The Robotic Nervous System: Sensing, Communication, and Computation

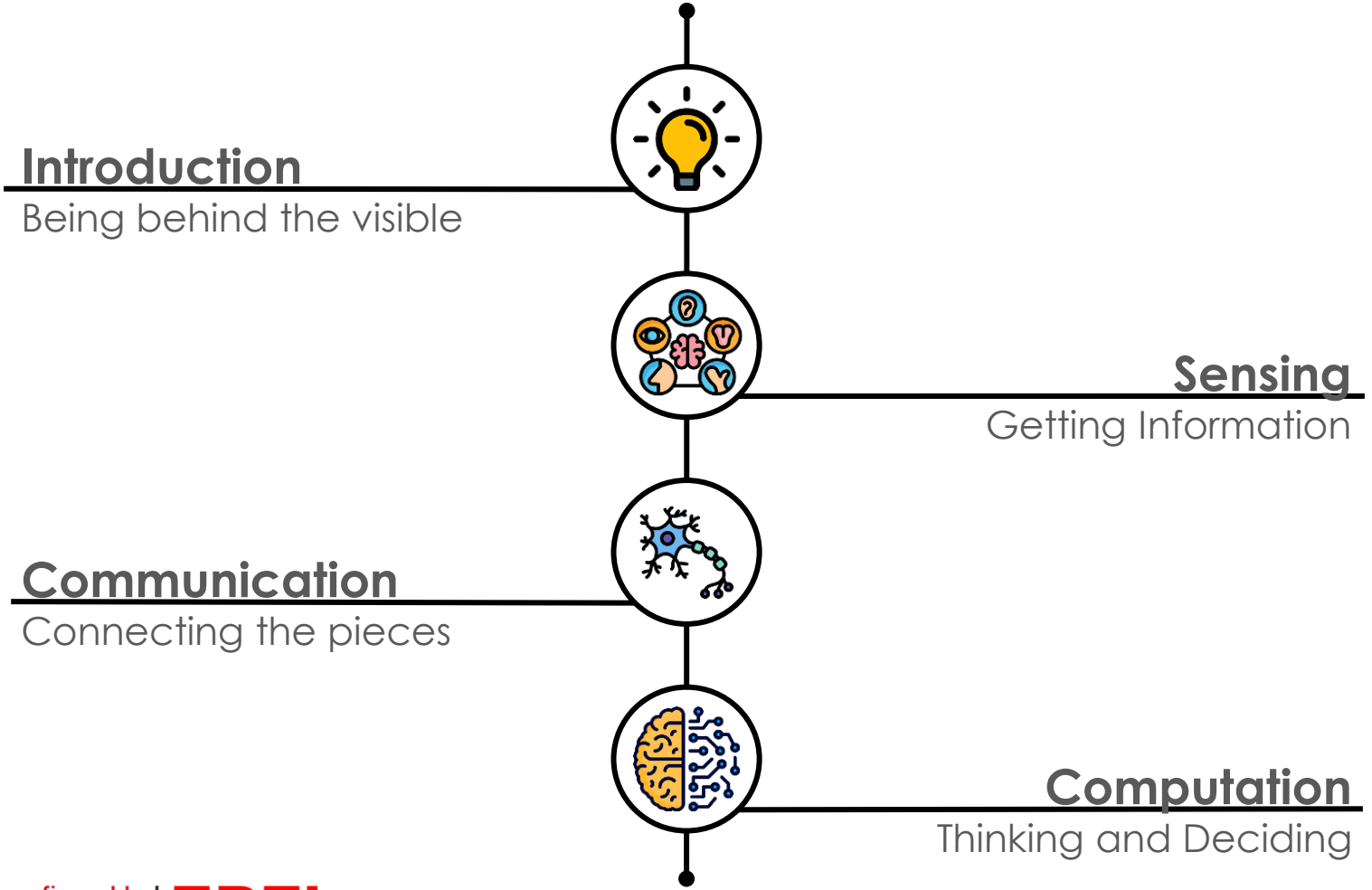
Thinking like a systems engineer to build robust, real-time robotic systems

Prof. Jamie Paik
Doc. Assist. Theodoros Papafotiou

Reconfigurable Robotics Laboratory
EPFL, Switzerland
03/10/25

RRL Reconfigurable Robotics Lab | EPFL

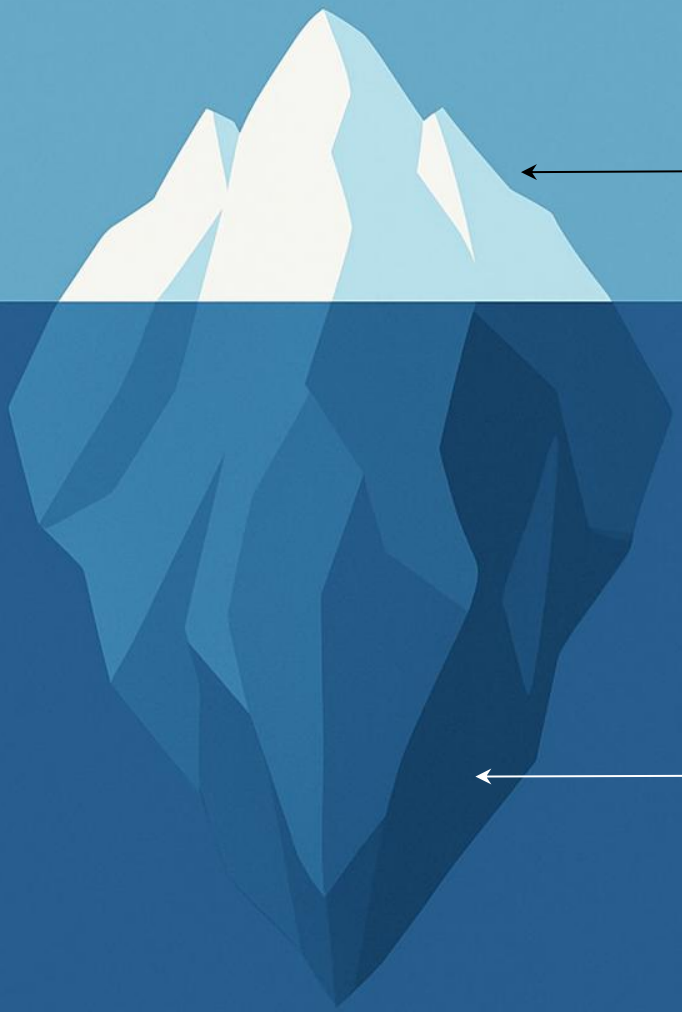
ME410 - Mechanical product design & development


