System Architectures

System decomposition

- All starts with a decomposition of a system in components.
- What is a **system** and what are **components**?

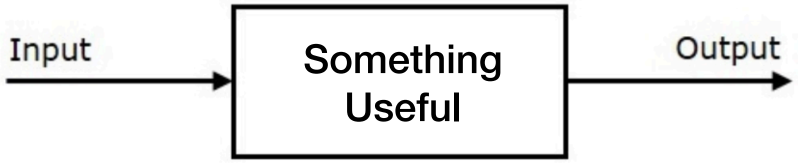
> **"A system is composed of components;**
> **a component is something you understand."**
>
> — Howard Aiken (1900-1973)

Physical and logical architectures

- There are **two aspects** for “architectures”:
  - **Logical** architecture - **what** the system is doing
    - e.g., system decomposition, data flow
  - **Physical** architecture - **how** it is doing it
    - e.g., which computer runs which component

- **Containerization**: Which computer runs which virtual computer that runs which component?

Logical architecture



- The logical architecture describes:
  - **System decomposition** in components
  - Data flow (**Who tells whom what**)
    - Representations
  - Priors (**Who knows what**)

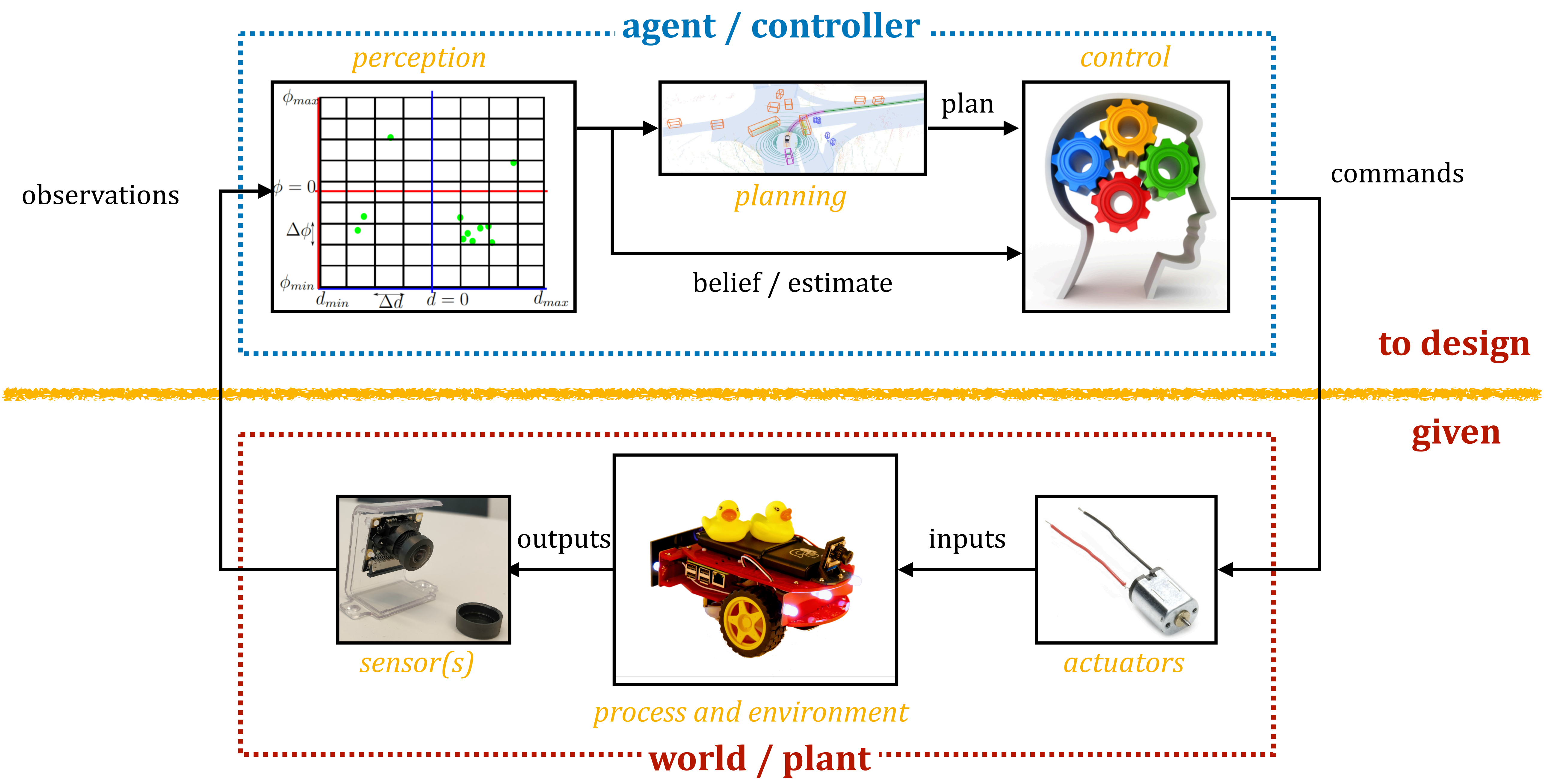
- The logical architecture is <u>independent of language, middleware, and other implementation details.</u>

The LINE

to design

given

Logical architecture overview



agent / controller

perception

planning

control

plan

belief / estimate

observations

commands

to design

given

world / plant

sensor(s)

