Lax: Driver Interfaces

for Approximate Sensor Device Access

Phillip **Stanley-Marbell** and Martin **Rinard**
psm@mit.edu

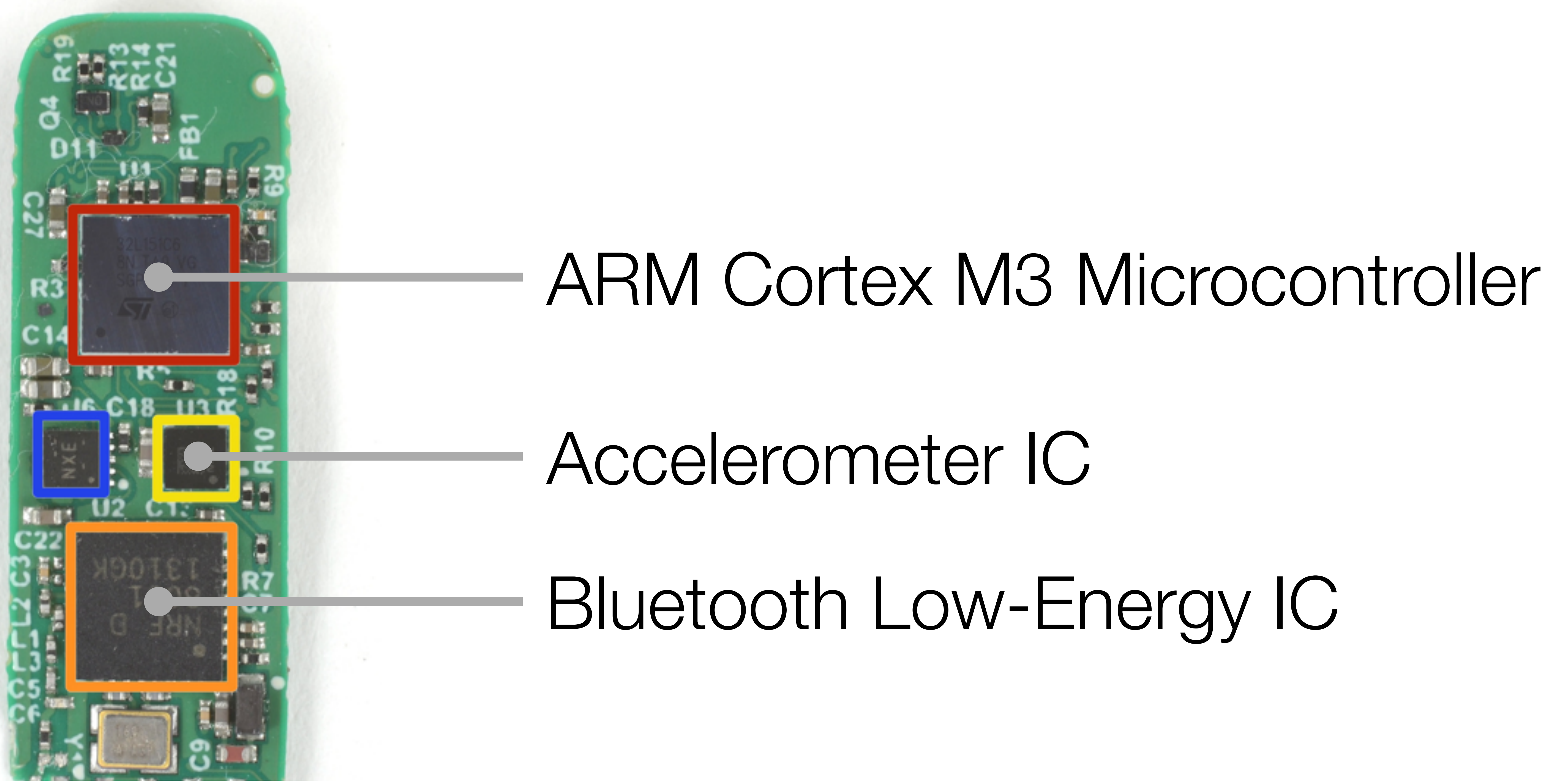
Massachusetts Institute of Technology

HotOS XV, Kartause Ittingen, Switzerland.

Lax: Driver Interfaces

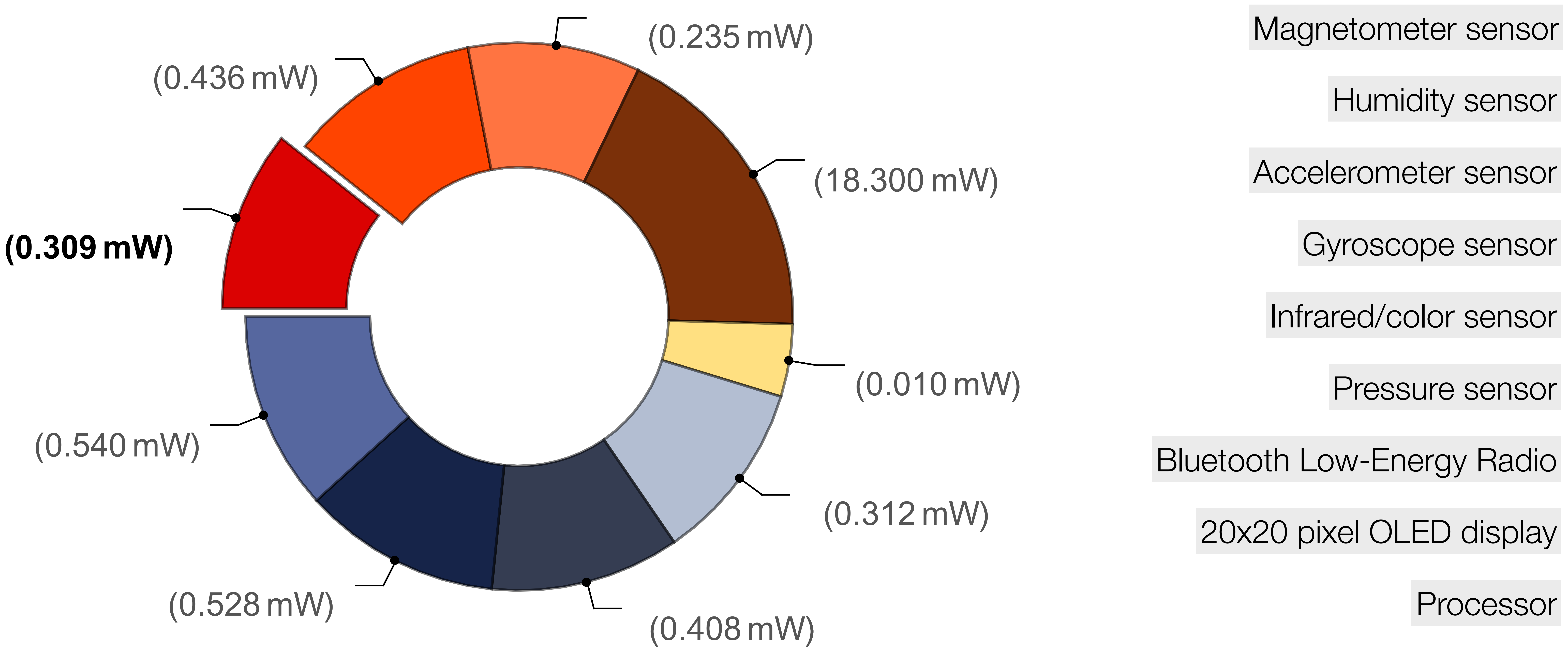
for Approximate Sensor Device Access

(Source: Fitbit)