Introduction
Being behind the visible
Sensing
Getting Information
Communication
Connecting the pieces
Computation
Thinking and Deciding

The "Magic" of a Wearable Robot

Nahoko Yamamura, Daisuke Uriu, Mitsuru Muramatsu, Yusuke Kamiyama, Zendai Kashino, Shin Sakamoto, Naoki Tanaka, Toma Tanigawa, Akiyoshi Onishi, Shigeo Yoshida, Shunji Yamanaka, and Masahiko Inami. 2023. Social Digital Cyborgs: The Collaborative Design Process of JIZAI ARMS. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23). Association for Computing Machinery, New York, NY, USA, Article 369, 1–19. https://doi.org/10.1145/3544548.3581169

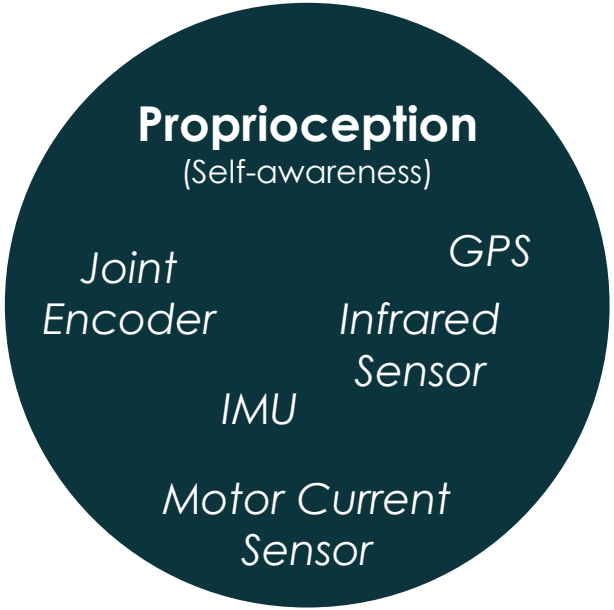
The Systems Engineering

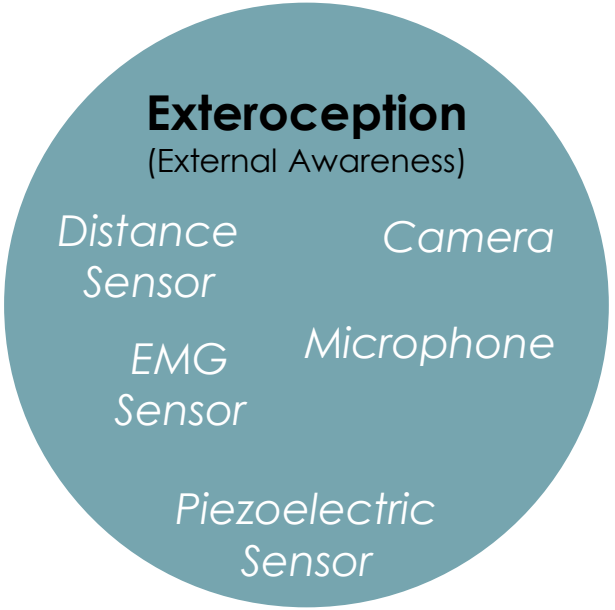


The physical hardware (links, motors, sensors)

Real-time data flow, software architecture, fault handling, communication busses.

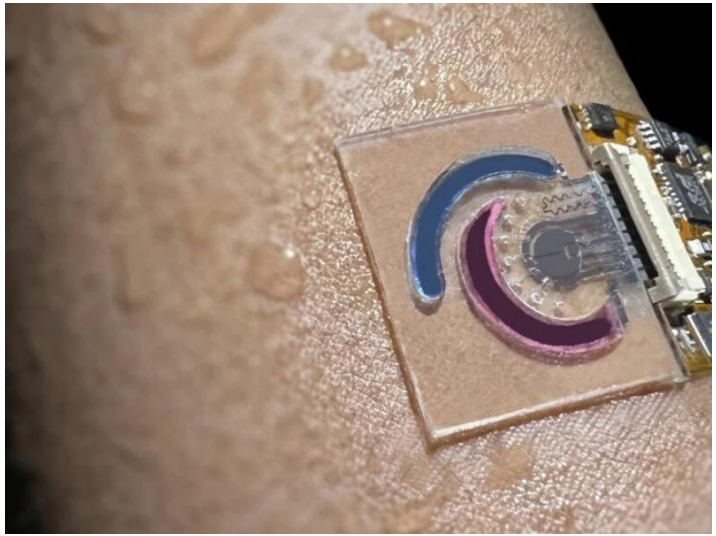
What Do We Need to Sense?



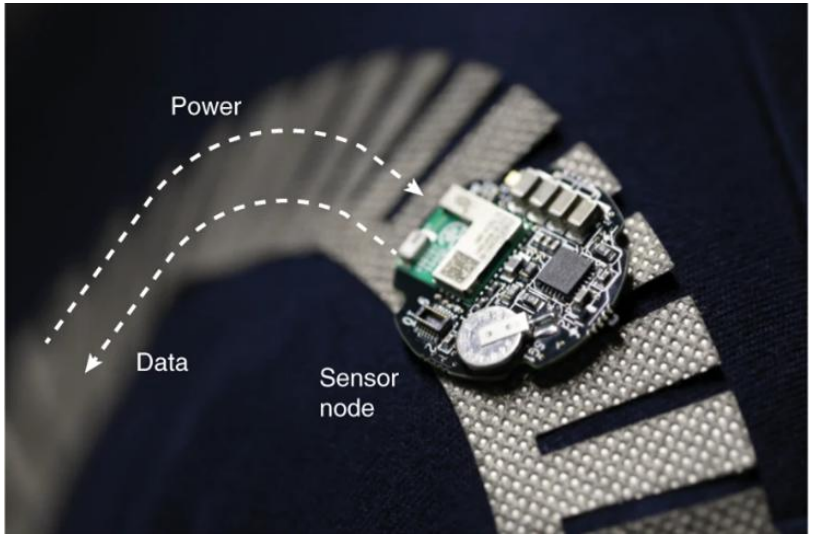


"What information does the robot **absolutely** need to know?"

What Do We Need to Sense?