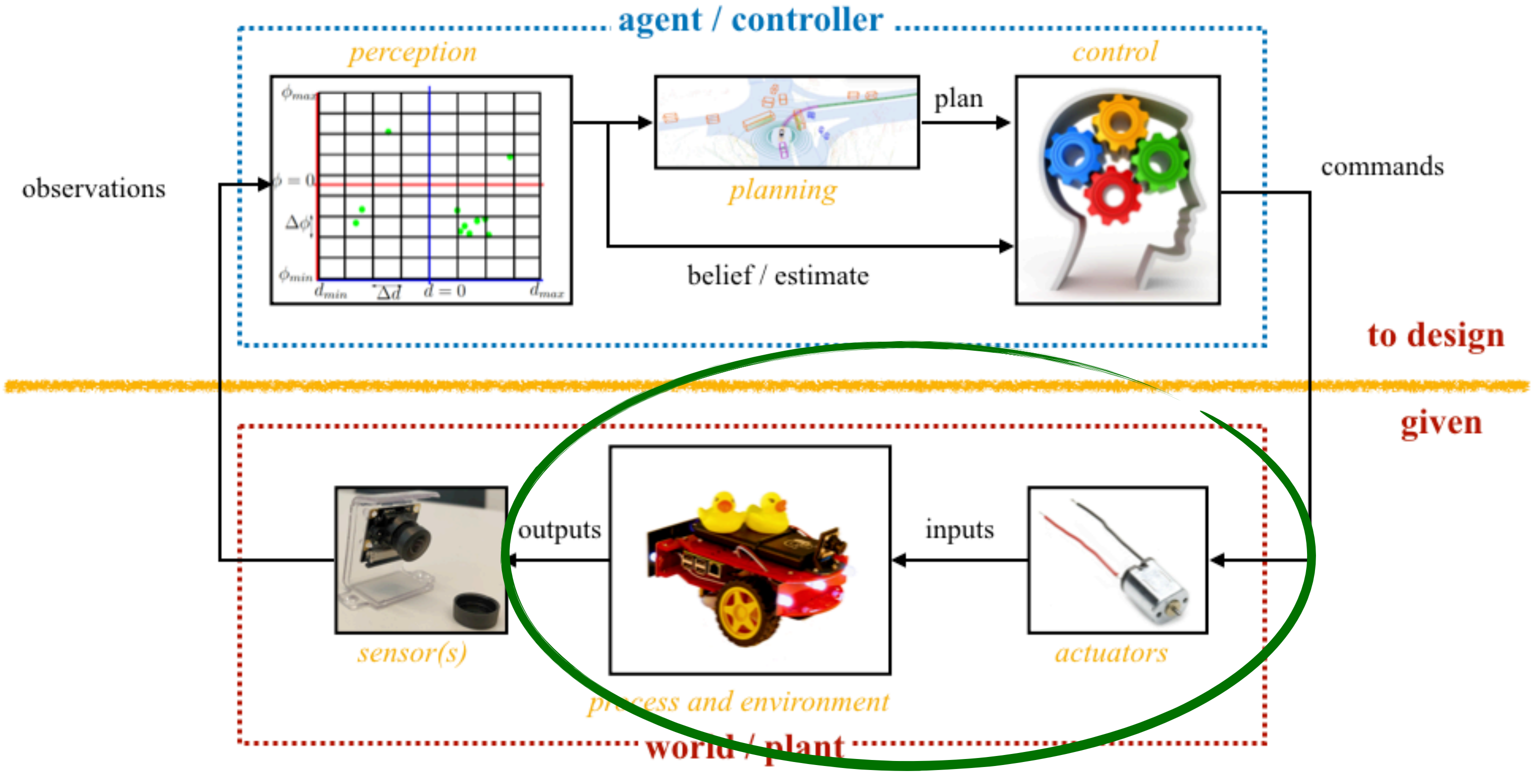
process and environment

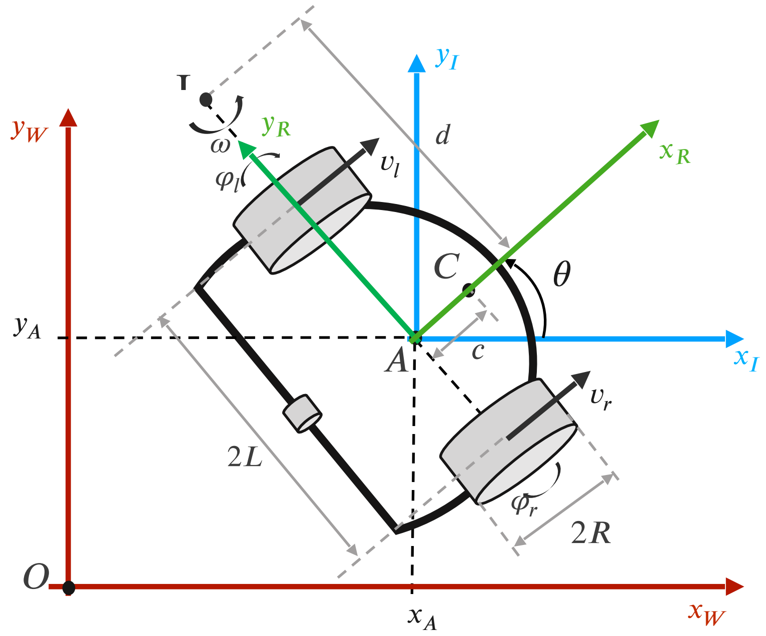
actuators

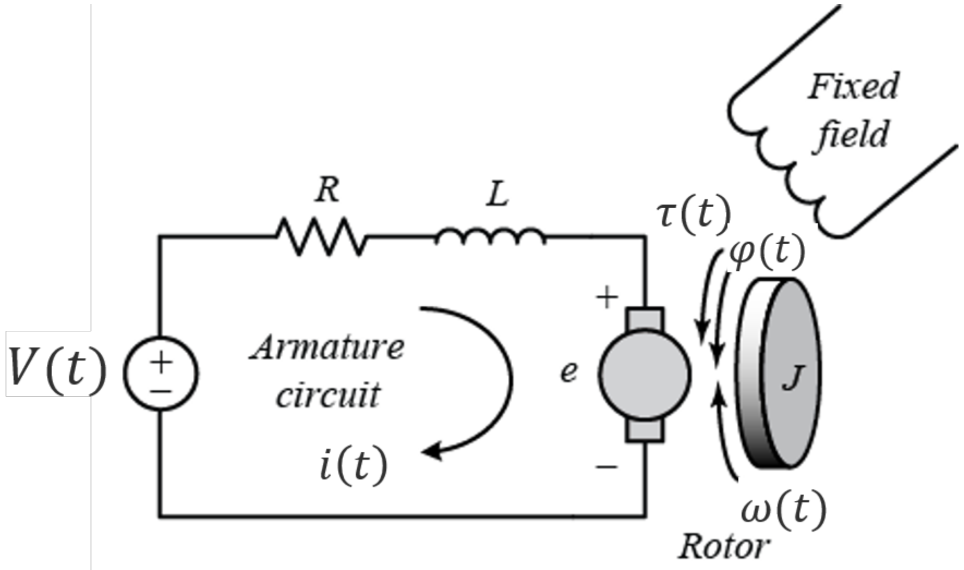
outputs

inputs

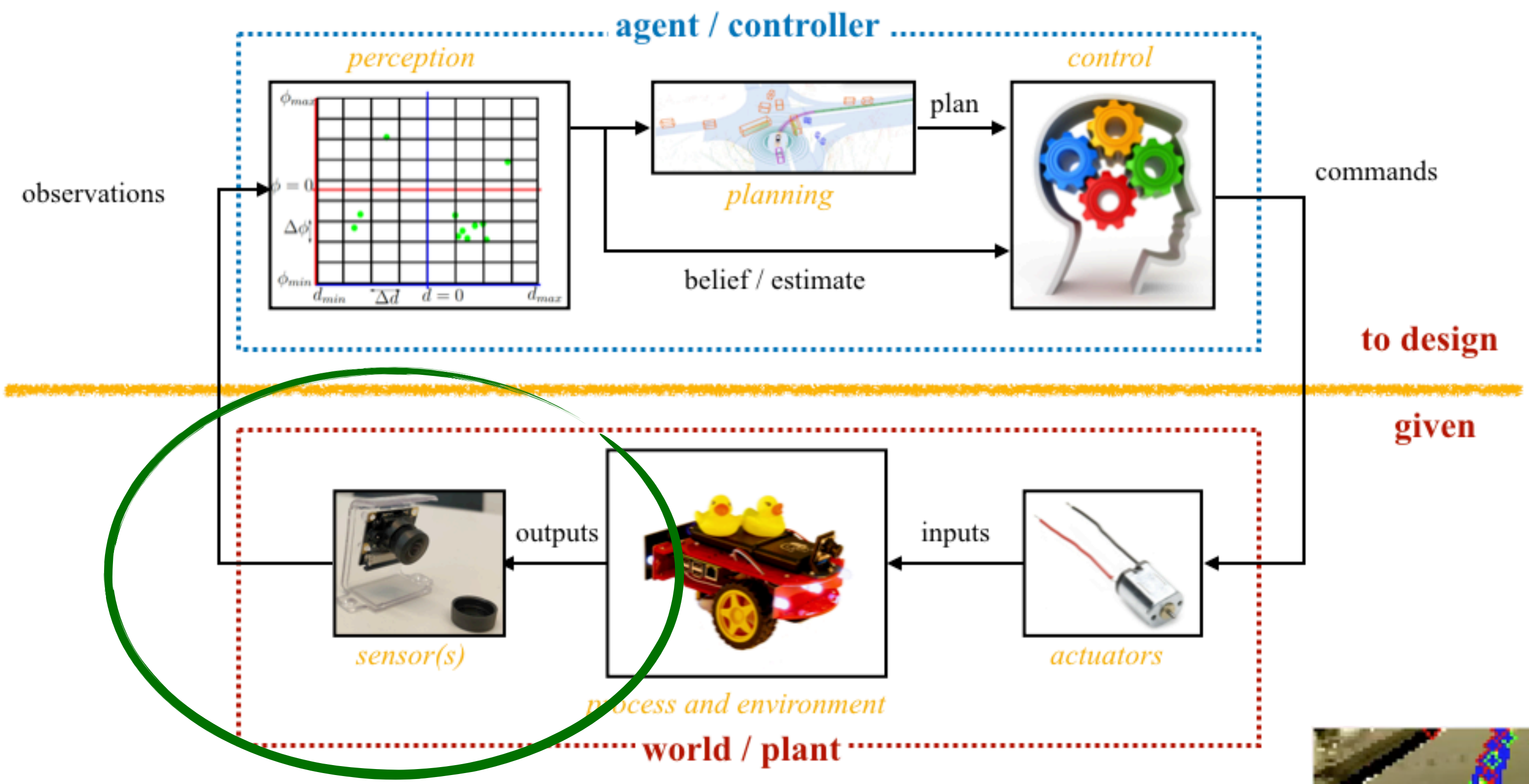
Modeling

