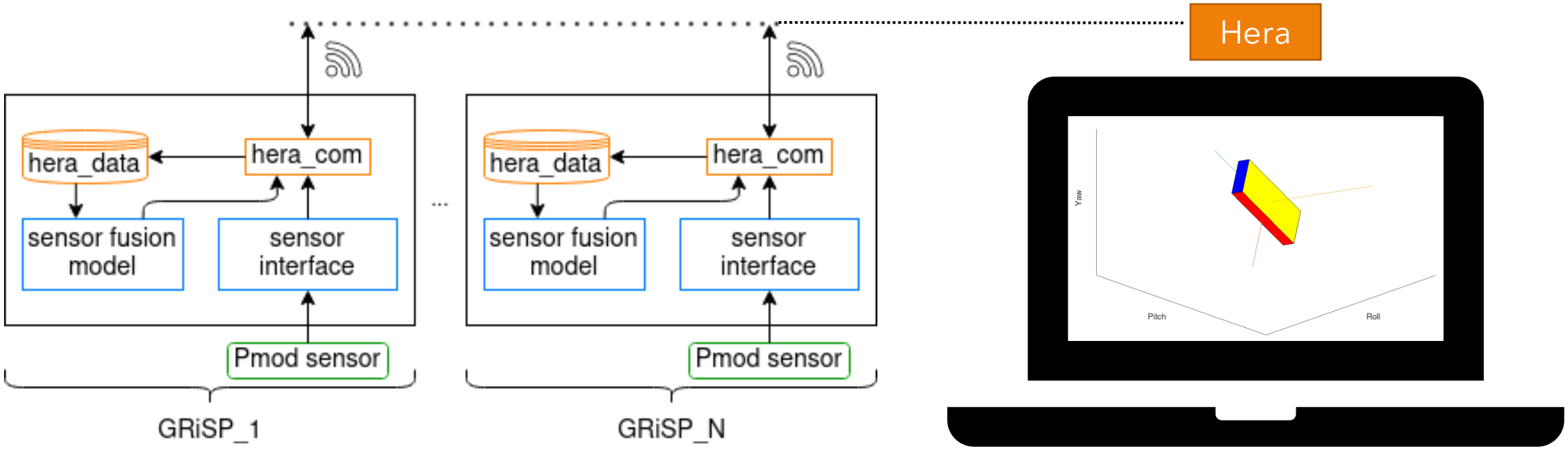
Software architecture

Modular



Software architecture

Soft real-time



Fault tolerance

Fault tolerance is important in IoT

Multiple points of failure

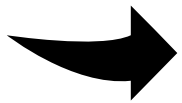
- Hardware: power, network
- Software: driver, user code