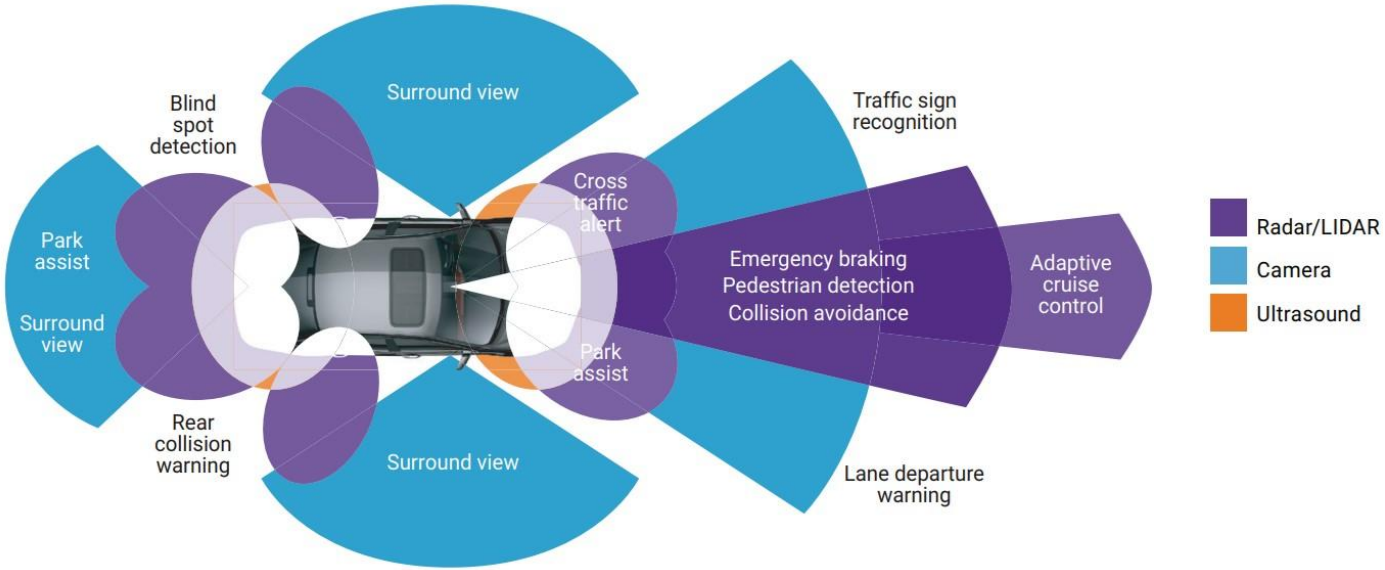
Wang, M., Yang, Y., Min, J. et al. A wearable electrochemical biosensor for the monitoring of metabolites and nutrients. Nat. Biomed. Eng 6, 1225–1235 (2022). https://doi.org/10.1038/s41551-022-00916-z



Tian, X., Lee, P.M., Tan, Y.J. et al. Wireless body sensor networks based on metamaterial textiles. Nat Electron 2, 243–251 (2019). https://doi.org/10.1038/s41928-019-0257-7

The Challenge: Sensor Fusion

"You rarely trust one sensor alone."



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).

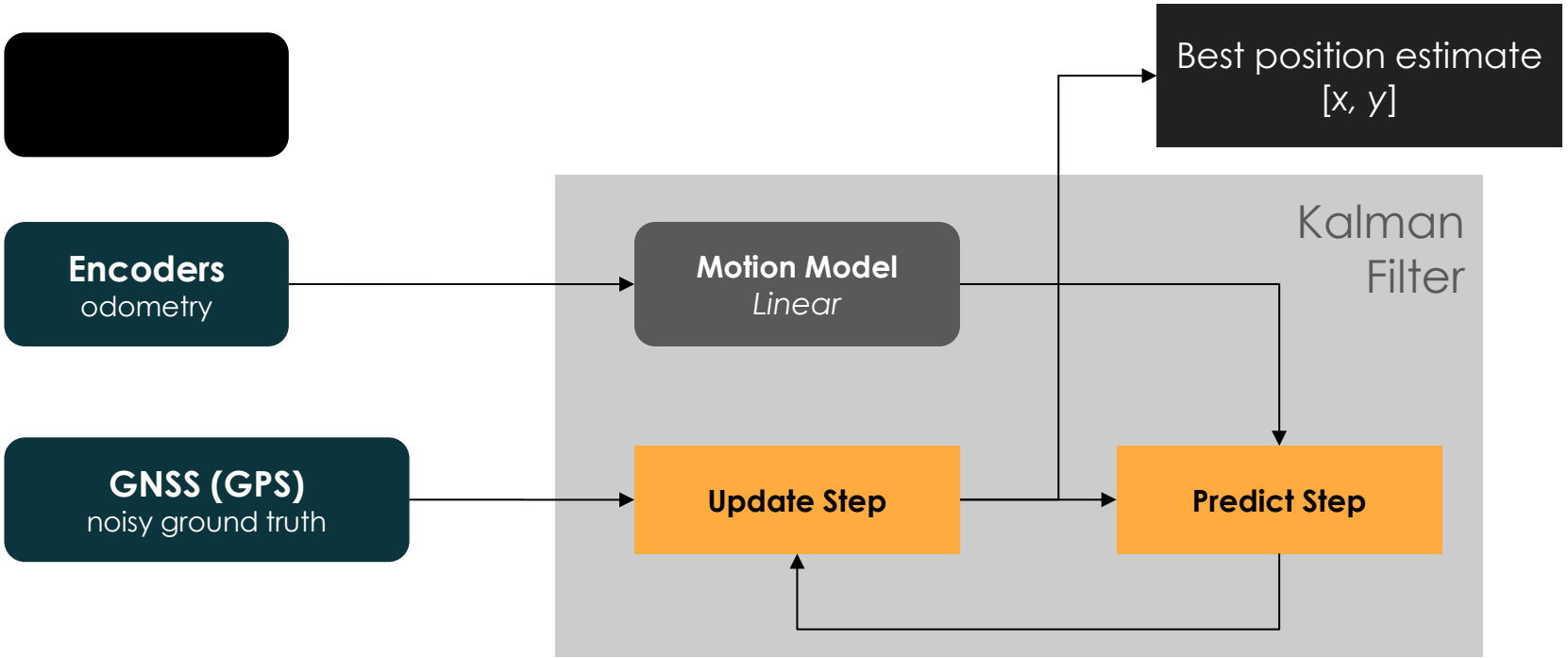
- **Guard against correlated failures**
  place sensors independently; consider *2oo3* (two-out-of-three) voting for safety-critical channels.