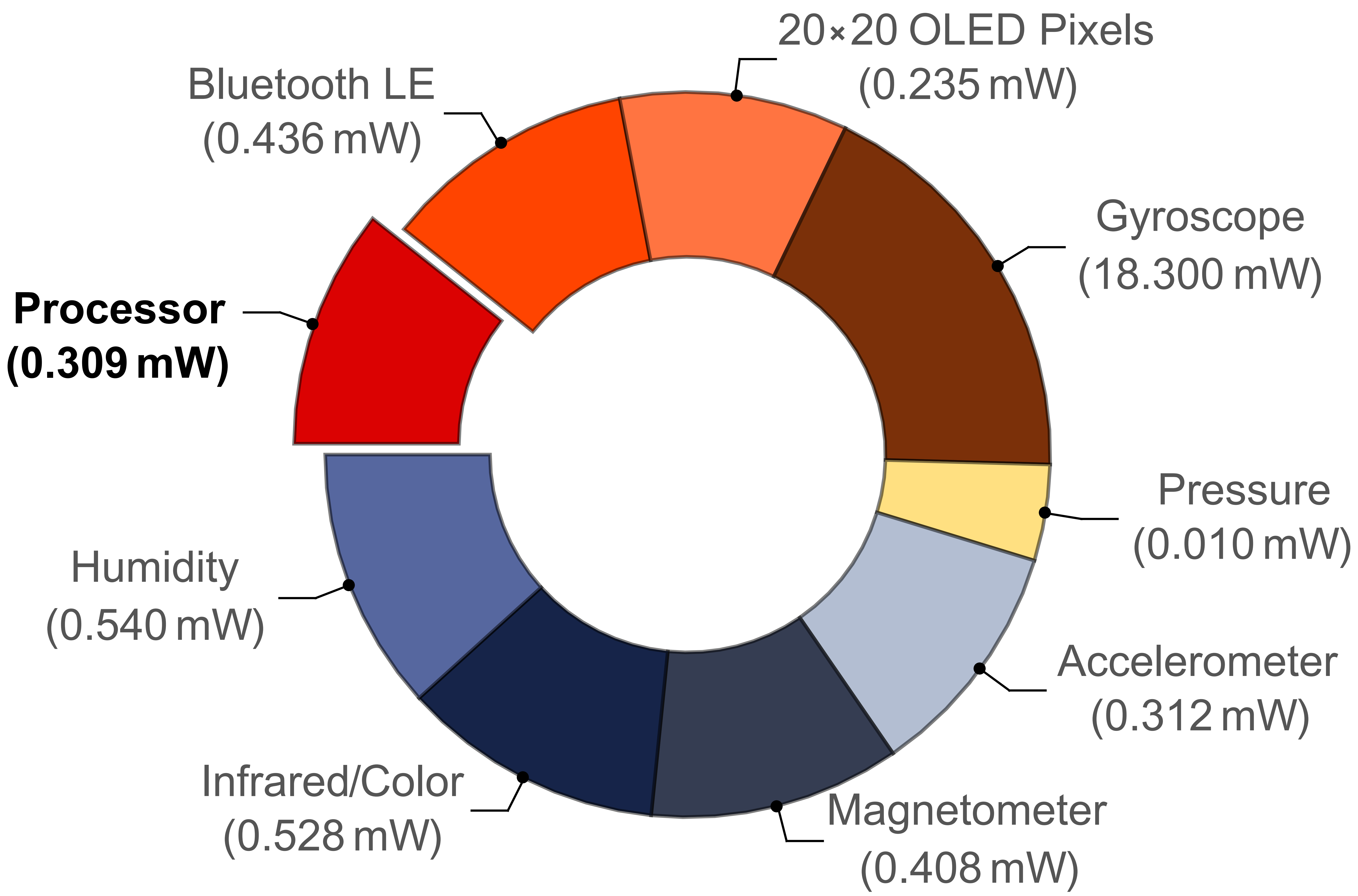
(Source: ifitxit.com)

Power Breakdown for a Wearable Sensor System



(Note: sectors scaled logarithmically due to large range of values)

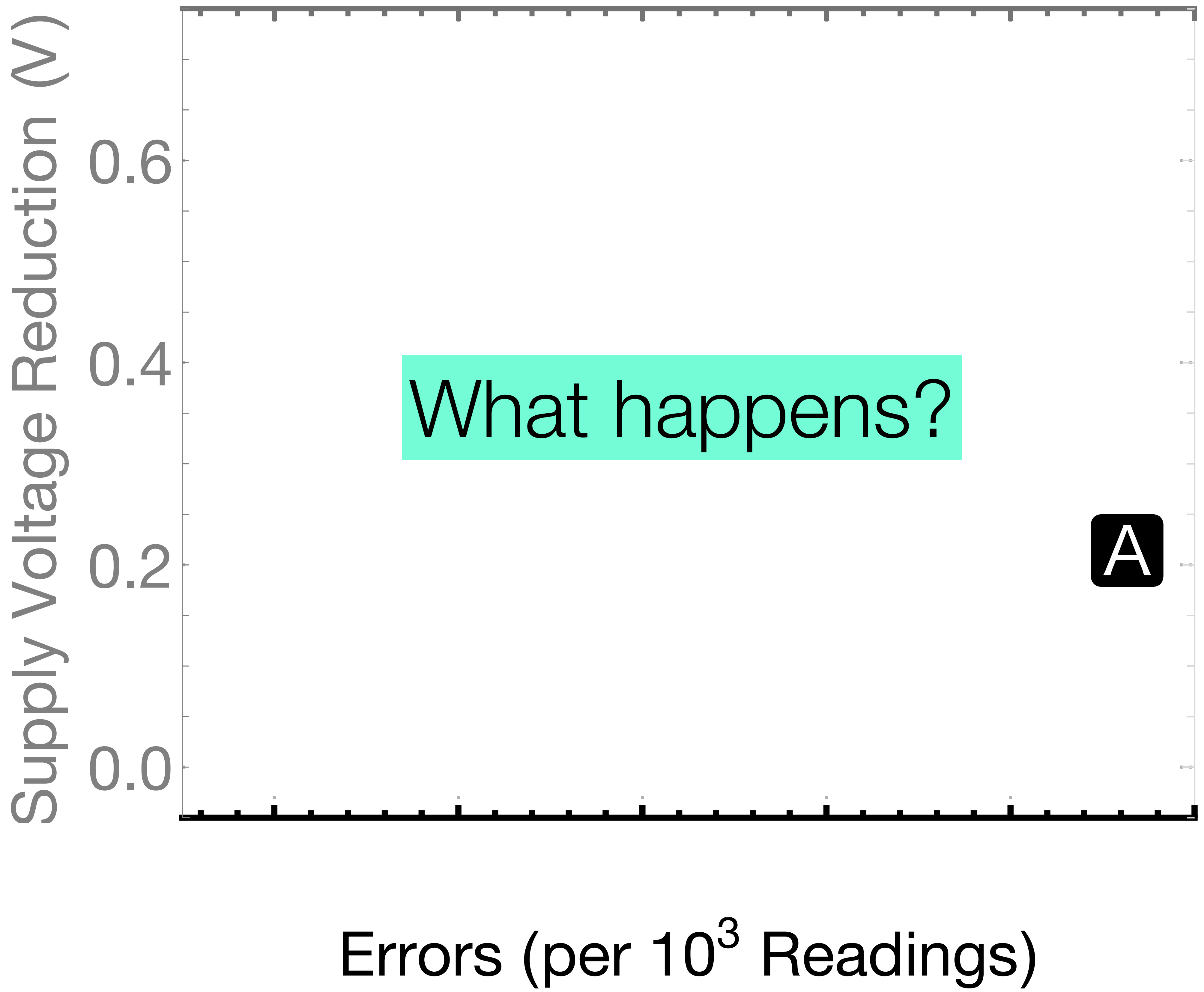
Sensors/Displays Dominate Power Dissipation in Wearables

