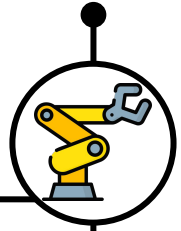
Smart Systems for a Sustainable Future: The Anatomy of Intelligent Design

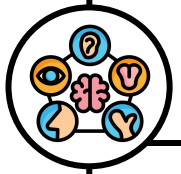
Applying Sensing, Communication, and Computation to Solve Real-World Challenges

Prof. Jamie Paik
Doc. Assist. Theodoros Papafotiou

Reconfigurable Robotics Laboratory
EPFL, Switzerland
02/10/25

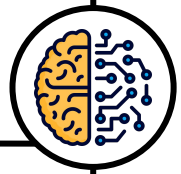
Sensor Fusion
Combining Information

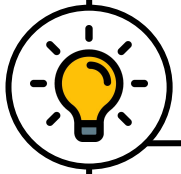




Communication
Connecting the pieces

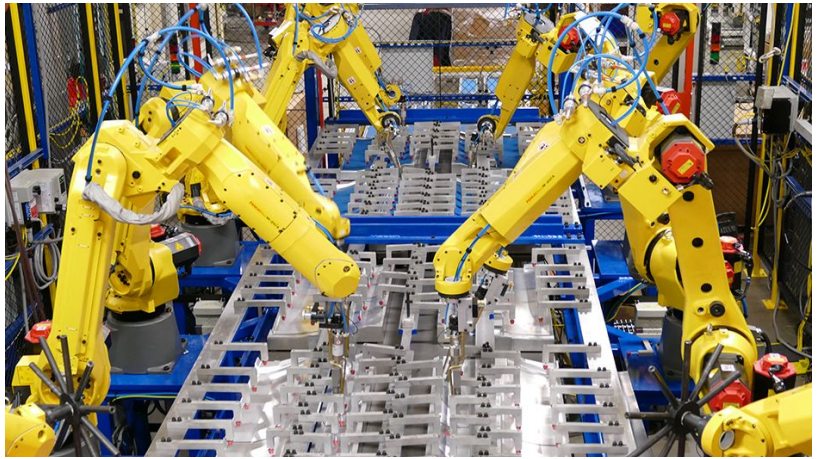
Computation
Thinking and Deciding





Online Demo
Give it a try!

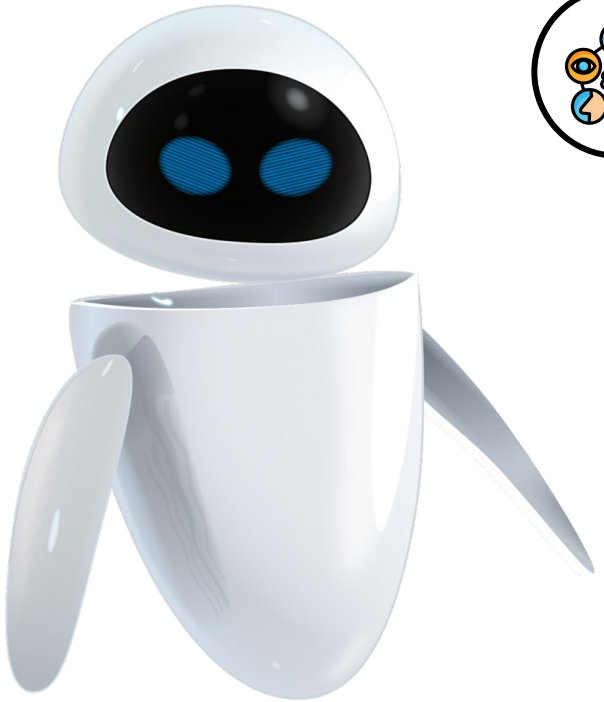
The Shift in Automation's Purpose

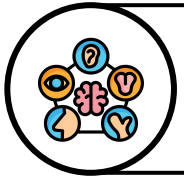


Automation = **Replacing** human labor, increasing speed, ensuring **repeatability**.

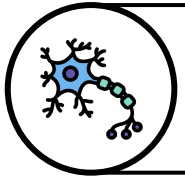
Automation = **Augmenting** systems for **optimal** performance. It's about making systems *aware*, *responsive*, and *efficient*.