- **Graceful degradation**
  define fallback modes when a sensor goes stale or saturated

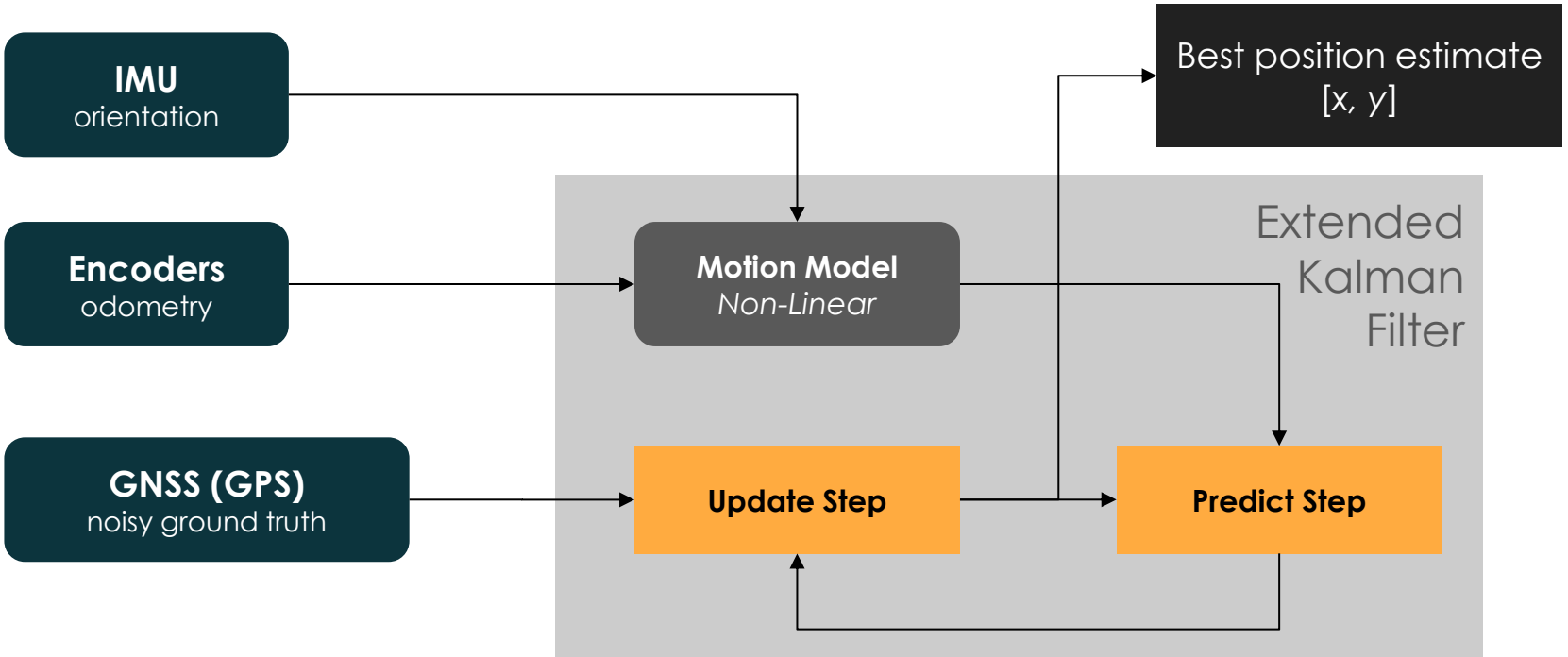
Fusion Techniques (among others…)

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.

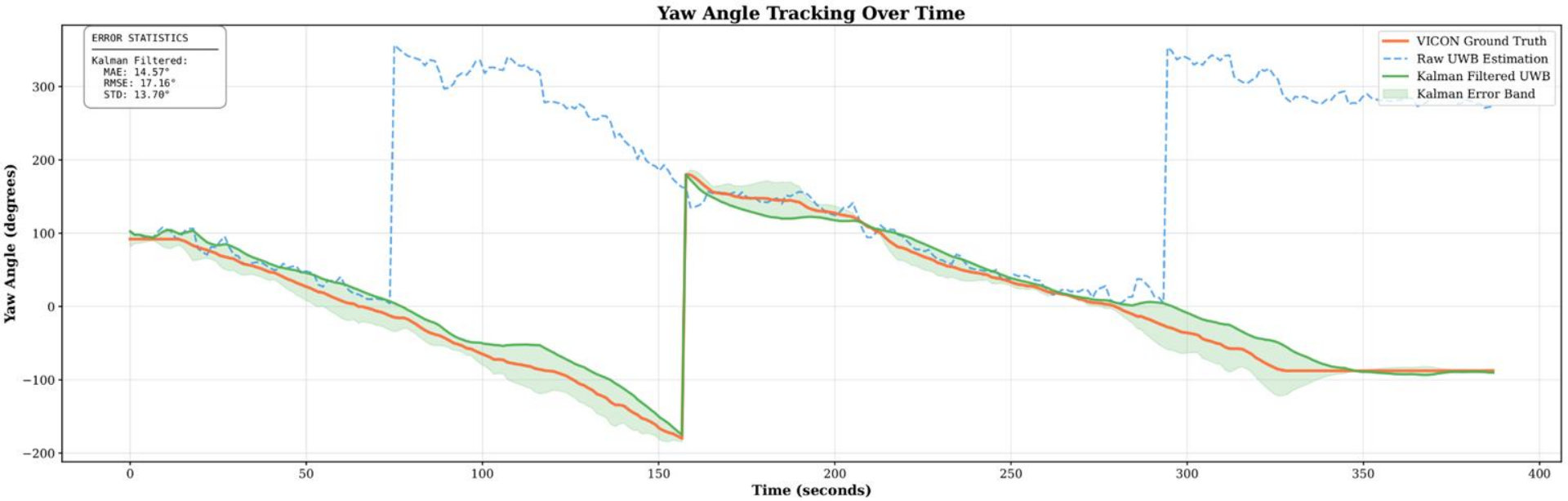
Robot Localization in open-field (straight)