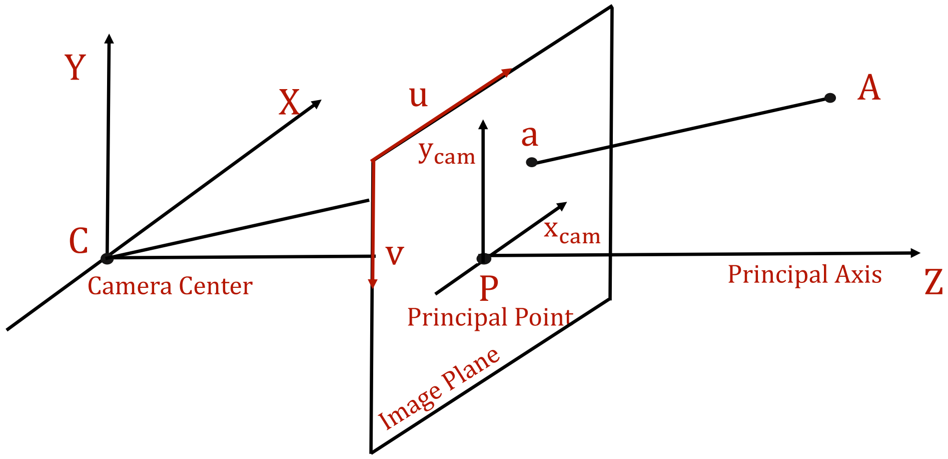

agent / controller
perception
planning
plan
control
observations
belief / estimate
commands
to design
given
outputs
inputs
sensor(s)
actuators
process and environment
world / plant