The System as a Living Organism

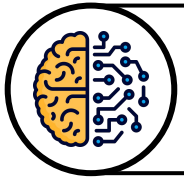




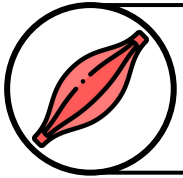
Sensing & Fusion



Communicating

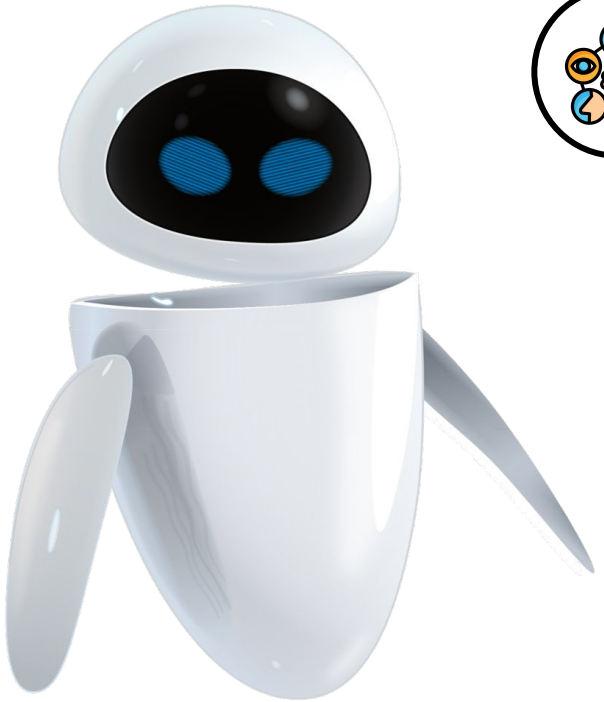


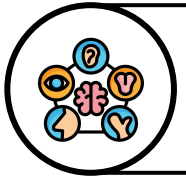
Computing



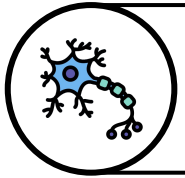
Actuating

The System as a Living Organism

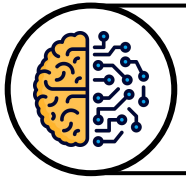




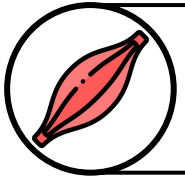
What is the world's state?



How does information travel?



What should I do?



Actuating

What Do We Need to Sense?

Proprioception
(Self-awareness)

Is the hydraulic press at the right pressure?

Is the motor overheating?

Exteroception
(External Awareness)

Is there a person nearby?

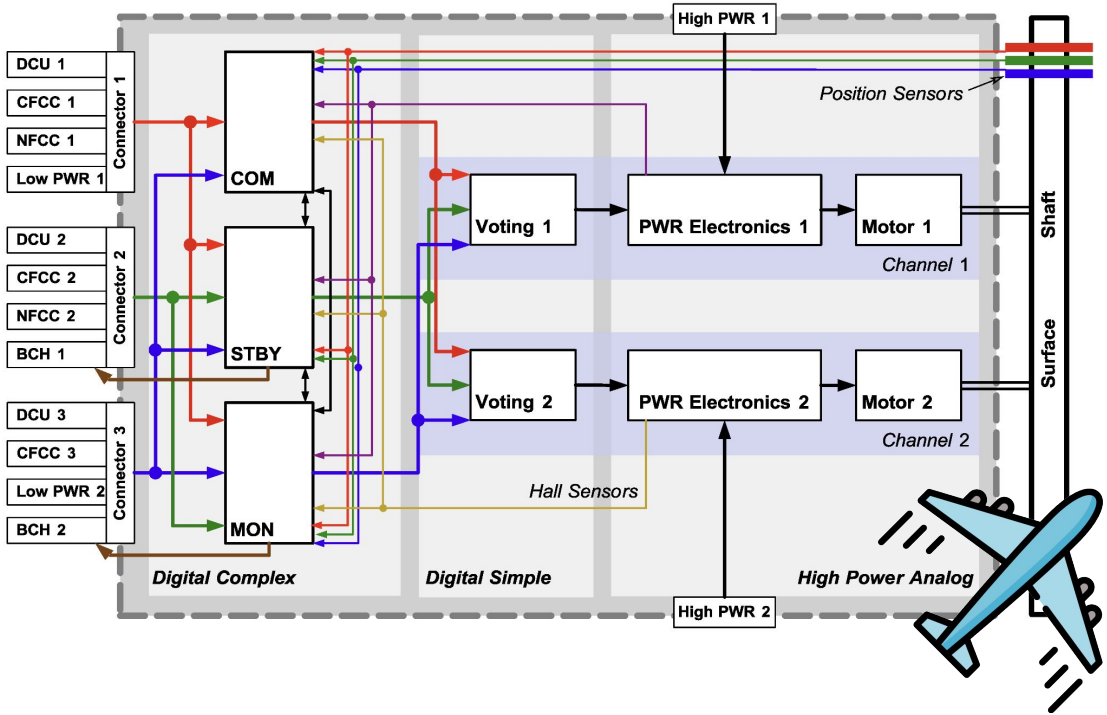
Is the raw material correctly aligned?

"What info does the robot **absolutely** need to know?"

"Sensing enables you to **Manage & Optimize**"

Sensor Fusion

"Critical systems **never** trust a **single source** of information."