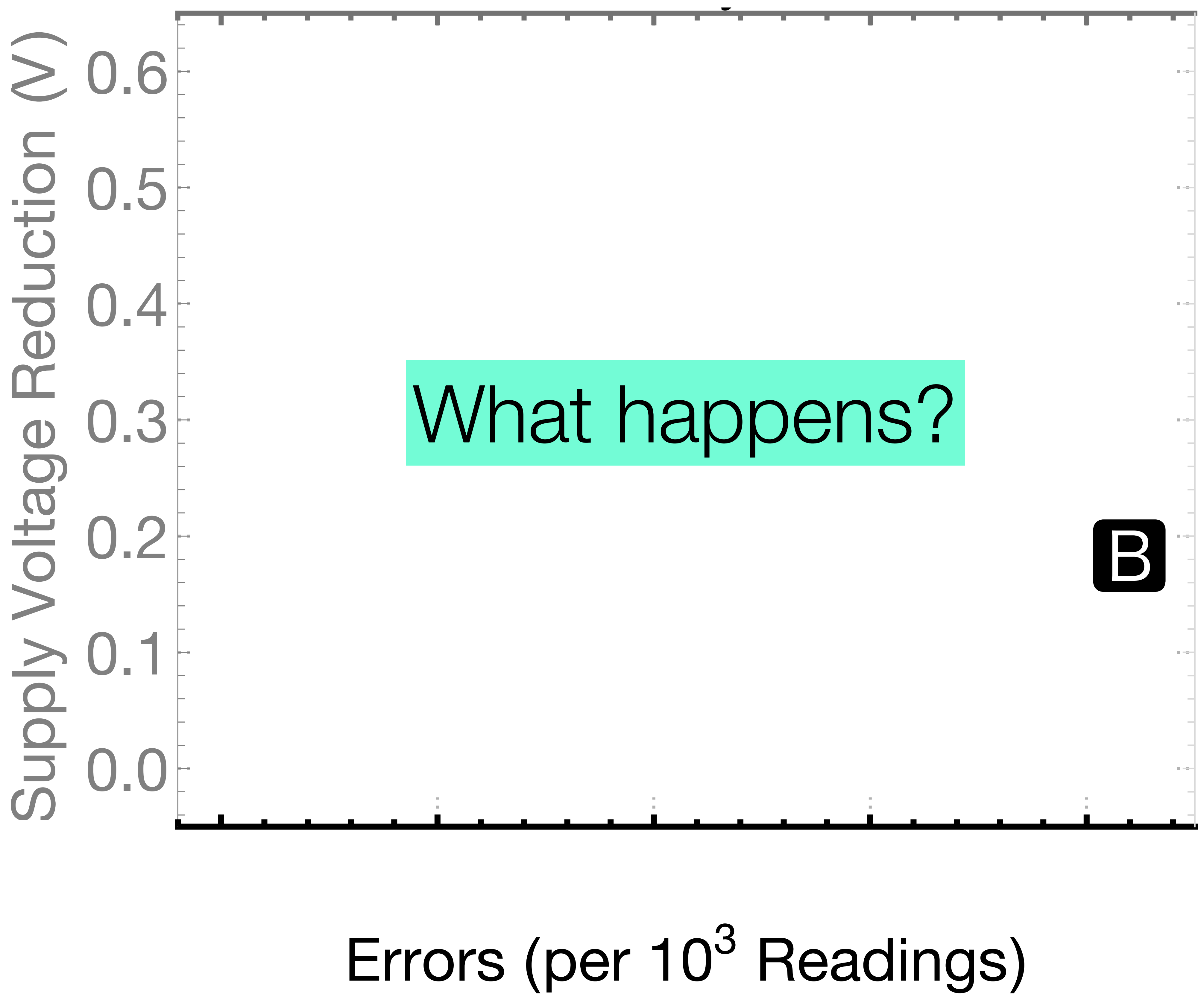


(Note: sectors scaled logarithmically due to large range of values)

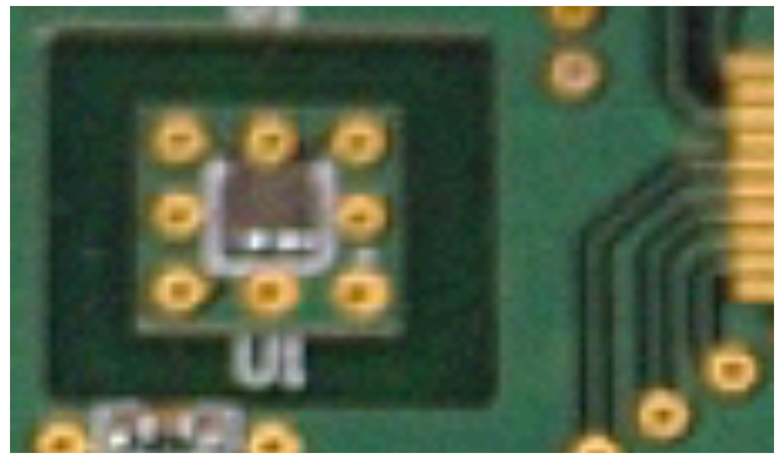
To improve system-wide energy-efficiency, **focus on the dominant fraction**