Sensor fusion is hard enough

- The user should not do defensive programming
- Nor error handling

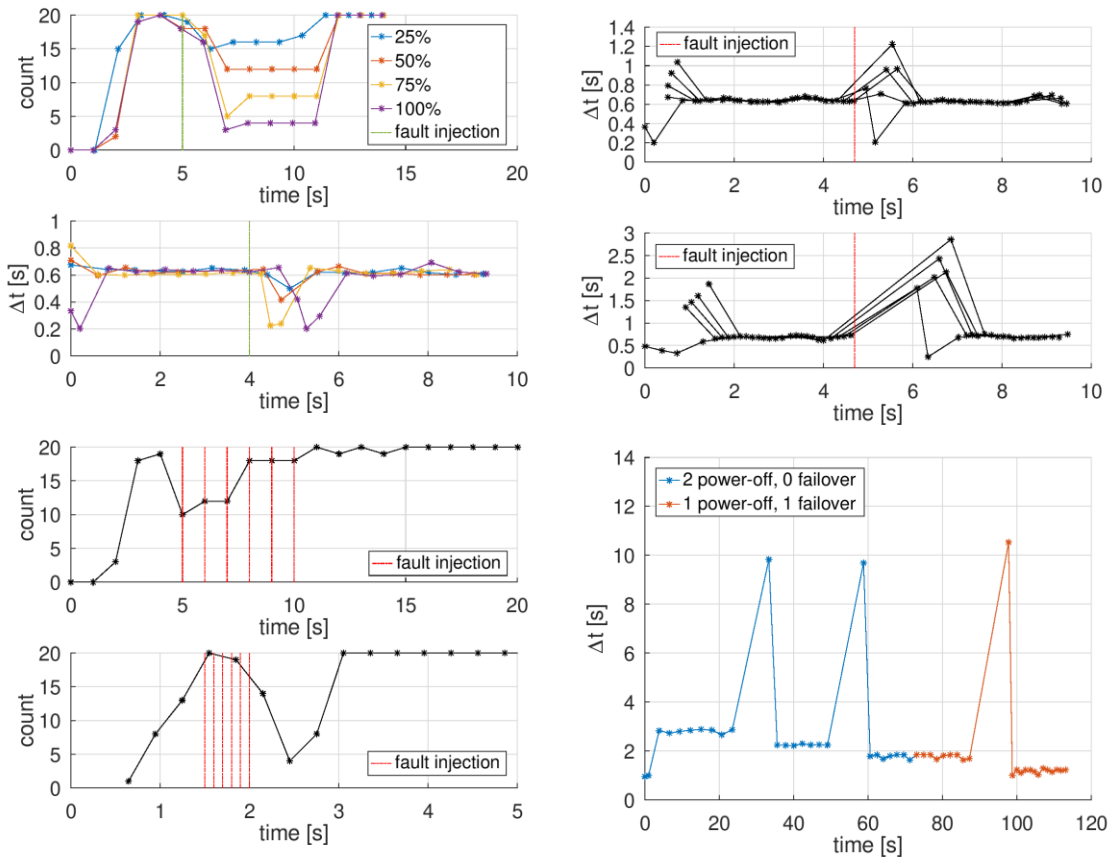
How does Hera achieve fault tolerance ?

- **Asynchronous model**: computes with the available data (not blocking)
- **Dynamic system**: nodes and processes can join/leave at any time
- **Supervision**: when a process fails it is restarted
- **Hardware redundancy**: sensors and multiple nodes compute the same thing