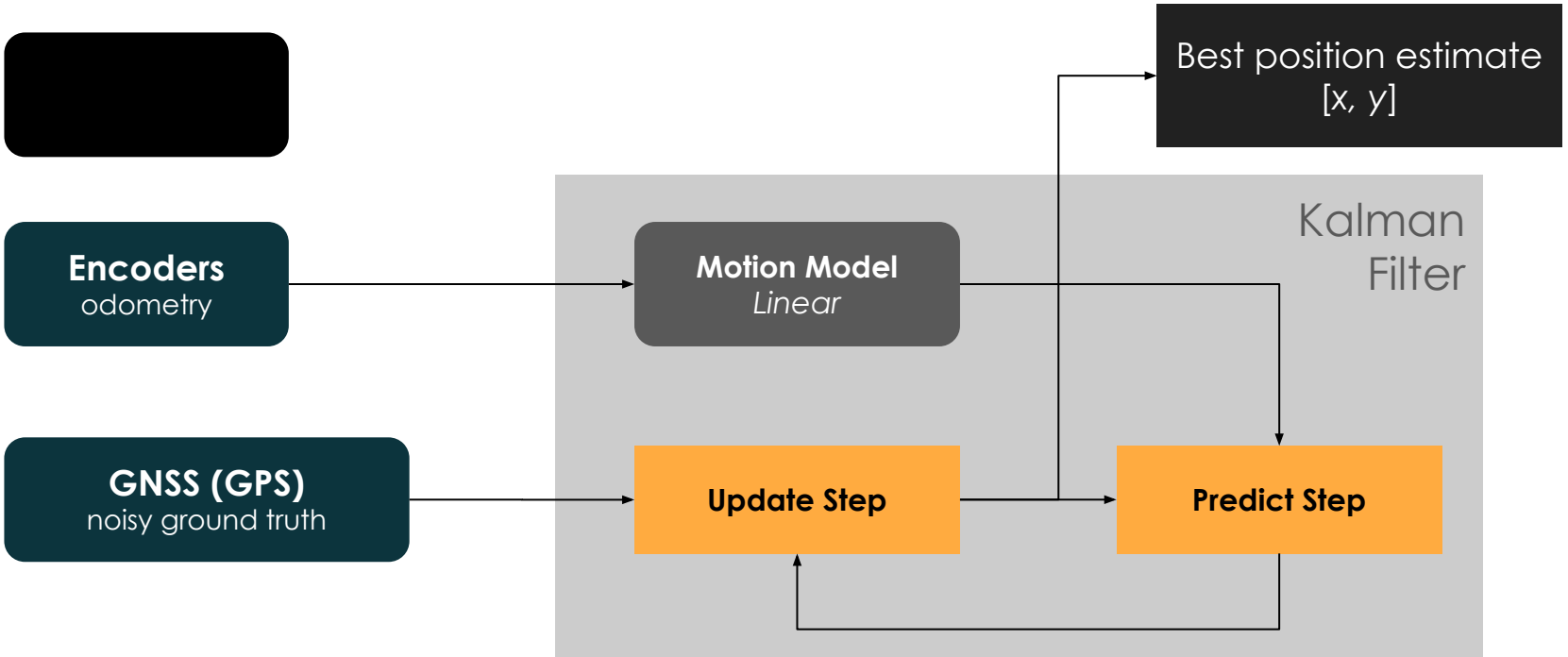
Design rules for trustworthy fusion

- **Diverse physics > duplicate parts**
  e.g., person detection from *vision + thermal + ultrasonic cameras*
- **Time sync is king (!)**
  consistent timestamps beat any algorithm
- **Know your error budget**
  bias vs noise vs drift; track units and covariances
- **Calibrate & align**
  intrinsic calibration per sensor and *extrinsics* between them (e.g., camera ↔ LiDAR).
- **Guard against correlated failures**
  place sensors independently; consider *2oo3* (two-out-of-three) voting for safety-critical channels.
- **Graceful degradation**
  define fallback modes when a sensor goes stale or saturated