Reducing Power Dissipation of Sensor Devices



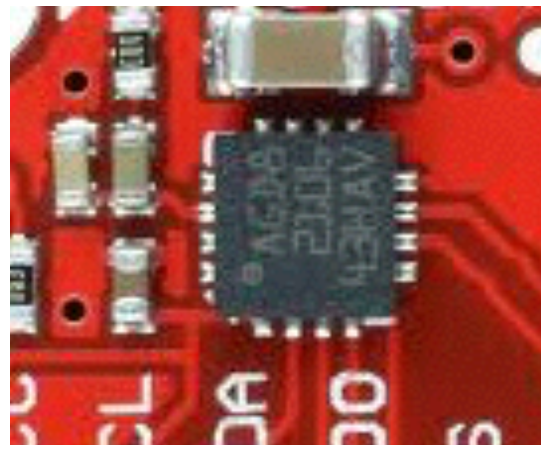


A



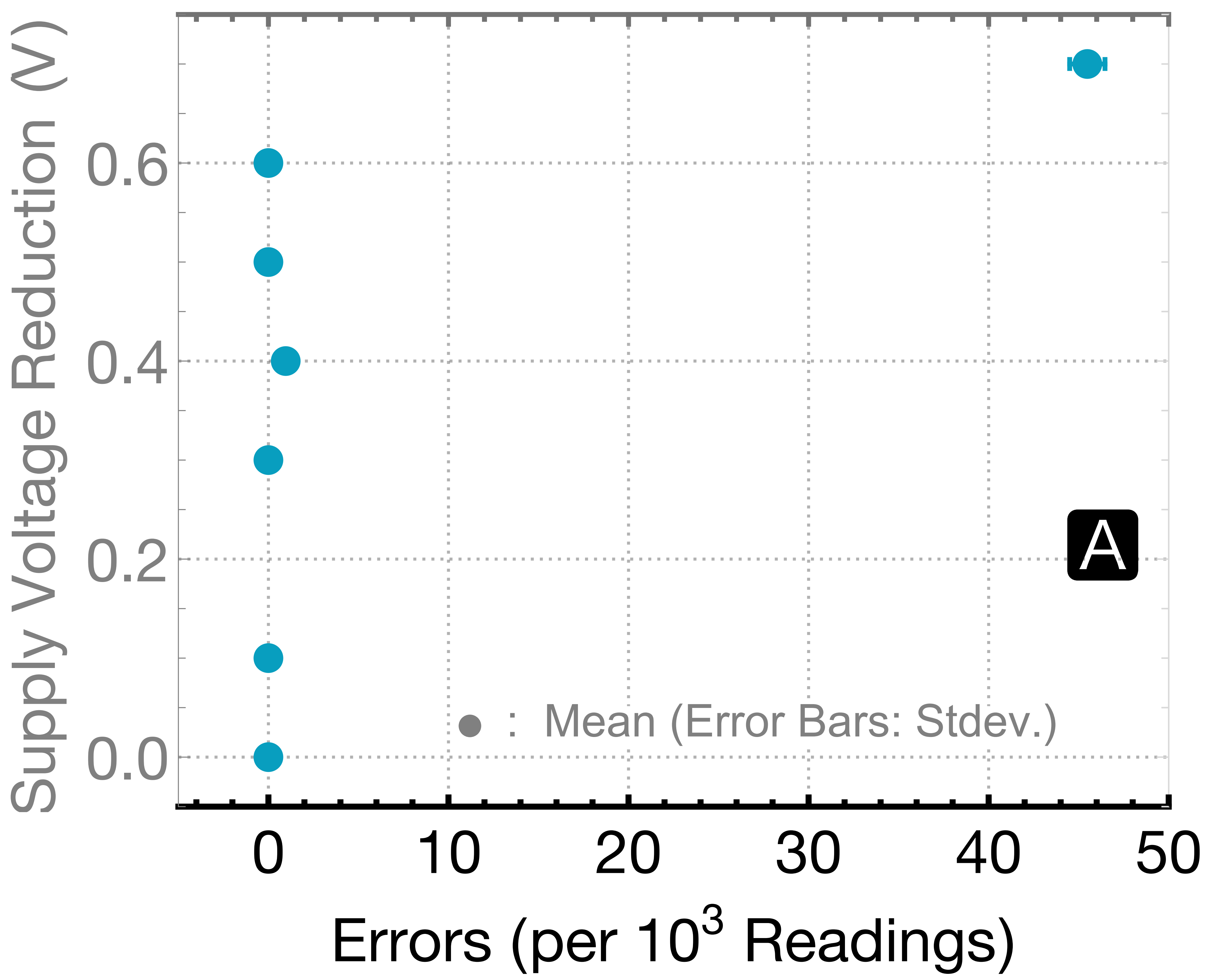
TI TMP006
IR Temperature
Sensor

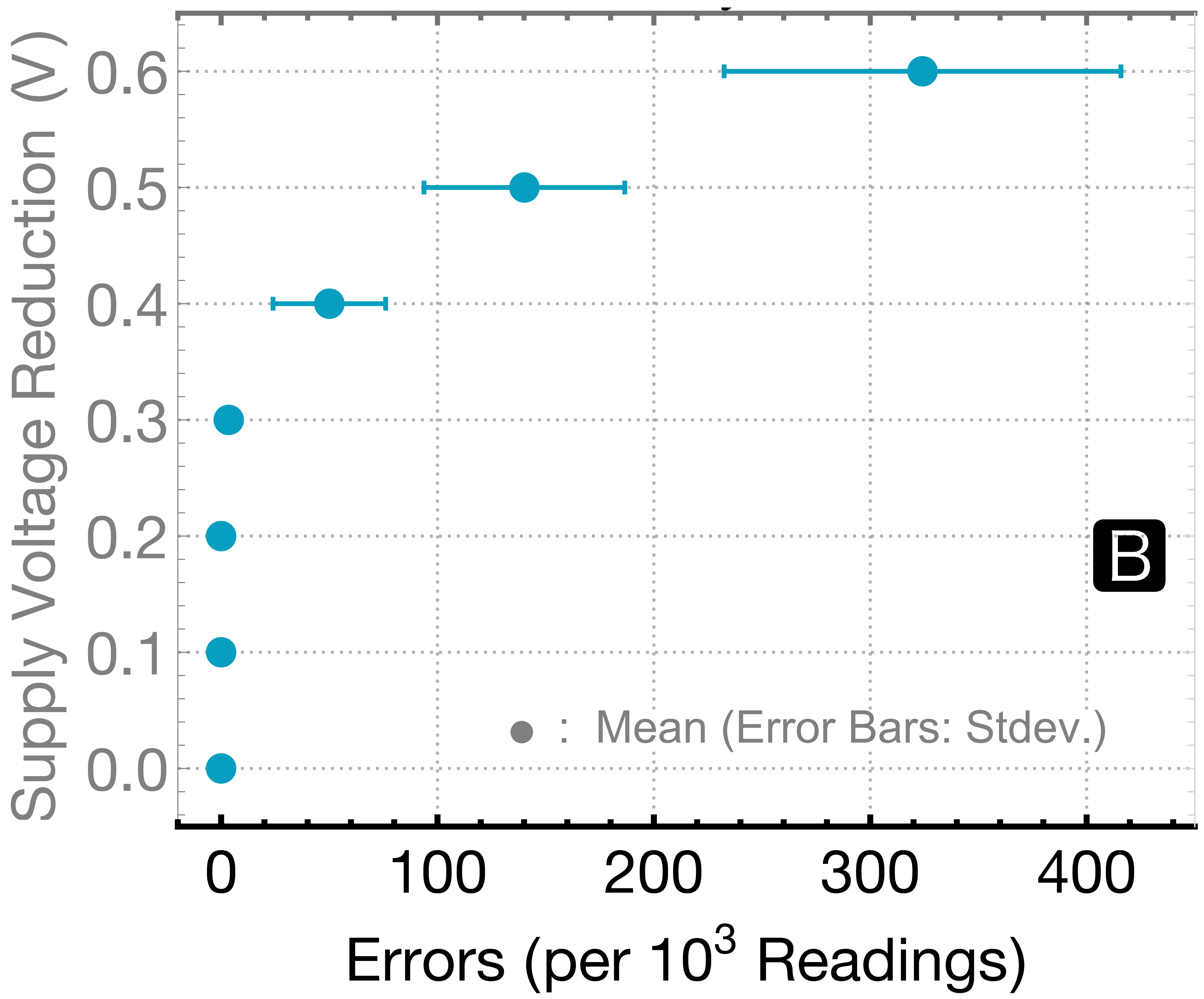
B



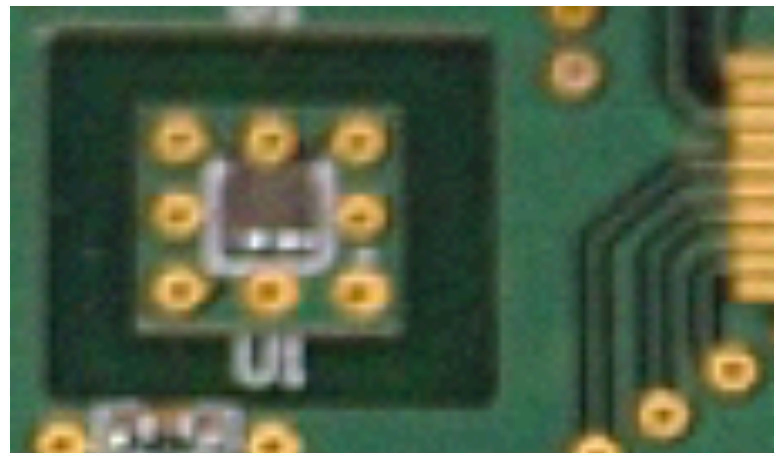
ST L3G4200D
Gyroscope
Sensor

Reducing Power Dissipation of Sensor Devices



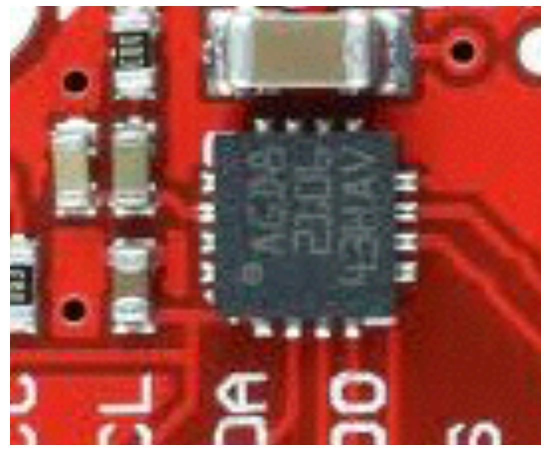


A



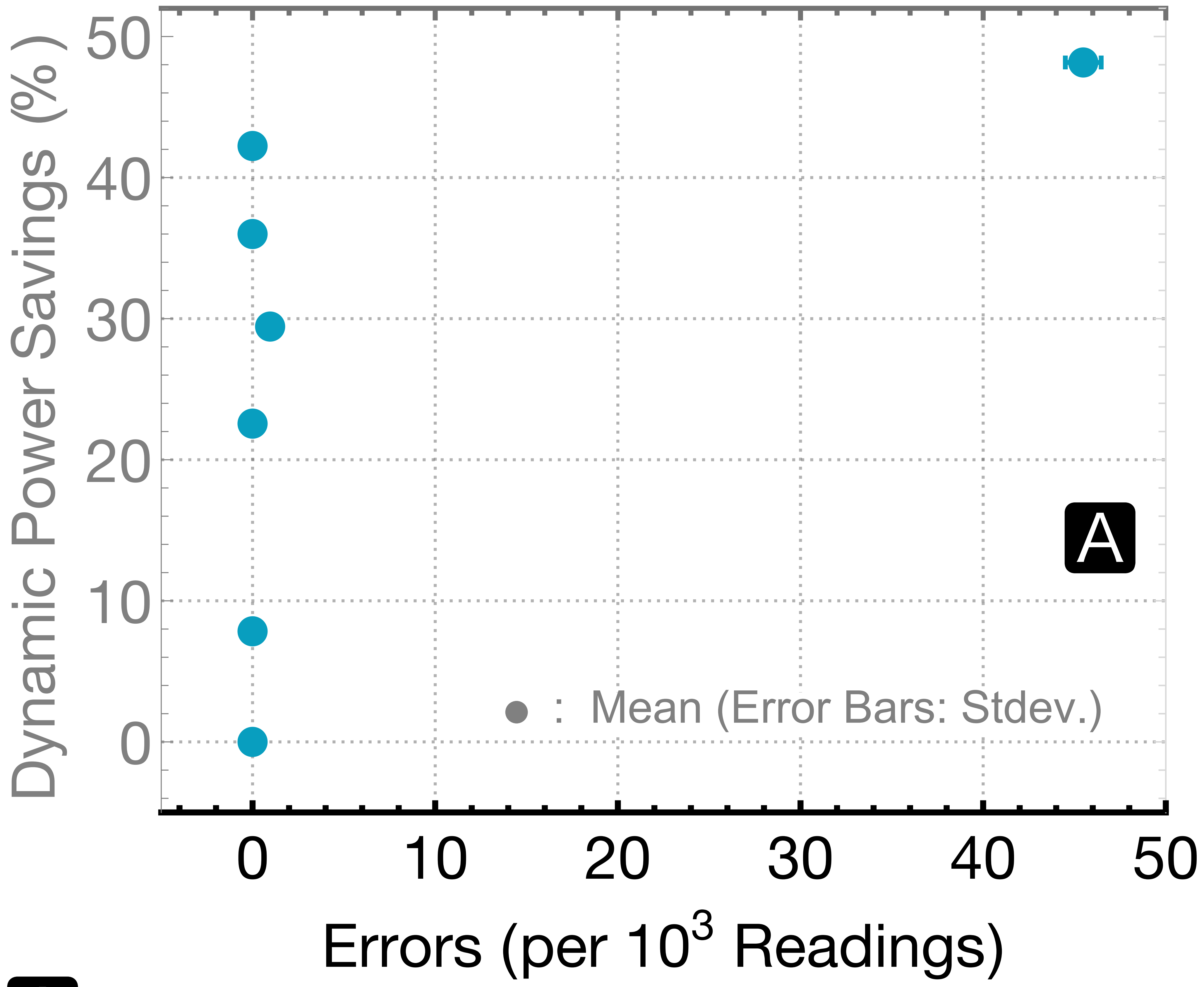
TI TMP006
IR Temperature
Sensor

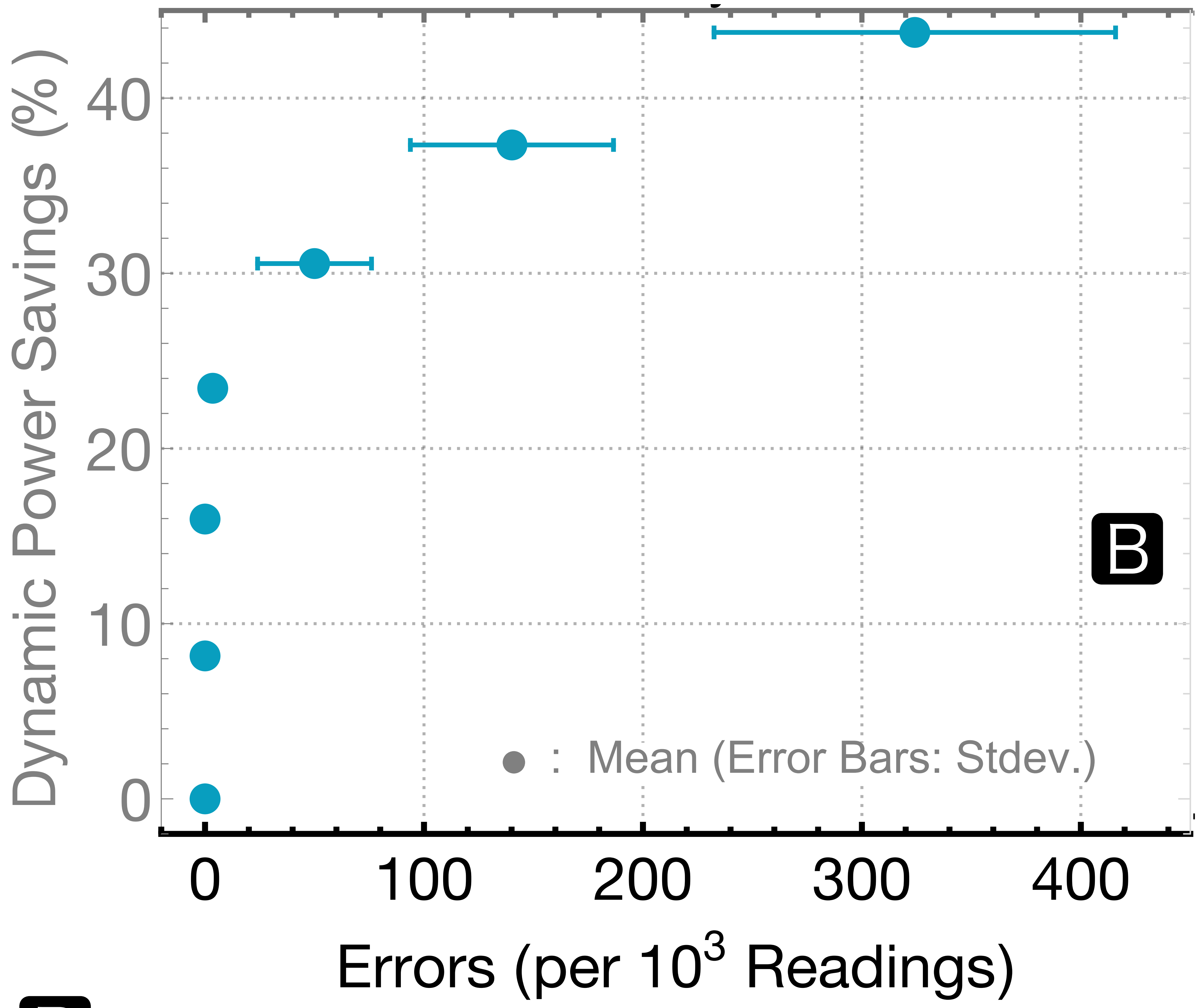
B



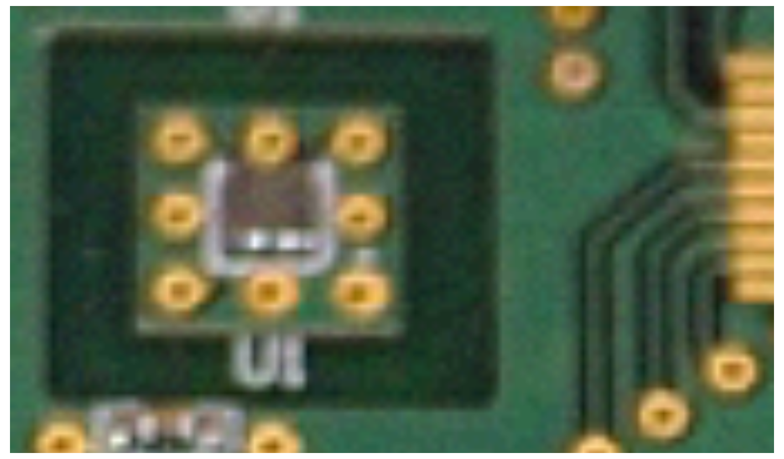