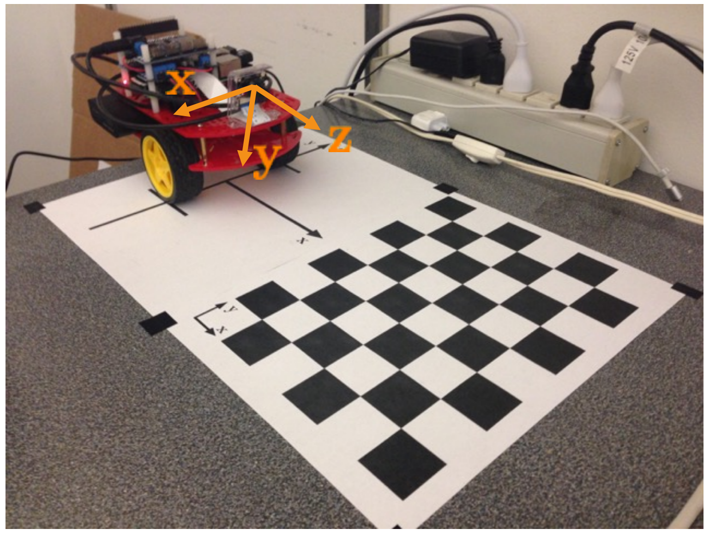

y_W
y_I
y_R
x_R
x_I
x_W
y_A
x_A
O
A
C
c
d
θ
ω
φ_l
φ_r
v_l
v_r
$2L$
$2R$

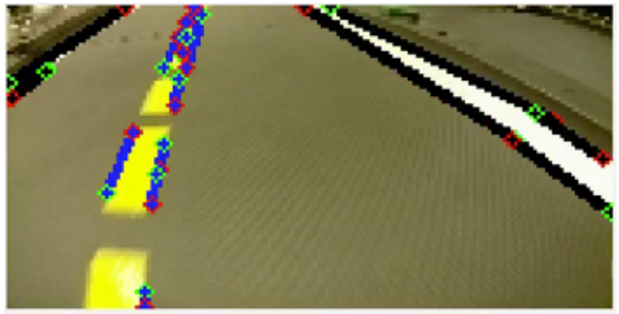

Fixed field
R
L
$\tau(t)$
$\varphi(t)$
$V(t)$
+
−
Armature circuit
$i(t)$
e
J
$\omega(t)$
Rotor

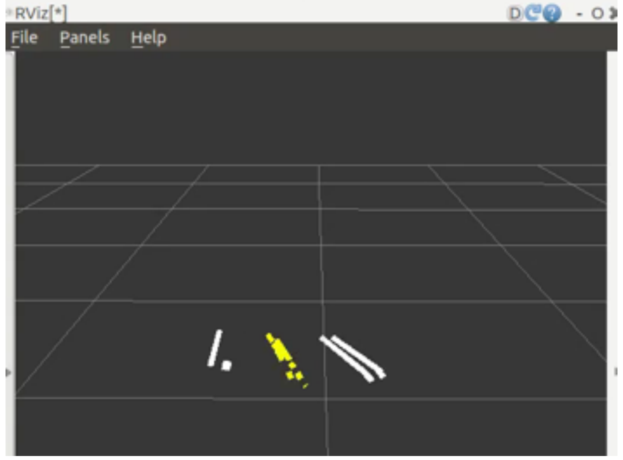
Sensing


agent / controller
perception
ϕ_{max}
$\phi = 0$
$\Delta\phi$
ϕ_{min}
d_{min}
Δd
$d = 0$
d_{max}
planning
plan
control
belief / estimate
observations
commands
to design
given
sensor(s)
outputs
process and environment
inputs
actuators
world / plant