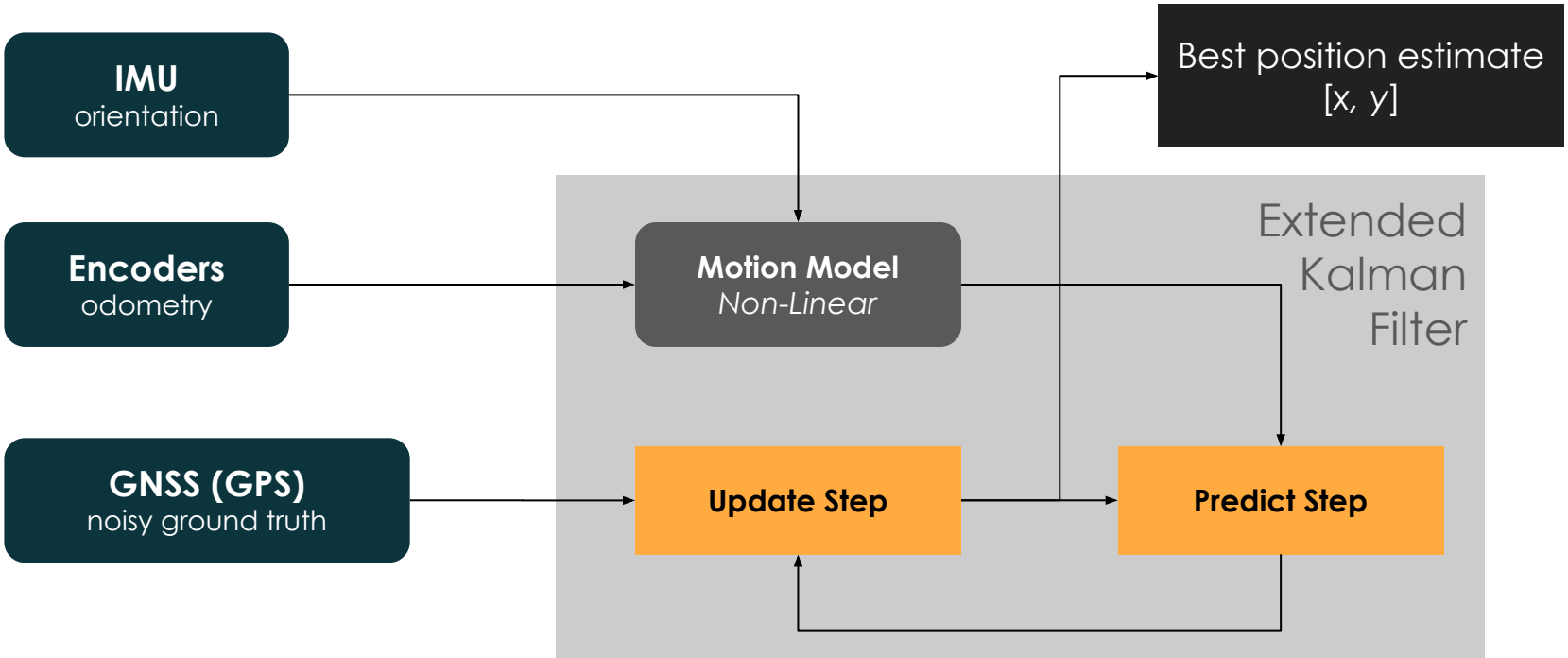
Fusion Techniques (among others…)

Technique	Core Idea *The Analogy*	Best For *The Use Case*
Kalman Filter	Predict & Correct	**Linear Systems:** When combining noisy measurements to track a single, smoothly changing state.
Extended KF / Unscented KF	Handling Curves	**Moderately Non-Linear Systems:** When estimating states in systems with curved or nonlinear dynamics.
Particle Filter	The Search Party	**Highly non-linear or multi-modal systems:** When there may be multiple plausible states.
Complementary Filter	The Fast Talker & the Slow Thinker	**Resource-limited real-time fusion:** Combining fast-drifting and slow-stable sensors.
Neural Networks	The Black Box Learner	**Extremely Complex Systems with lots of data:** When the physics is too hard or impossible to model.