ST L3G4200D
Gyroscope
Sensor

Reducing Power Dissipation of Sensor Devices



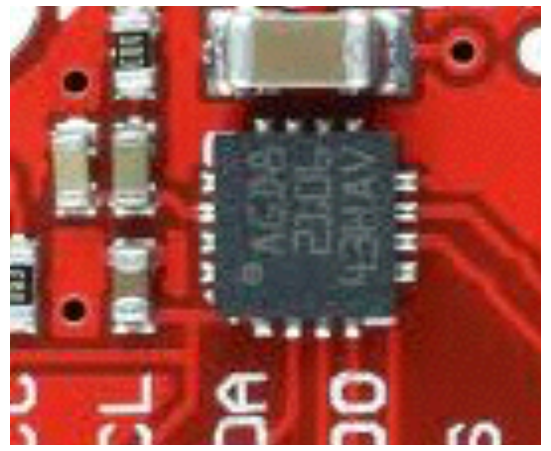


A



TI TMP006
IR Temperature
Sensor

B



ST L3G4200D
Gyroscope
Sensor

How Can We Exploit What We Have Just Seen?

(access error rate versus power dissipation tradeoffs)

We can trade tolerance to errors for performance and power of sensors

Our objective: provide a **principled way to tradeoff error for energy** in systems

```
sample = sensor_read(sensorName, tolerableError);
```

How Can We Exploit What We Have Just Seen?