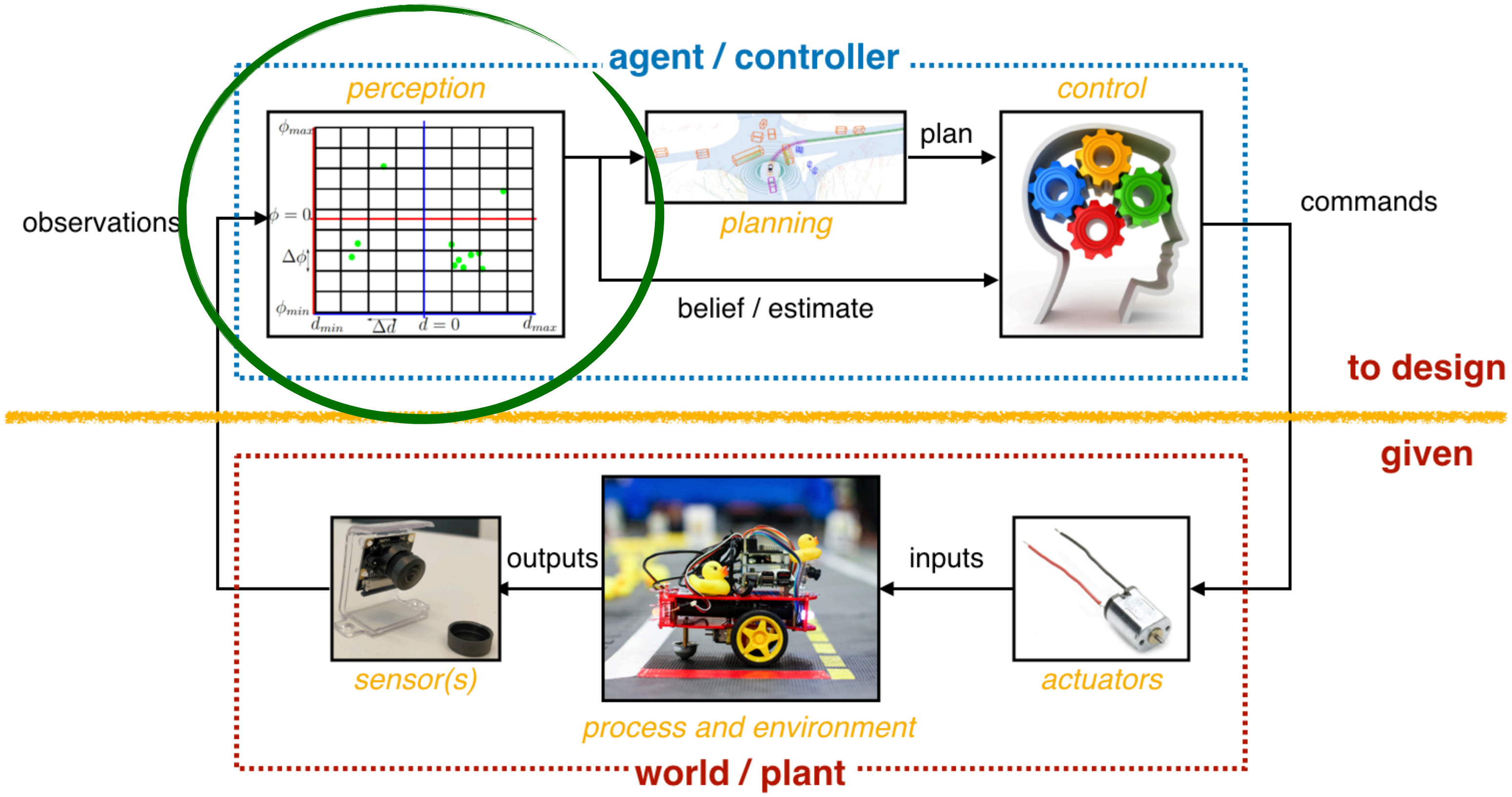

Y
X
C
Camera Center
u
y_{cam}
v
a
x_{cam}
P
Principal Point
Image Plane
A
Principal Axis
Z