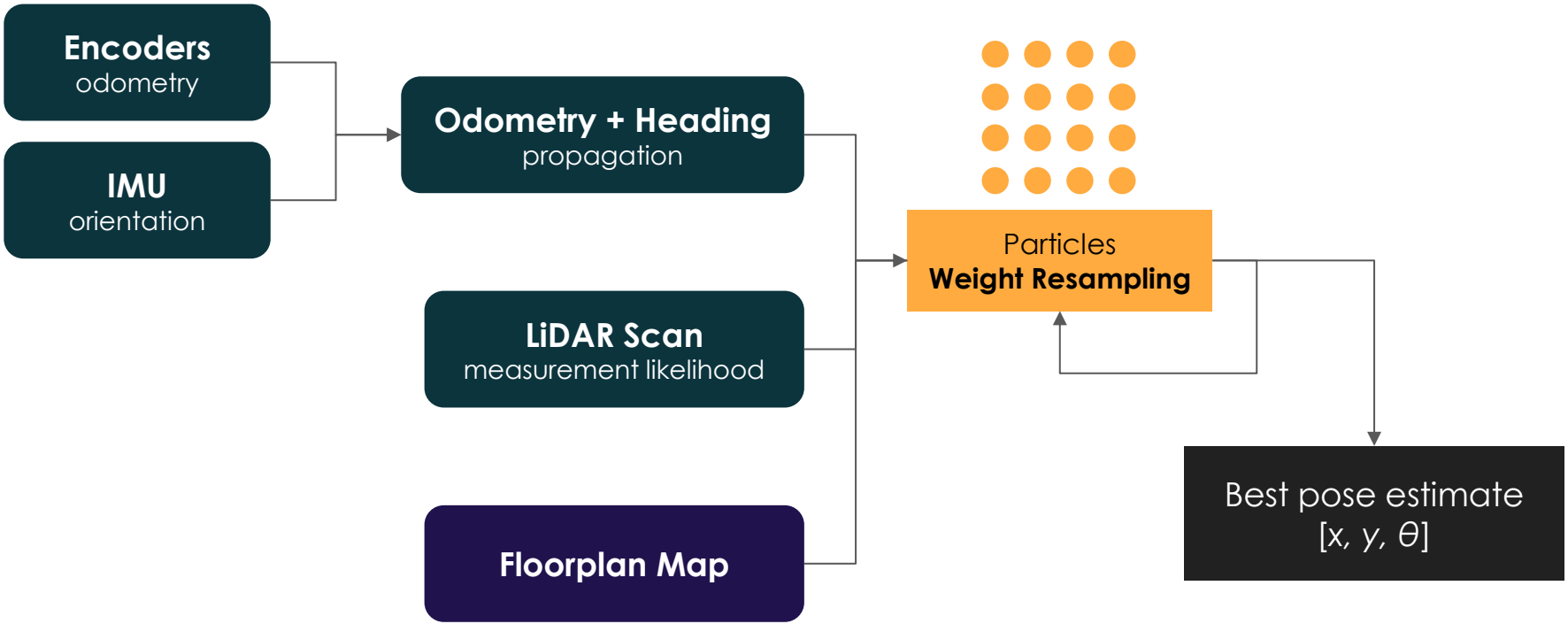
Robot Localization in open-field (curved)



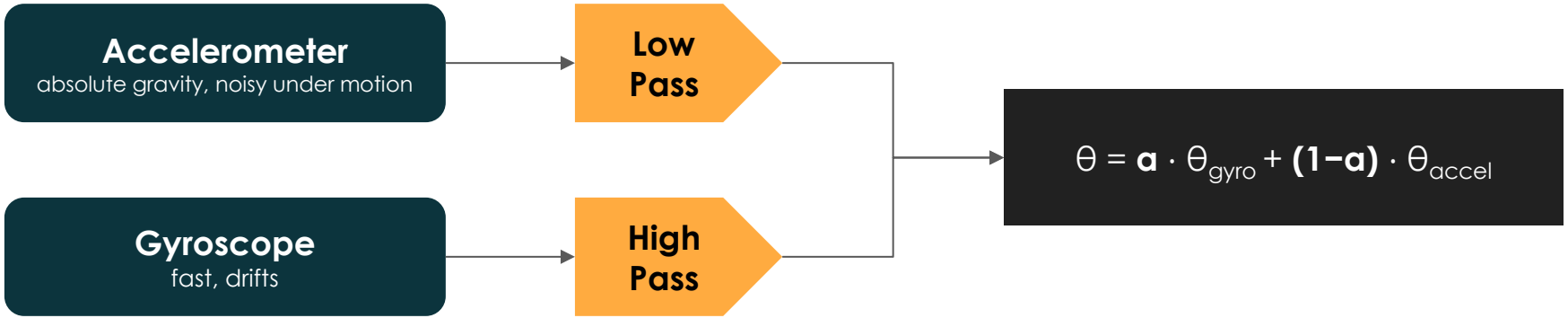
Last week's Kalman Filtering Application