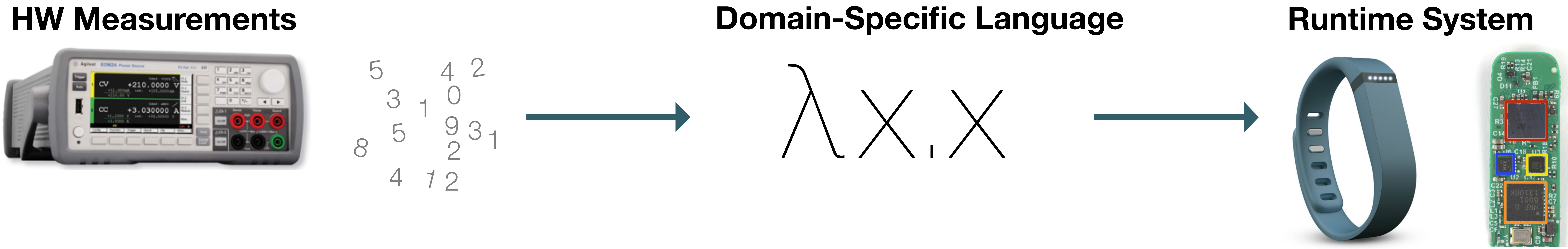
(access error rate versus power dissipation tradeoffs)

We can trade tolerance to errors for performance and power of sensors

Our objective: provide a **principled way to tradeoff error for energy** in systems

```
sample = sensor_read(sensorName);
```

How Can We Exploit this Property in Real Systems?



(Sources: Fitbit and ifixit.com)

How Can We Exploit this Property in Real Systems?

❶ Characterize **HW error properties** (sometimes possible from datasheets)

HW Measurements

5 4 2 3 1 0 5 9 3 1 8 2 4 1 2

❶ →

Domain-Specific Language

$\lambda x.x$

→

Runtime System

(Sources: Fitbit and ifixit.com)

How Can We Exploit this Property in Real Systems?

❶ Characterize **HW error properties** (sometimes possible from datasheets)

❷ Capture **permitted error behavior** in Lax language (per application/domain)

HW Measurements

5 4 2 3 1 0 5 9 3 1 8 2 4 1 2

❷ **Domain-Specific Language**

$\lambda x.x$

Runtime System

(Sources: Fitbit and ifixit.com)

How Can We Exploit this Property in Real Systems?

❶ Characterize **HW error properties** (sometimes possible from datasheets)

❷ Capture **permitted error behavior** in Lax language (per application/domain)

❸ **Tools** to exploit HW properties modulo permitted behavior, per application

HW Measurements

Domain-Specific Language

λx.x

❸

Runtime System

(Sources: Fitbit and ifixit.com)

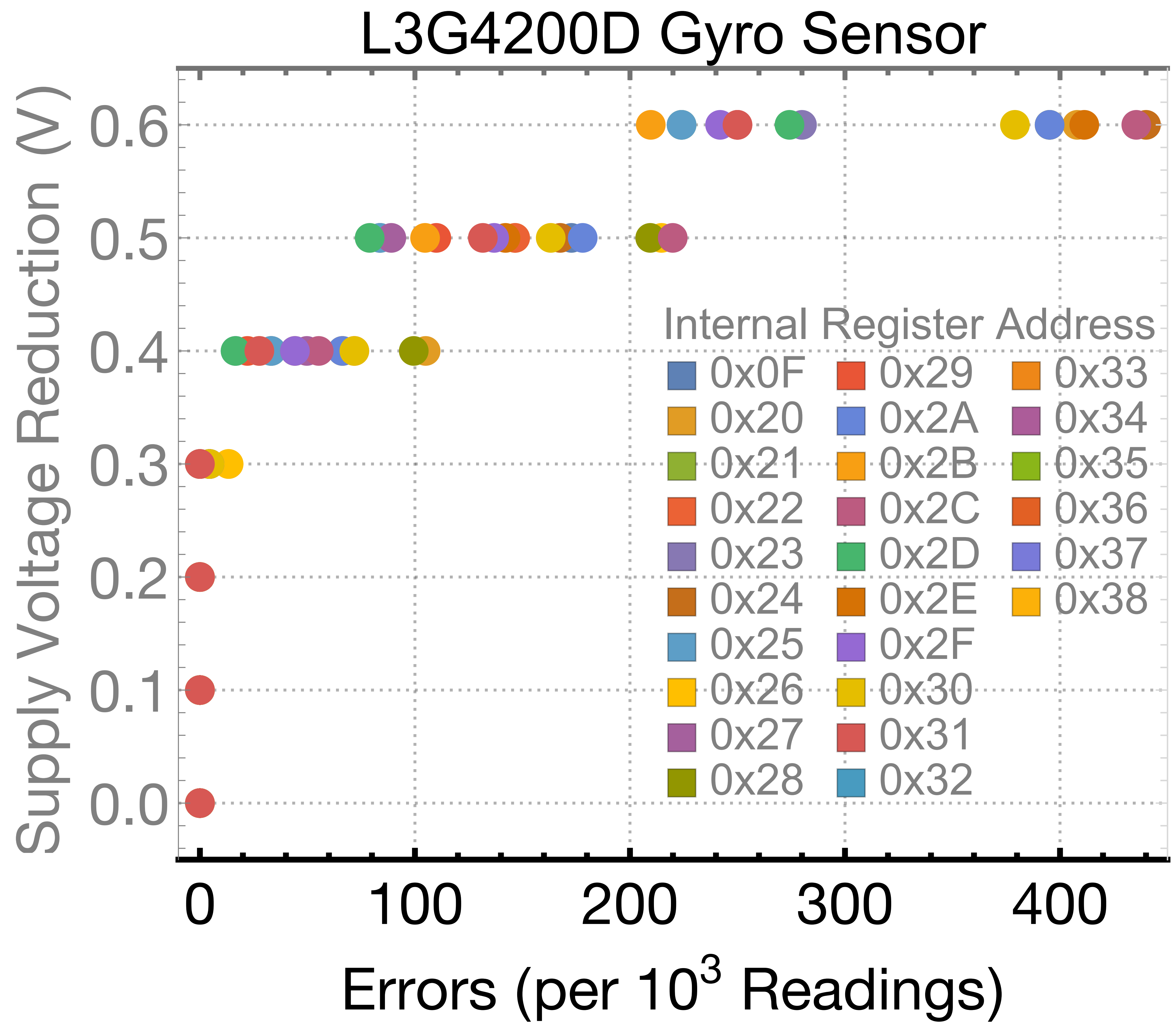
How Can We Exploit this Property in Real Systems?

1. Characterize **HW error properties** (sometimes possible from datasheets)
2. Capture **permitted error behavior** in Lax language (per application/domain)
3. **Tools** to exploit HW properties modulo permitted behavior, per application
4. **Runtime system** to use information generated from tools to control HW

(Sources: Fitbit and ifixit.com)

An Example

1



2

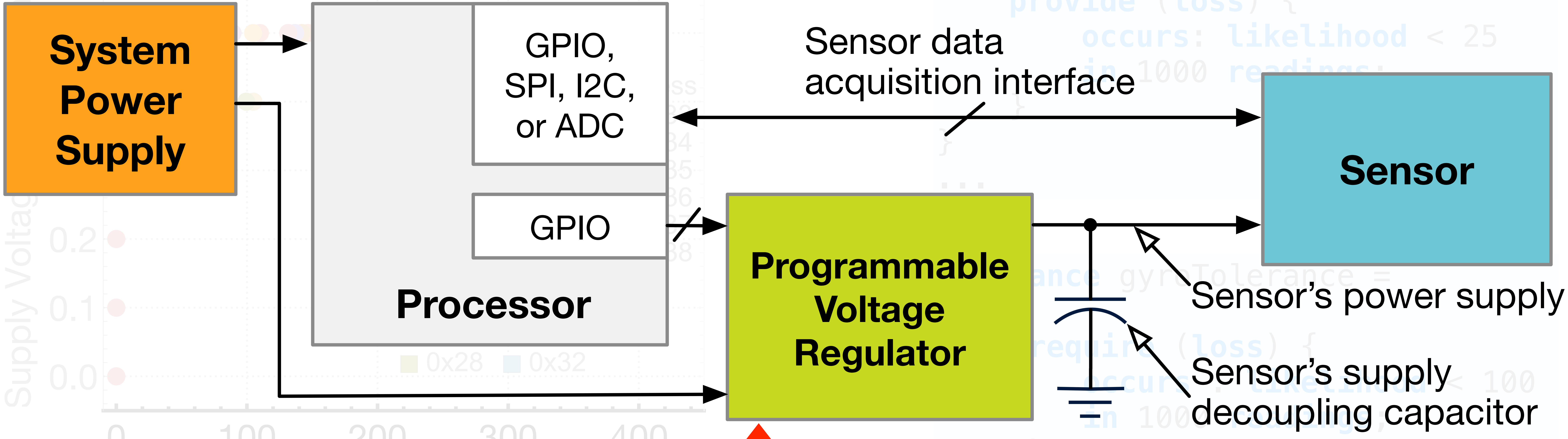
```
sensor gyroSourceXaxis @ 1.6V =
{
    provide (loss) {
        occurs: likelihood < 25
        in 1000 readings;
    }
}
...
```

```
tolerance gyroTolerance =
{
    require (loss) {
        occurs : likelihood < 100
        in 1000 readings;
    }
}
```