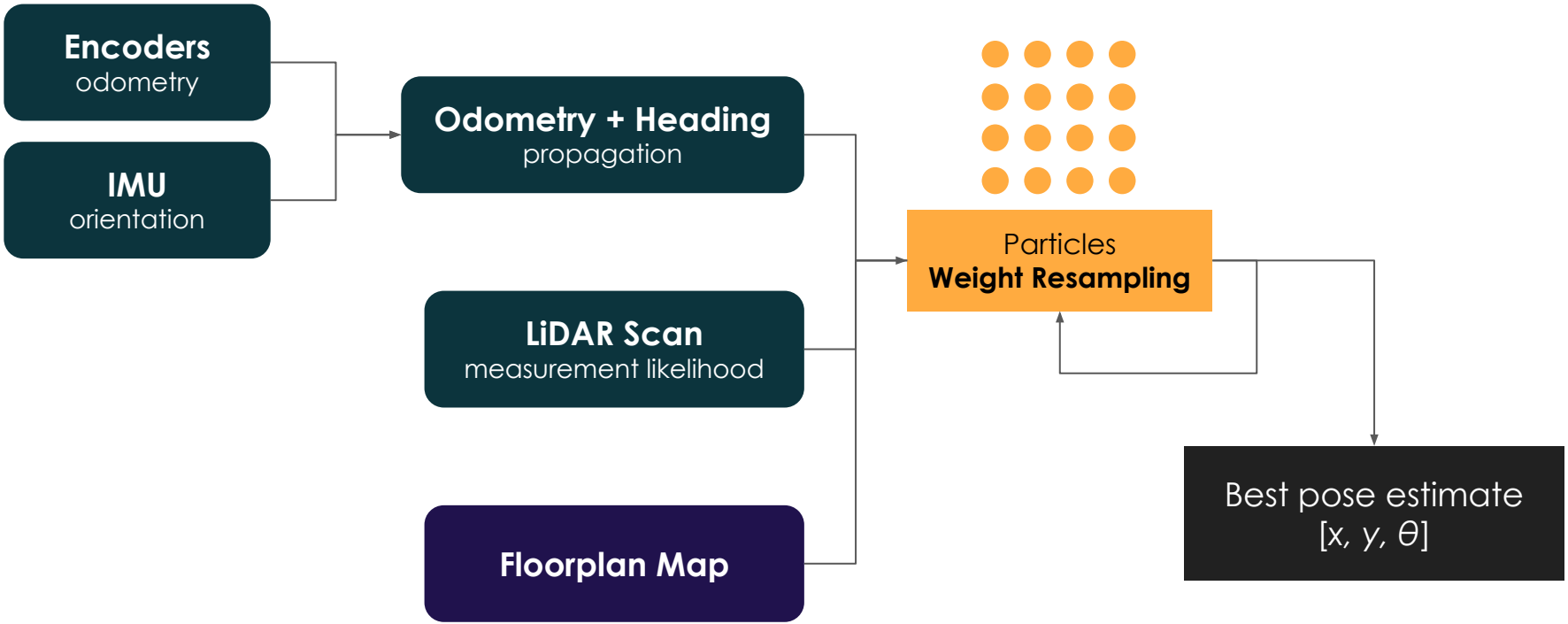
Robot Localization in open-field (straight)



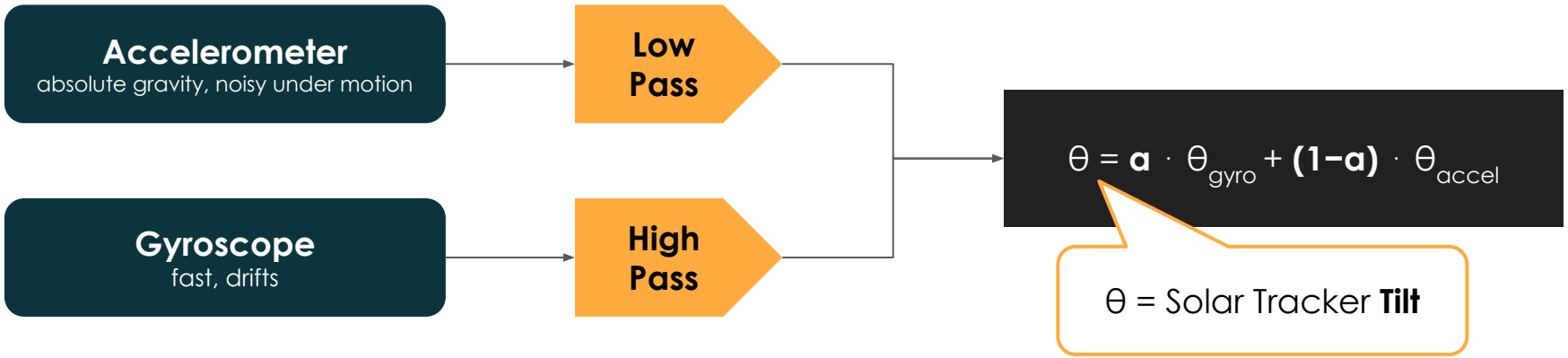
Robot Localization in open-field (curved)