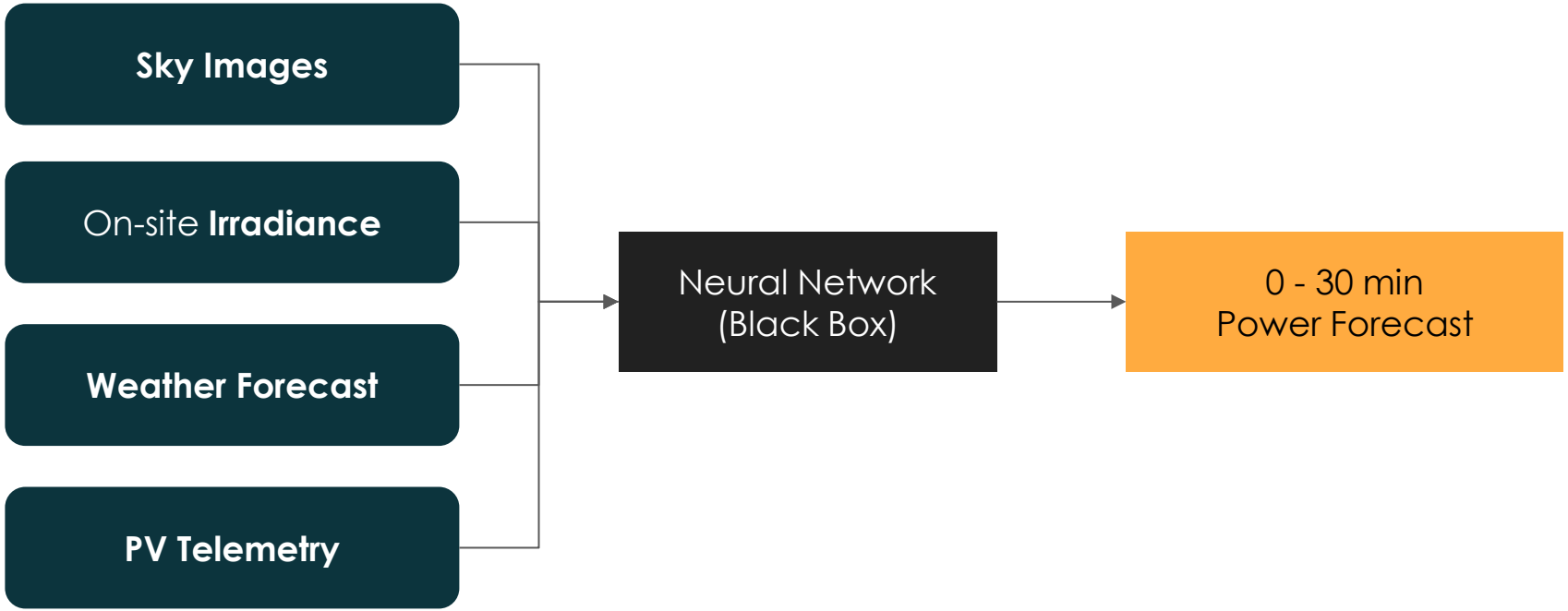
Robot Localization in a building



Tilt Estimation



Photovoltaic (PV) Power Nowcasting



Designing the Communication Pathways

"Where does the data need to go?"

System Level

?

"How do different software tasks share data?"

Board-to-Board

?

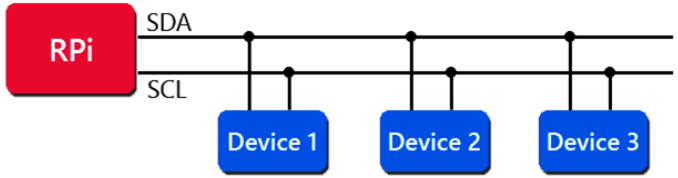
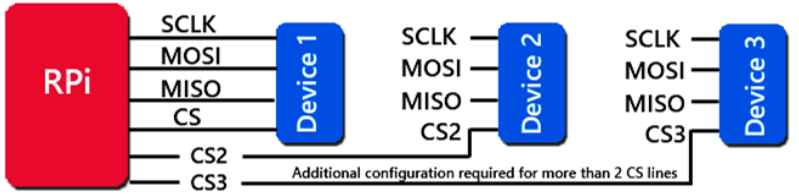
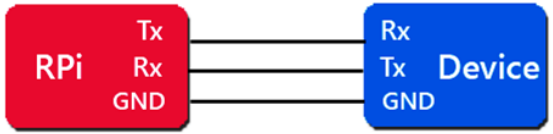
"How do my components coordinate?"

On-Board

?

"How do I read a sensor?"

On-Board Communication Protocols

Protocols	Characteristics	Connectivity
I2C	Slow, 2 wires, great for talking to many sensors on a single bus (e.g., multiple IMUs).	
SPI	Faster, 4 wires, point-to-point, good for high-throughput sensors like some ADCs.	
UART	Simple serial, point-to-point. The classic way for an SBC and MCU to exchange data and commands	

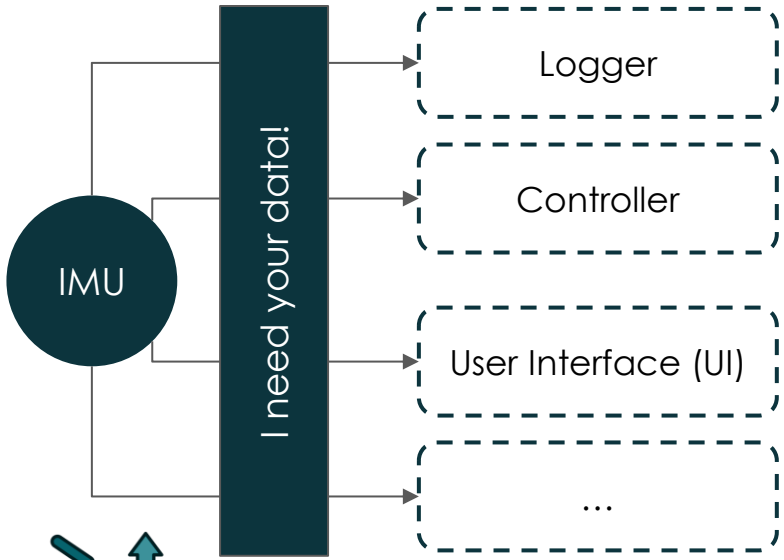
Board-Board Communication Protocols

Category	Data Rate	Examples
SR-HP	WiFi: 100-1000+ Mbps Bluetooth: 1–3 Mbps	
SR-LP	BLE: ~125 kbps – 2 Mbps RFID: ~100 bps – 640 kbps ZigBee: ~20–250 kbps	
LR-HP	5G: up to 10 Gbps NB-IoT: ~20–250 kbps	
LR-LP	LoRa: 0.3–50 kbps UWB: ~110 kbps – 27 Mbps	