3 **Example Usage:**

```
/*  Use Lax to achieve lowest power for required accuracy.  */
sampleC = lax_sensor_read(gyroSourceX, gyroTolerance);
```

An Example

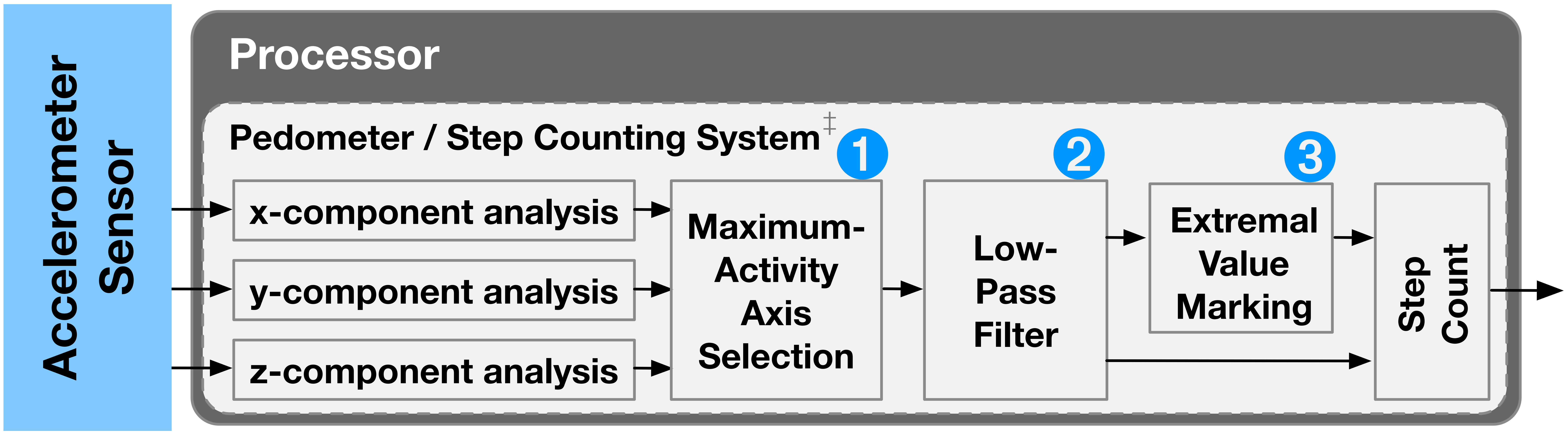


Needs minimal hardware support

Example Usage:

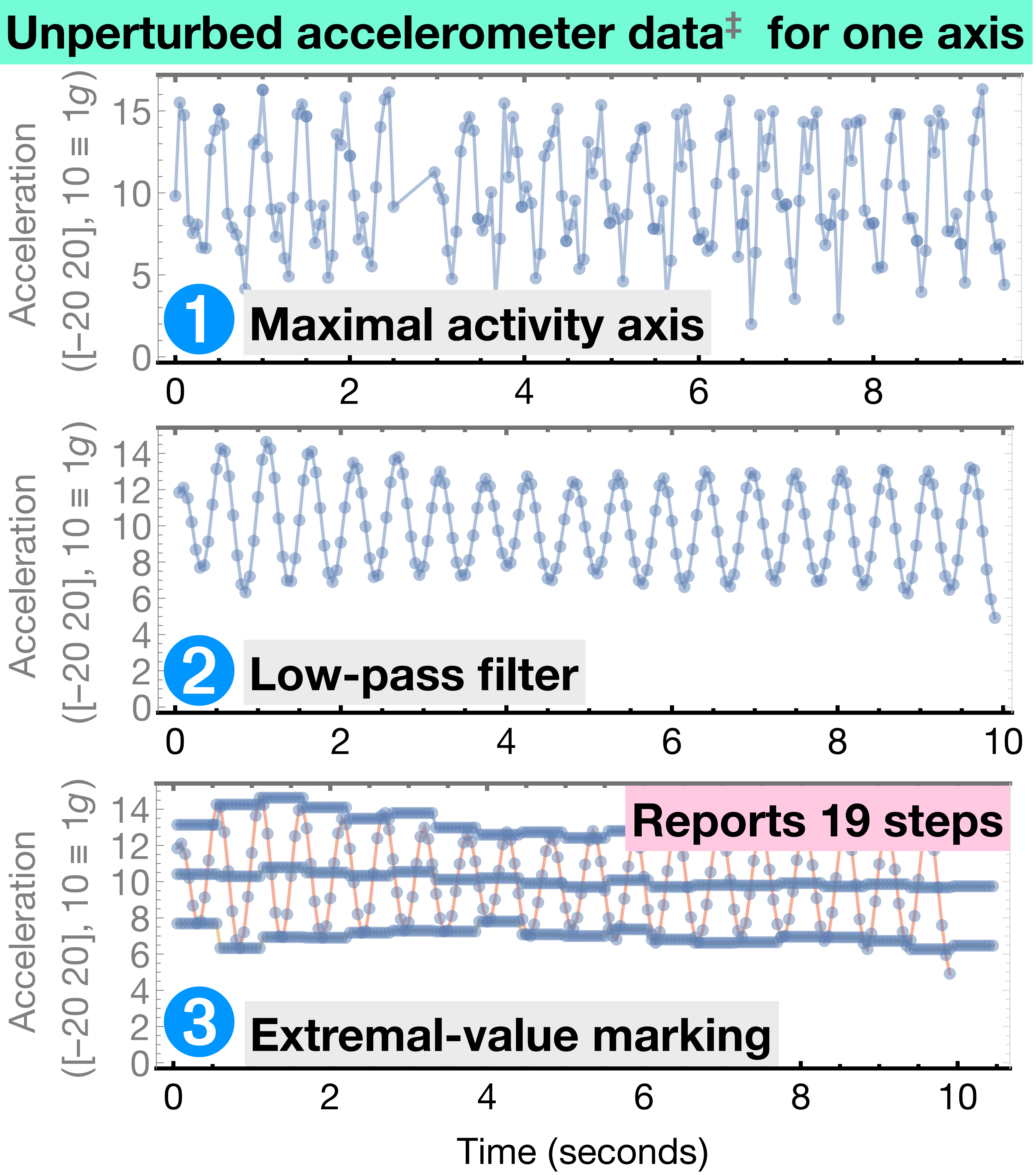
```
/*   Use Lax to achieve lowest power for required accuracy.  */
sampleC = lax_sensor_read(gyroSourceX, gyroTolerance);
```