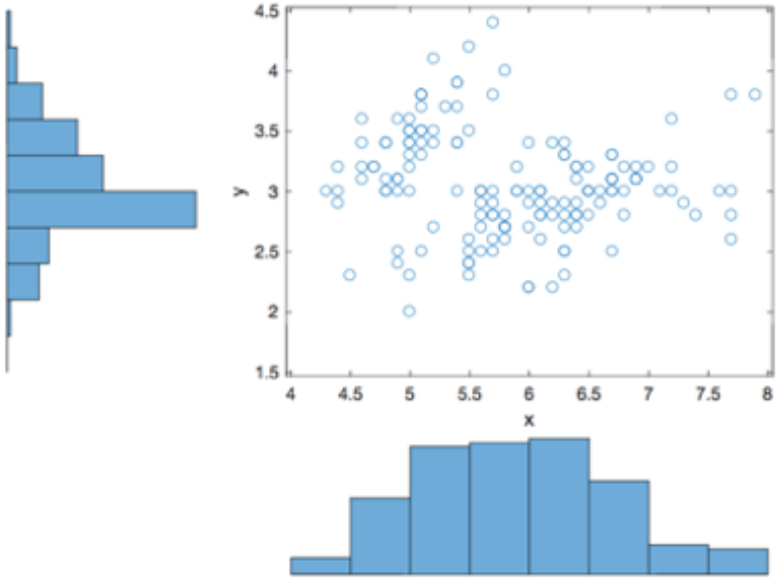

x
y
z

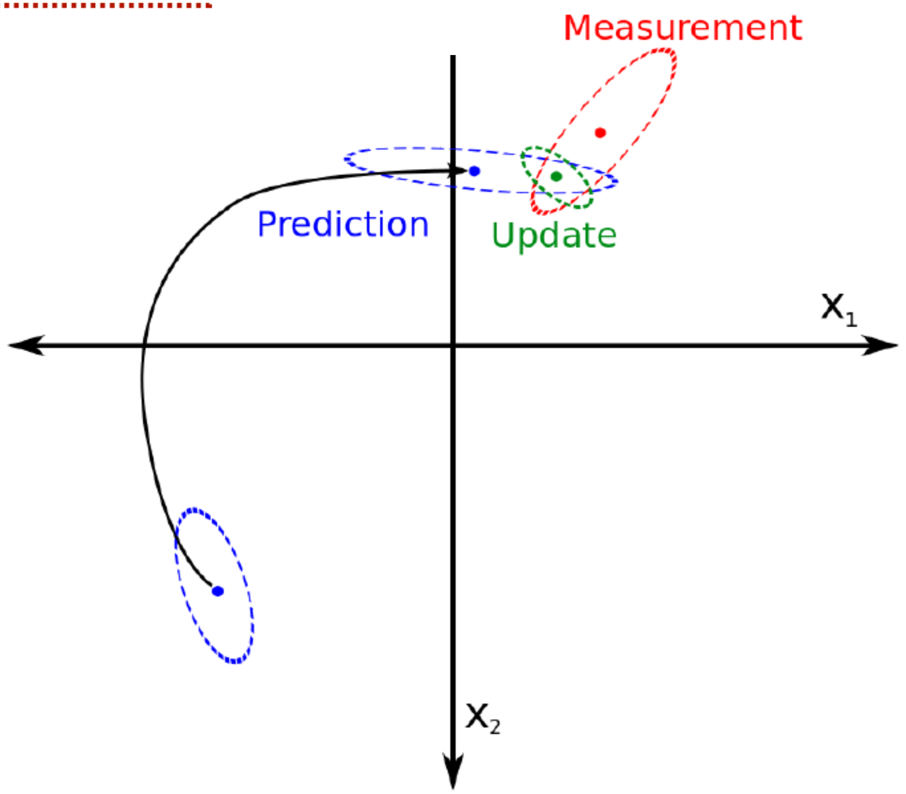



RViz[*]
File
Panels
Help

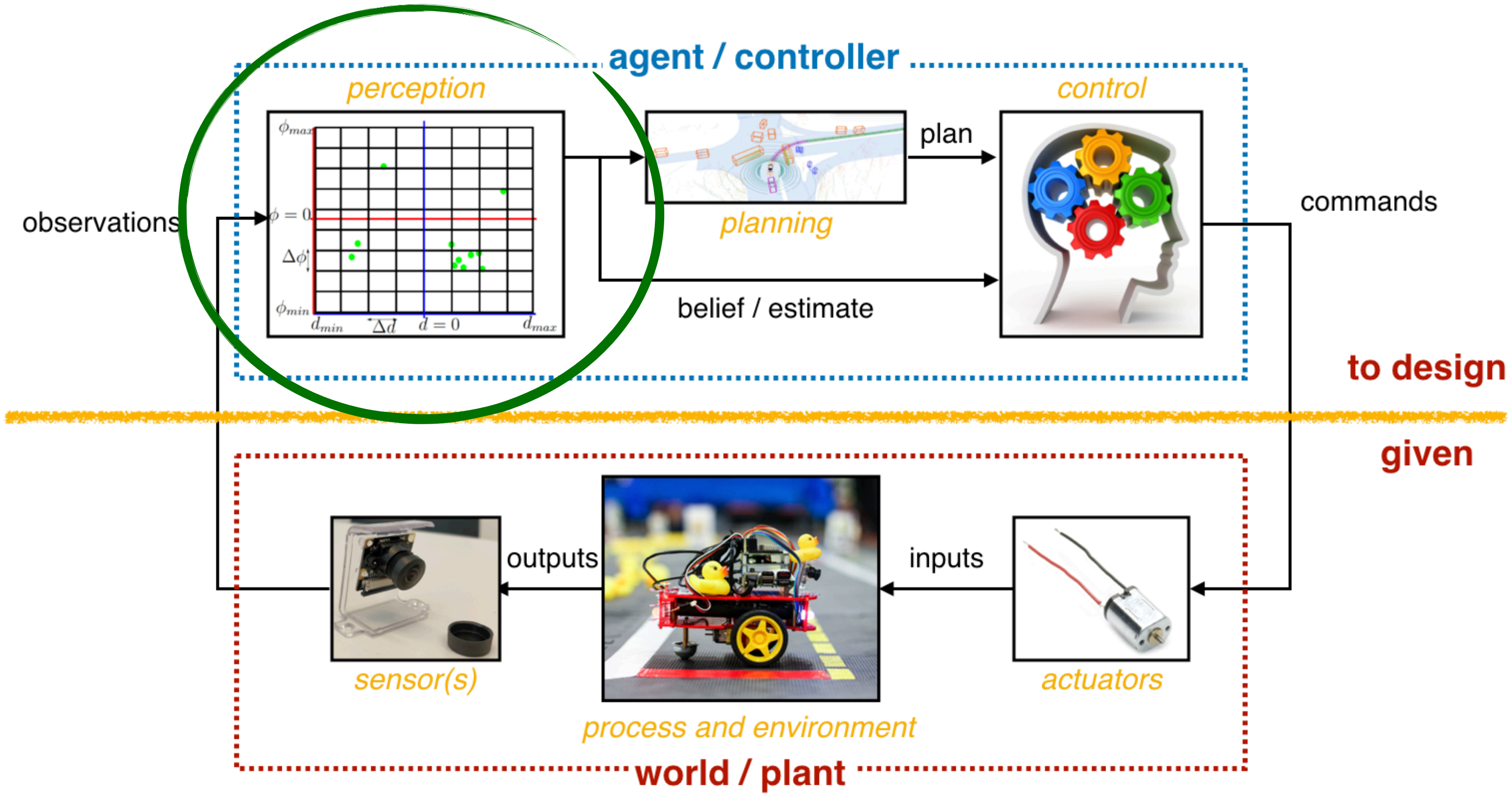
Estimation: Probability basics, Bayesian filtering