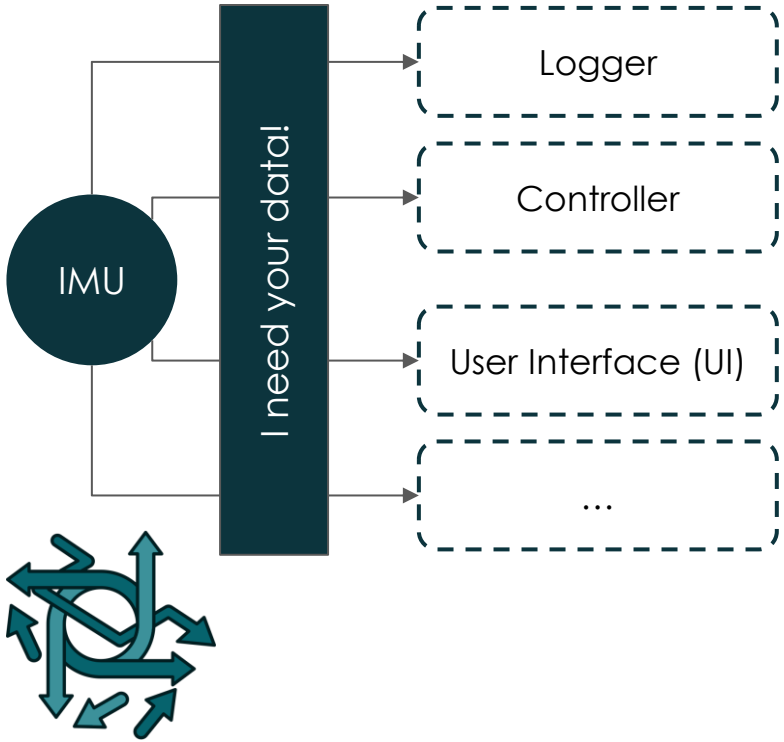
IP Networks

Short Range = SR, Long Range = LR, High Power = HP, Low Power = LP

The Publish-Subscribe Pattern

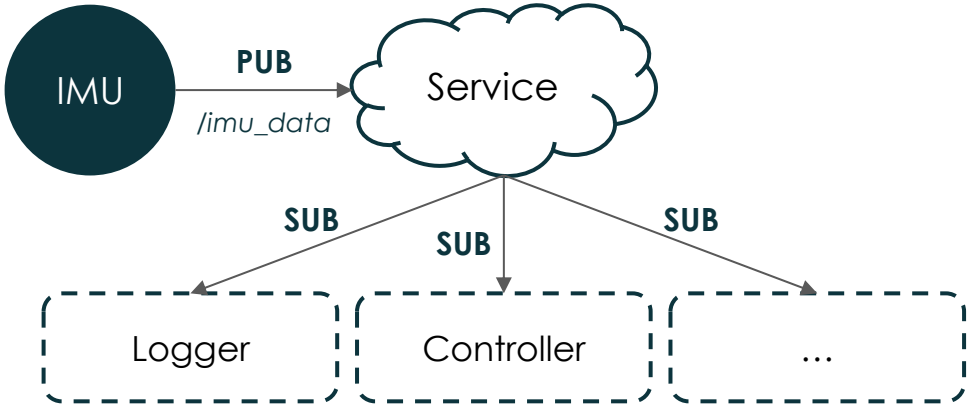


The Publish-Subscribe Pattern



Solution: Pub/Sub

- **Publisher nodes** post messages to named "Topics". They don't know or care *who is reading it*.
- **Subscriber nodes** subscribe to the topics they are interested in. They don't care *who published it*.



Decouples your system. You can add/remove components without breaking everything.

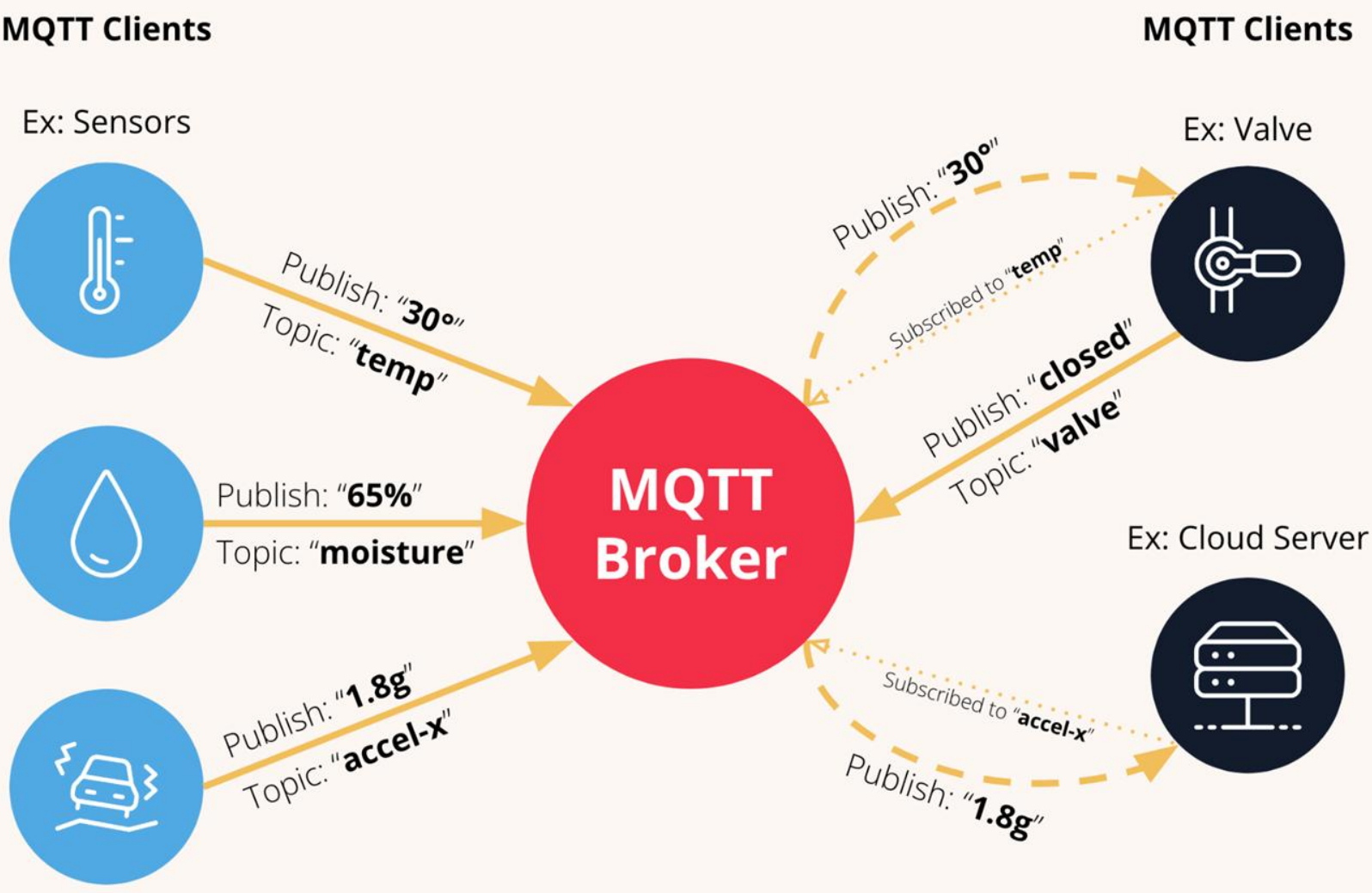
Pub/Sub in Practice: Middleware & Brokers

ROS (Robot Operating System): The de-facto standard 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).