Robot Localization in a building

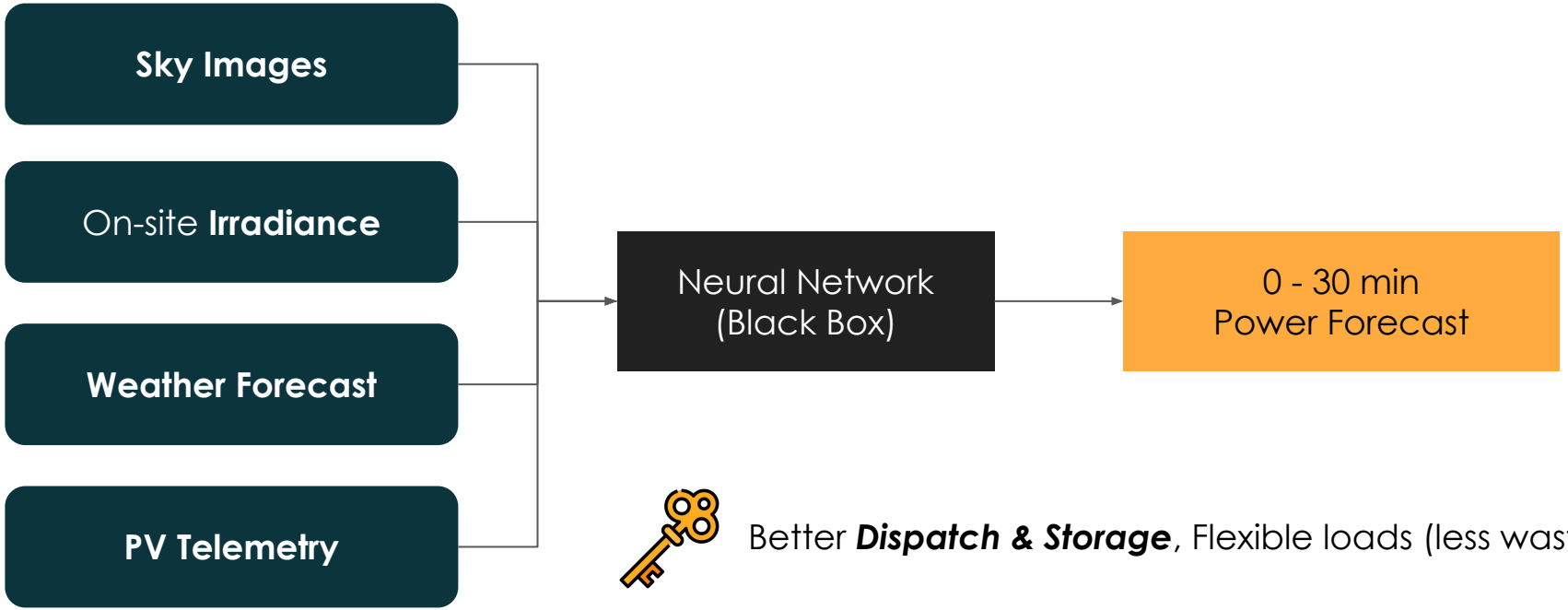


Tilt Estimation



Better ***pointing*** → More kWh/day without expensive sensors.

Photovoltaic (PV) Power Nowcasting



Better ***Dispatch & Storage***, Flexible loads (less waste)

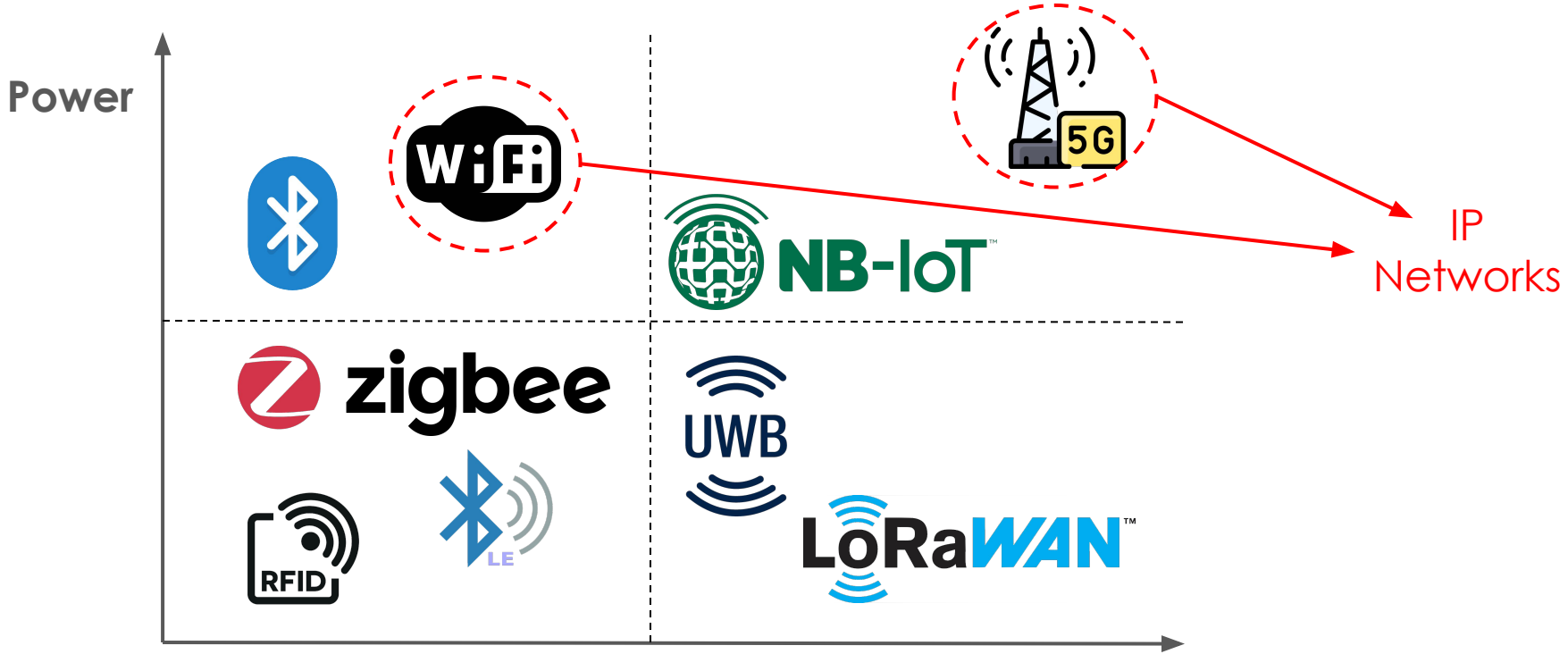
The Nervous System

Key Considerations:

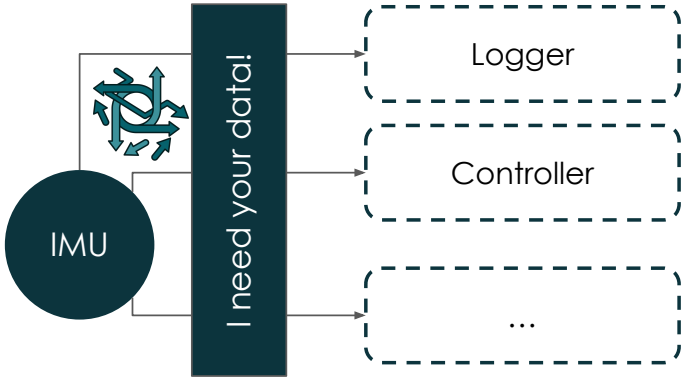
Power Consumption

Range & Bandwidth

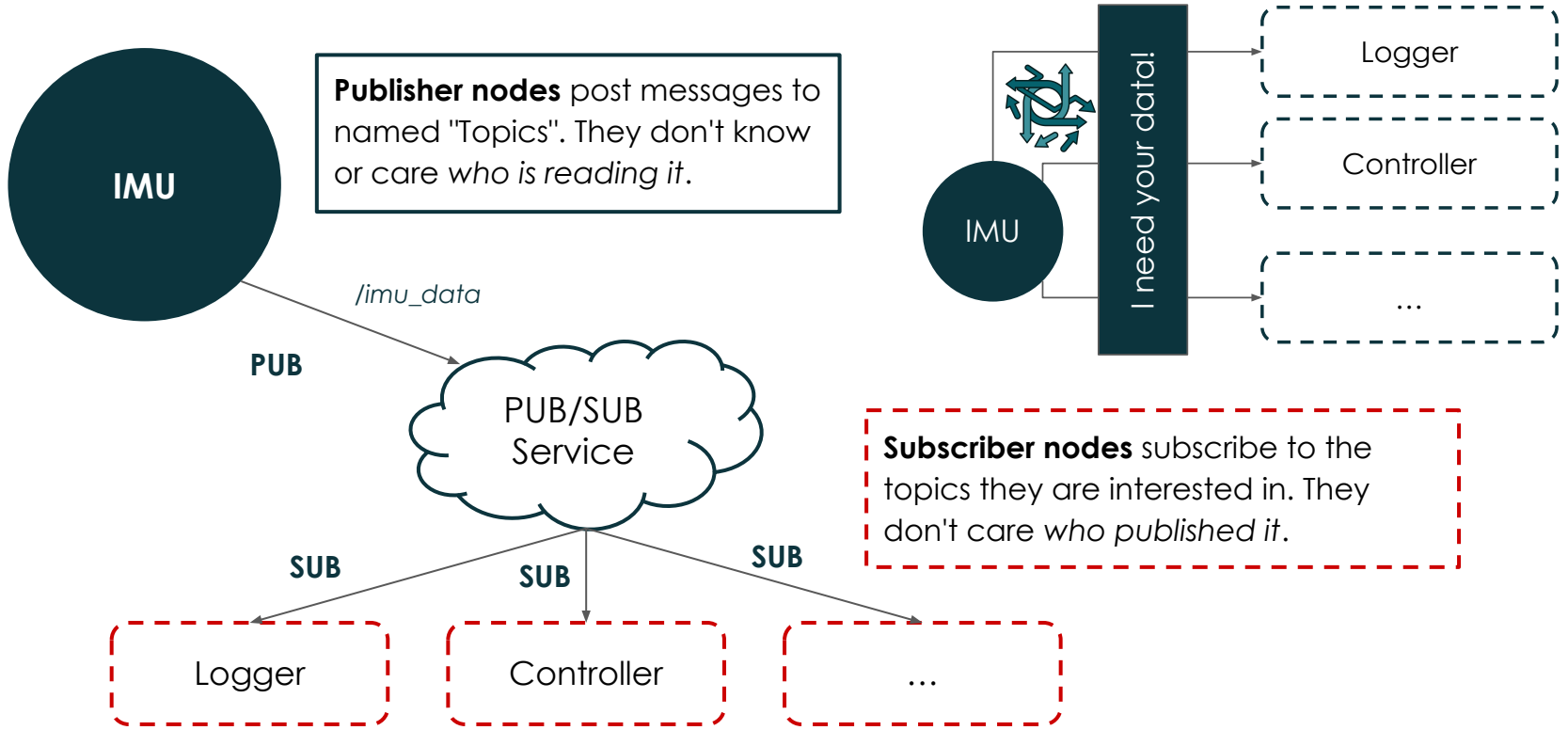
A Spectrum of Choices



IP Networks - The Pub/Sub Pattern



IP Networks - The Pub/Sub Pattern



IP Networks - Middleware & Brokers

ROS (Robot Operating System): Widely used in academic robotics. A popular middleware implementation of the Pub/Sub pattern, with lots of tools for robotics.

MQTT: Lightweight, publish–subscribe network protocol for message queuing service. It is designed for connections with devices with resource constraints or **limited network bandwidth**, such as in the Internet of things (IoT).

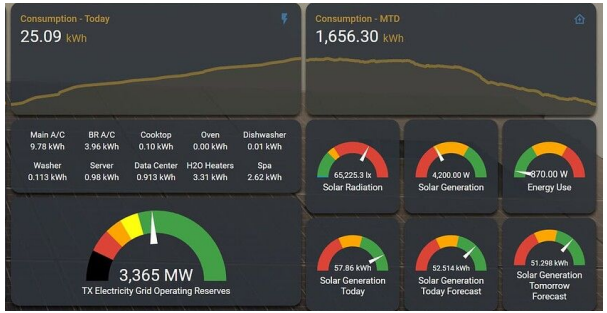
Among Others:

The Spectrum of "Brains"

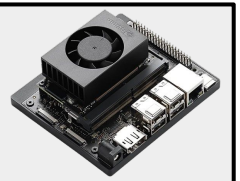
Cloud Services
Extending the capabilities

Google Cloud aws Red Hat IBM Cloud

- Massive computational power.
- Great for: analyzing historical data, training AI models, and managing a fleet of devices.

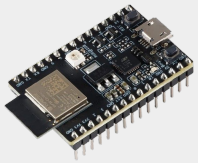


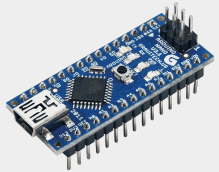
Single-Board Computers (SBCs)
Handling the thinking



- High-level logic
- Runs a full OS (Linux).
- Great for: sensor fusion, running AI models, logging data, providing a user interface, and complex decision-making.

Microcontrollers (MCUs)
Handling the reflexes