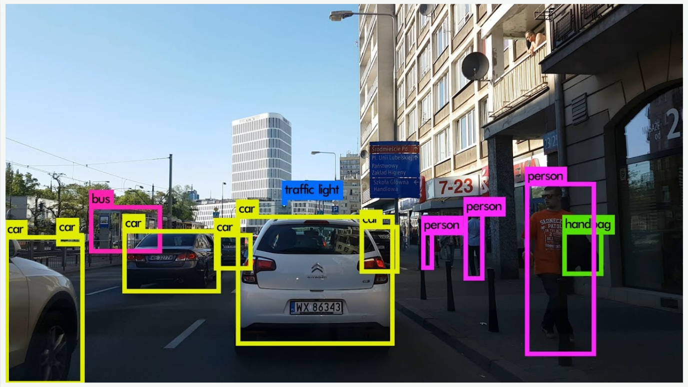


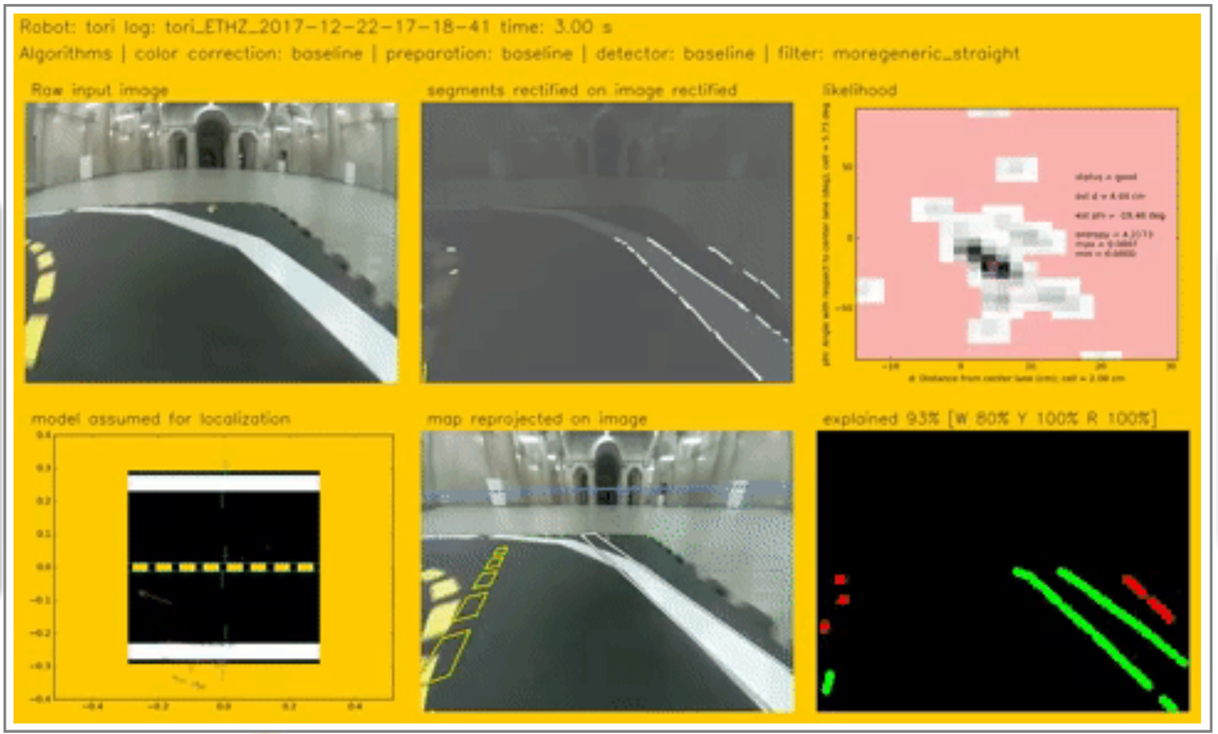




Perception



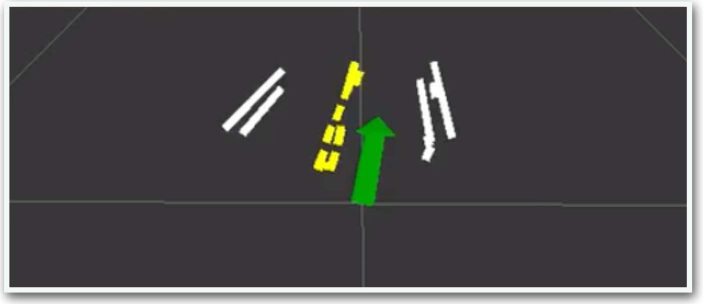




Robot: tori log: tori_ETHZ_2017-12-22-17-18-41 time: 3.00 s

Algorithms | color correction: baseline | preparation: baseline | detector: baseline | filter: moregeneric_straight

Raw input image
segments rectified on image rectified
likelihood
model assumed for localization
map reprojected on image
explained 93% [W 80% Y 100% R 100%]