Sensor fusion as long as one GRiSP board is alive

Hera fault tolerance

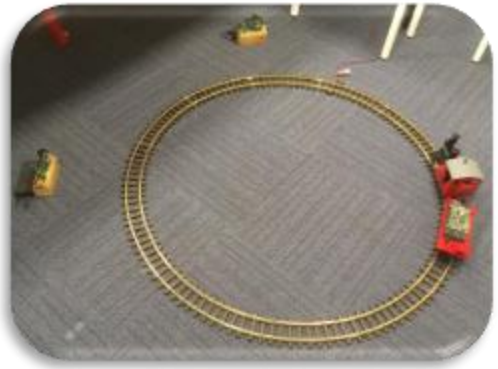


Proved by fault injection

Disable parts of the system and observe the reaction

Examples of sensor fusion with Hera

Position and velocity of a model train with 3 GRiSP-Base boards

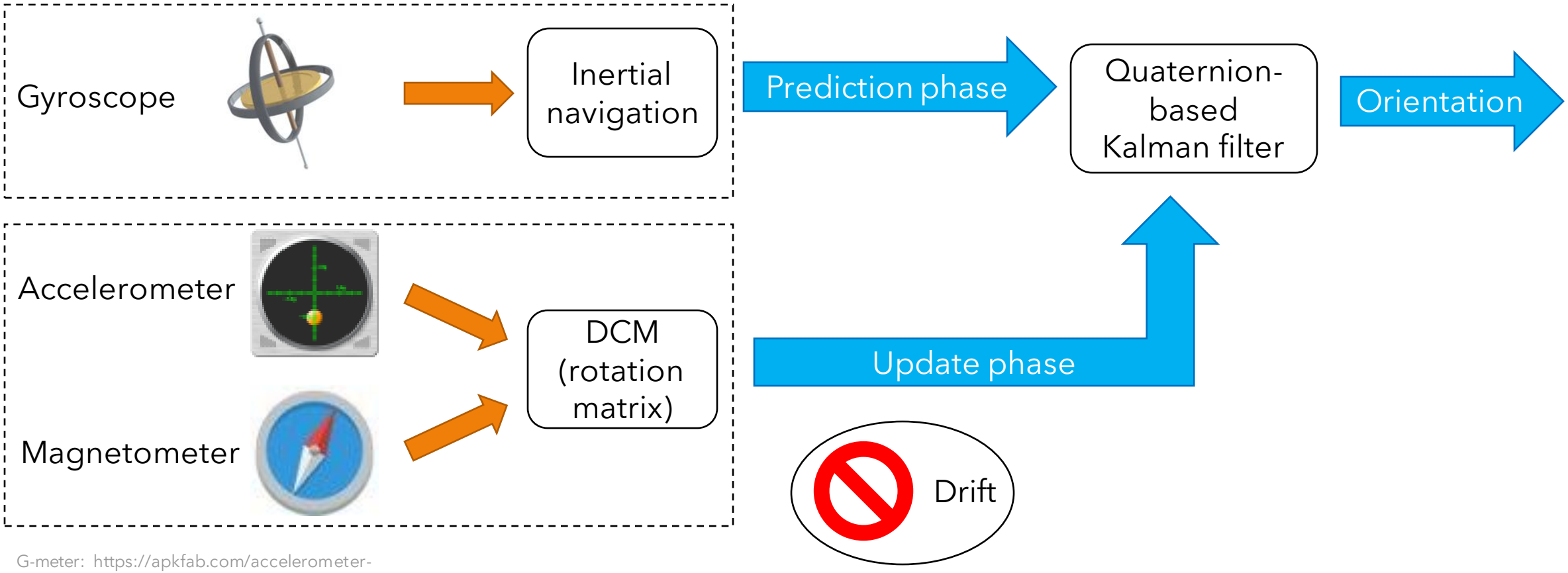


```
measure({T0, X0, P0}) ->
N = [Data || {_,_,Ts,Data} <- hera_data:get(nav), T0 < Ts],
M = [Data || {_,_,Ts,Data} <- hera_data:get(mag), T0 < Ts],
S = [Data || {_,_,Ts,Data} <- hera_data:get(sonar), T0 < Ts],
T1 = hera:timestamp(),
if
length(N) + length(M) + length(S) == 0 ->
  {undefined, {T0, X0, P0}};
true ->
  Dt = (T1 - T0)/1000,
  F = fun([_, _, [O], [W], [Radius]]) -> [
    [Radius*math:cos(O)],
    [Radius*math:sin(O)],
    [O+W*Dt],
    [W],
    [Radius]
  ] end,
  Jf = fun([_, _, [O], _, [Radius]]) -> [
    [0,0,-Radius*math:sin(O),0,math:cos(O)],
    [0,0,Radius*math:cos(O),0,math:sin(O)],
    [0,0,1,Dt,0],
    [0,0,0,1,0],
    [0,0,0,0,1]
  ] end,
  Q = mat:zeros(5,5),
  H = fun([[X], [Y], [O], [W], [Radius]]) ->
    [[Radius*W*W] || _ <- N] ++
    [[W] || _ <- N] ++
    [[shortest_path(-OZ, O)] || [OZ] <- M] ++
    [[dist({X,Y},{Px,Py})] || [_,Px,Py] <- S]
  end,
  Jh = fun([[X], [Y], _, [W], [Radius]]) ->
    [[0,0,0,2*Radius*W,W*W] || _ <- N] ++
    [[0,0,0,1,0] || _ <- N] ++
    [[0,0,1,0,0] || _ <- M] ++
    [[dhdx({X,Y},{Px,Py}),dhdx({Y,X},{Py,Px}),0,0,0]
      || [_,Px,Py] <- S]
  end,
  Z = [[-Ay] || [Ay,_] <- N] ++
    [[-Gz] || [_,Gz] <- N] ++
    [[-O] || [O] <- M] ++
    [[Range] || [Range,_,_] <- S],
  R = mat:diag(
    [?VAR_A || _ <- N] ++
    [?VAR_G || _ <- N] ++
    [?VAR_M || _ <- M] ++
    [?VAR_S || _ <- S]
  ),
  {X1,P1}=kalman:ekf({X0,P0},{F,Jf},{H,Jh},Q,R,Z),
  {ok, lists:append(X1), {T1, X1, P1}}
end.
```