- Hard real-time tasks
- Deterministic and Reliable
- Great for: Reading sensors at a fixed high rate, running fast control loops (e.g., a PID loop for a motor), generating PWM signals

https://www.etsy.com/listing/1394621847/esp32-diy-4-heads-peristaltic-dosing

Robustness in Software

A system must do many things at once. How you structure this is a key design choice for reliability.

Multithreading

"They can work fast but might bump into each other or grab the same knife"

Multiprocessing

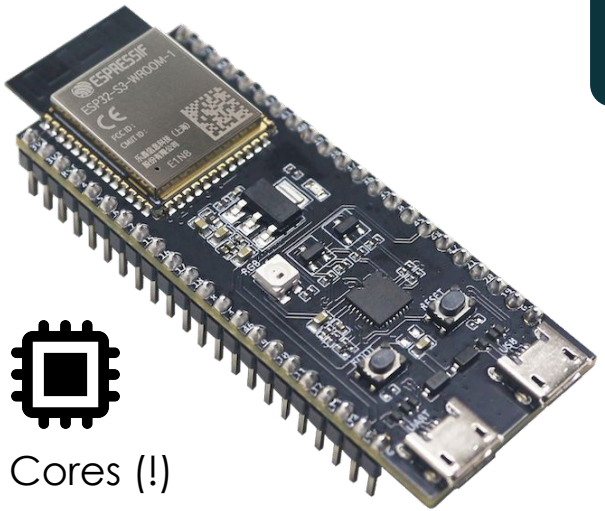
"Safer and more organized, but they need to use a service window to pass ingredients"

Tools = Memory

Robustness in Software

	Pros	Cons
Multithreading	Very **fast communication** between threads.	Dangerous due to **Race Conditions** that can lead to data corruption. **Mutexes** (locks) are a sufficient solution.
Multiprocessing	Safer and more robust, as **memory is isolated**. If one process crashes, it doesn't bring down the whole system.	**Communication** between processes is **slower** (Inter-Process Communication or IPC)

Demo (online): Try these concepts!



2x Cores (!)

Kinda Multithreading
(FreeRTOS tasks pinned in **same** core)

Kinda Multiprocessing
(FreeRTOS tasks pinned in **both** cores)

MQTT Broker connection and Pub/Sub

WiFi Connected

Thank you!

Questions?