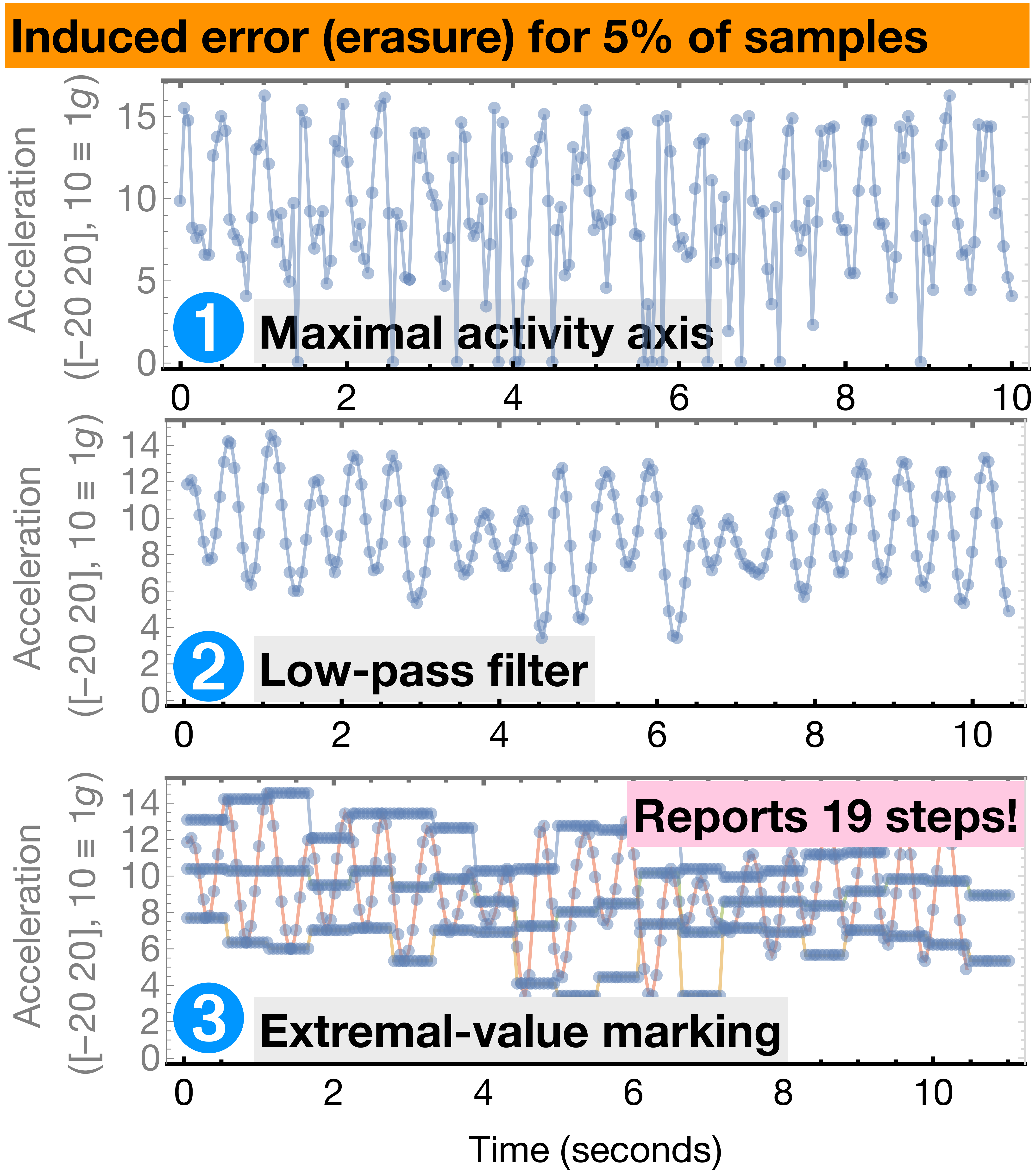
Great, But How Does it Affect Real End-to-End Applications?



‡ N. Zhao. Full-Featured Pedometer Design Realized with 3-Axis Digital Accelerometer. Analog Dialogue, 44(06), June 2010.

Great, But How Does it Affect Real End-to-End Applications?





‡ 3-axis accelerometer data from WISDM activity recognition dataset: J. R. Kwapisz, G. M. Weiss, and S. A. Moore. "Activity recognition using cell phone accelerometers". *SIGKDD Explor. Newsl.*, 12(2):74– 82, Mar. 2011.

Where We Are Today with **Lax**

We are pursuing both the language and hardware facets of Lax

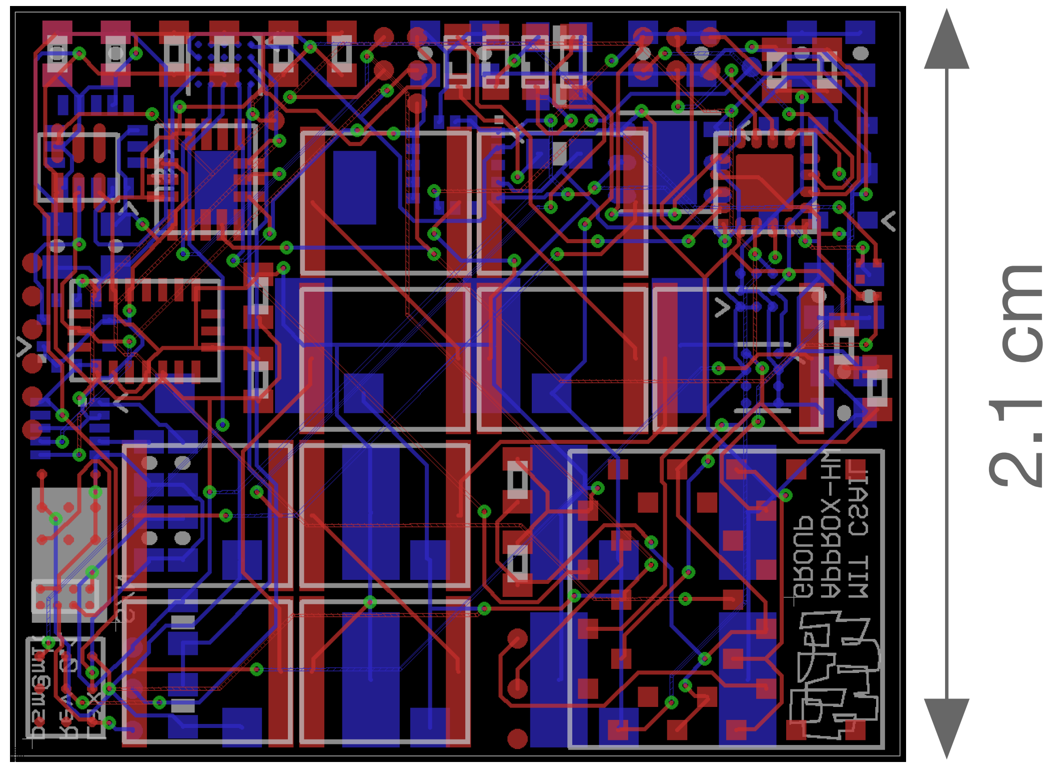
Evolving language design (e.g., temporal distributions for tolerable error)

Statically determining appropriate runtime settings from Lax descriptions

Other hardware techniques to expose new tradeoffs to Lax

Custom HW to explore using Lax in a “real” system

Why: usage experience, multiple sensors for ground truth, etc.



2x Pressure sensors
3x Accelerometers
2x Gyroscopes
2x Magnetometers
2x Infrared/light sensors
2x Humidity sensors
1x Bluetooth LE radio

Hardware support for sensor voltage control

+ More…

The Big Picture: What **Lax** Enables

- **A practical way to reduce power dissipation in sensor-driven systems**.

  Provides a **new dimension for tradeoffs**: error (accuracy / precision / reliability) in addition to traditional metrics of power and performance

The Big Picture: What **Lax** Enables

A practical way to reduce power dissipation in sensor-driven systems.

Provides a **new dimension for tradeoffs**: error (accuracy / precision / reliability) in addition to traditional metrics of power and performance

Opportunities for new research directions:

For language / compiler research

❶ Types for error properties (there is already some prior work)?

❷ Determine required accuracy/reliability from program source?

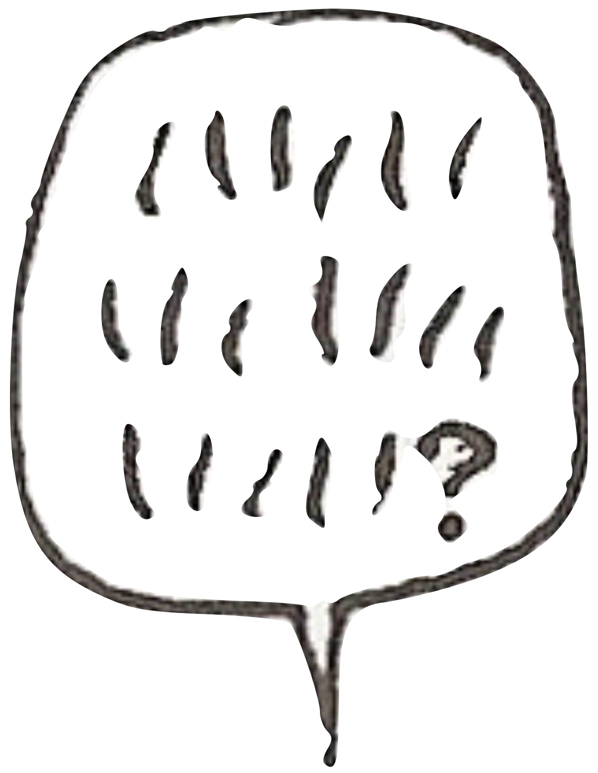
For systems research

❶ OS as an error-varying programmable VM?

❷ Scheduling to align **error-alike accesses**?

❸ Should OS manage error (like power, time)?

Q&A



Opportunities for new research directions:

For language / compiler research

❶ Types for error properties (perhaps, build on linear dependent types)?

❷ Estimate required accuracy/reliability from program source?

For systems research

❶ OS as an error-varying programmable VM?

❷ Scheduling to align **error-alike accesses**?

❸ Should OS manage error (like power, time)?