- Model encoded in Hera:
  - Accelerometer
  - Gyroscope
  - Magnetometer
  - Sonars x2

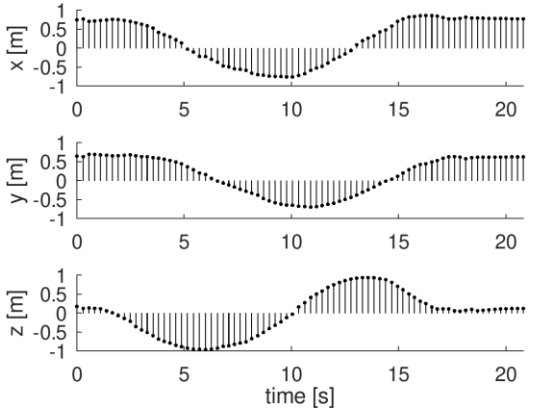
➔ Easy-to-use !

Attitude and heading reference system (AHRS)



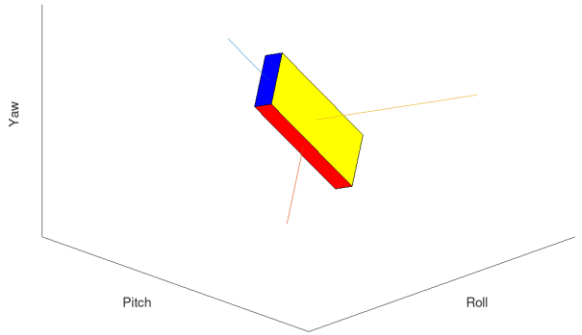
G-meter: https://apkfab.com/accelerometer-sensor/com.vicdmitrienko.log.accelerometer

Attitude and heading reference system (AHRS)



- **Orientation in real-time computed on a single GRiSP-Base board!**
- Pushes GRiSP-Base board, drivers and Erlang numeric computation to their limit!

- Video link: http://www.info.ucl.ac.be/~pvr/Hera-demo.mp4



Conclusion and future work

Conclusion

- Hera gives **satisfactory results** for simple sensor fusion
- **No Cloud**: simpler, cheaper, more reliable
- **High-level** with focus on sensor fusion model: no soldering, no datasheet, no C
- **Fault-tolerant**, **Easy-to-use**, **Low-cost, Open-source**

→ Ideal for **education** and **prototyping**

Hera is the best existing Kalman filter-based sensor fusion framework for low-cost prototyping

Future work

Performance improvements

- **GRiSP 2: 10x to 20x** faster
- **Improved drivers**
- **New matrix library: 10x to 100x** faster

  Tanguy Losseau. 2021. Concurrent Matrix and Vector Functions for Erlang. Master's thesis. UCLouvain.

End of 2021

Possible experimentations

- Machine learning with Hera (e.g. motion recognition)
  ML and KF have different purpose, but are complementary!
- Controlling physical devices
- Targeting rugged terrains

The IoT with Erlang

- Given the **success** we had with Erlang in our research, we **encourage** more work in **IoT with Erlang**

- GRiSP: https://www.grisp.org/
- Hera: https://github.com/sebkm/hera
- Demo application: https://github.com/sebkm/hera