Redis: In-memory **key–value database**, used as a distributed cache and **message broker**. Because it holds all data in memory and because of its design, Redis offers low-latency reads and writes.

Among Others:


MQTT Clients
MQTT Clients
Ex: Sensors
Ex: Valve
Ex: Cloud Server
Publish: "30°"
Topic: "temp"
Publish: "65%"
Topic: "moisture"
Publish: "1.8g"
Topic: "accel-x"
MQTT Broker
Publish: "30°"
Subscribed to "temp"
Publish: "closed"
Topic: "valve"
Subscribed to "accel-x"
Publish: "1.8g"

The Computational Hierarchy

Cloud Services
Extending the capabilities

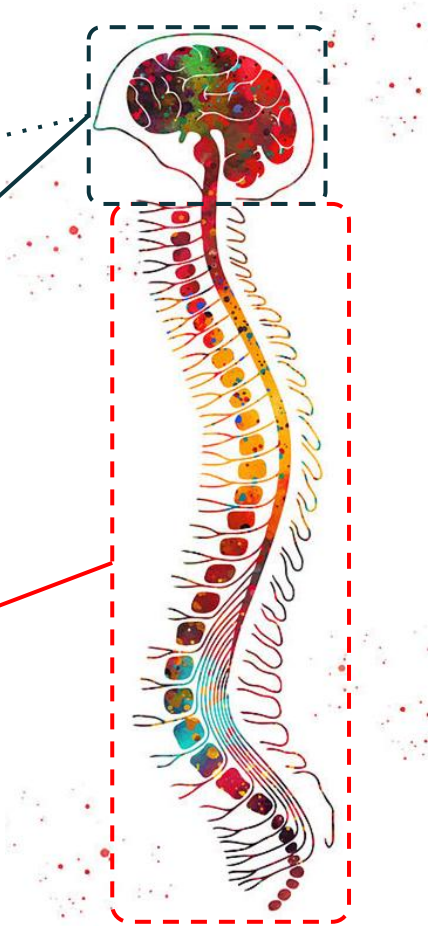
- Massive computational power.
- Great for: analyzing historical data, training AI models, and managing a fleet of devices.

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



The Software Challenge: Concurrency

	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)

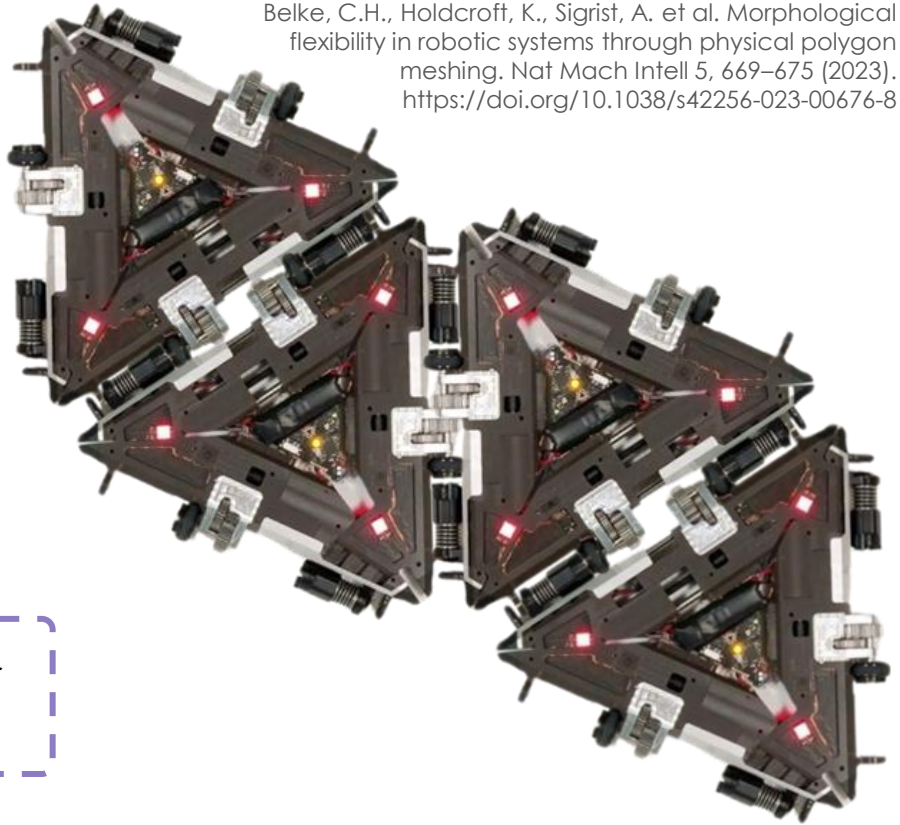
Tools = Memory

The MORI architecture

Accelerometer & Encoders

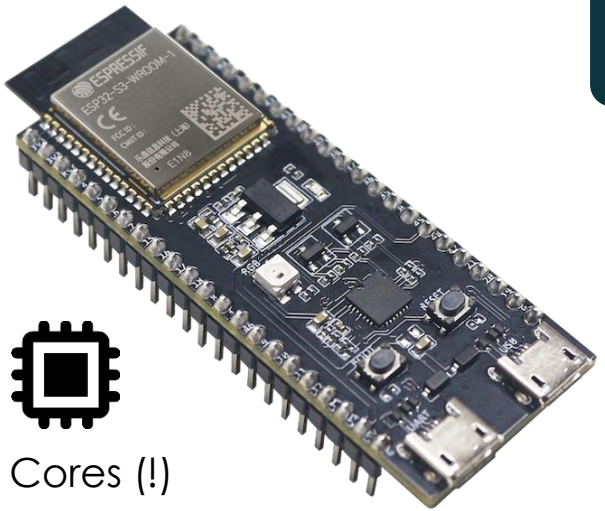
STM32 + ESP32 Microcontrollers
- **I2C** Protocol for Accelerometer
- **SPI** for Encoders
- **WiFi** for further-located modules
- **Wired** connection for coupled modules

MQTT Service for Pub/Sub with Central Controller (Desktop) and rest of modules



Belke, C.H., Holdcroft, K., Sigrist, A. et al. Morphological flexibility in robotic systems through physical polygon meshing. Nat Mach Intell 5, 669–675 (2023). https://doi.org/10.1038/s42256-023-00676-8

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?