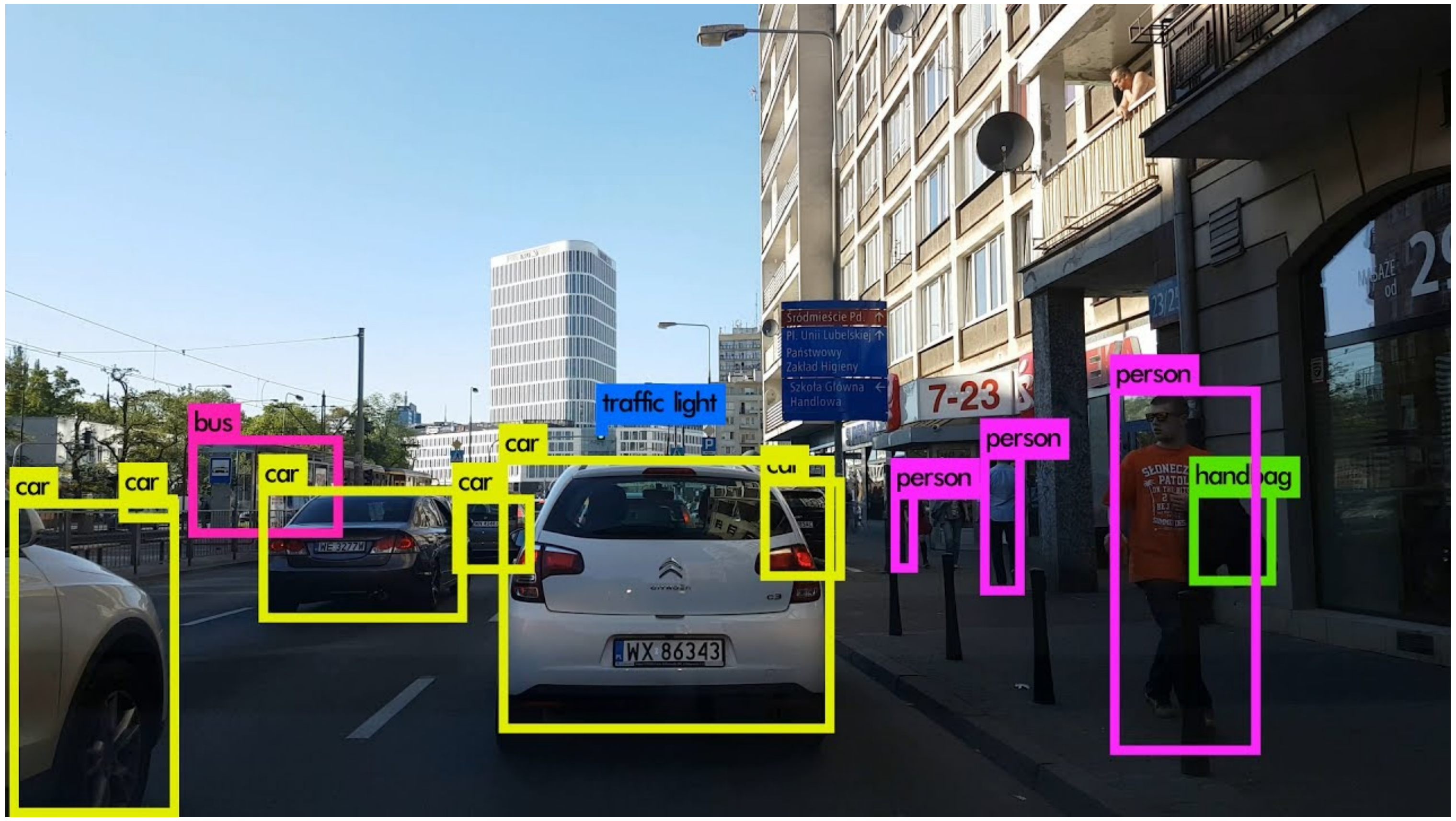
The basic principle of perception

- You should acquire (all and only) **“actionable information”.**
- **Actionable information**: the information that is needed to perform the task.
    - It is **task-dependent.**
- **Non-actionable information:** irrelevant to the task.
    - Processing it is a waste of time and resources.

	Information		
Task	weather	class location	referendum results
Dress up to go to class	actionable	not actionable	not actionable
Reach home from class	not actionable	actionable	not actionable

Self-driving car - actionable information

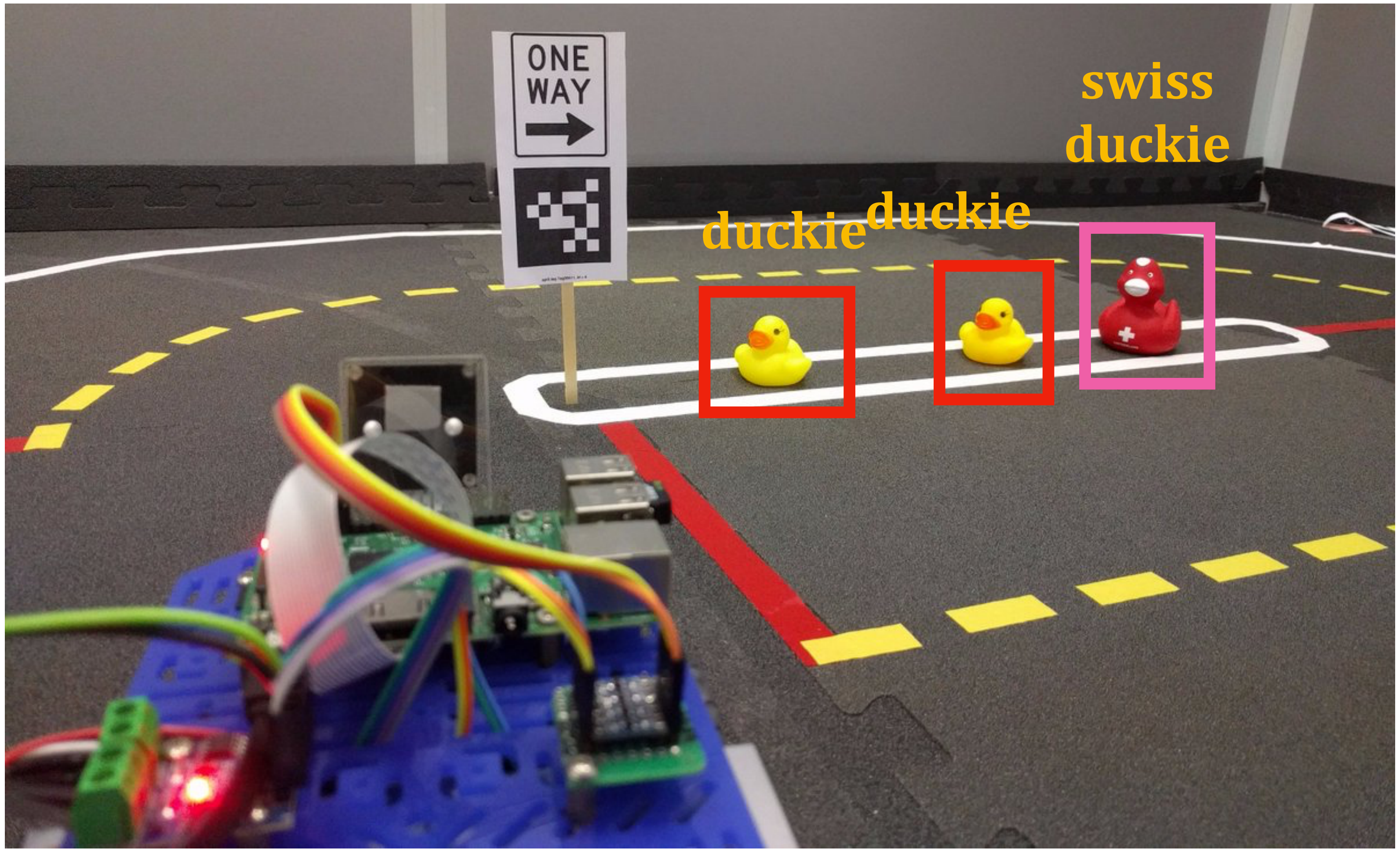
- Cars
- Pedestrians
- ...



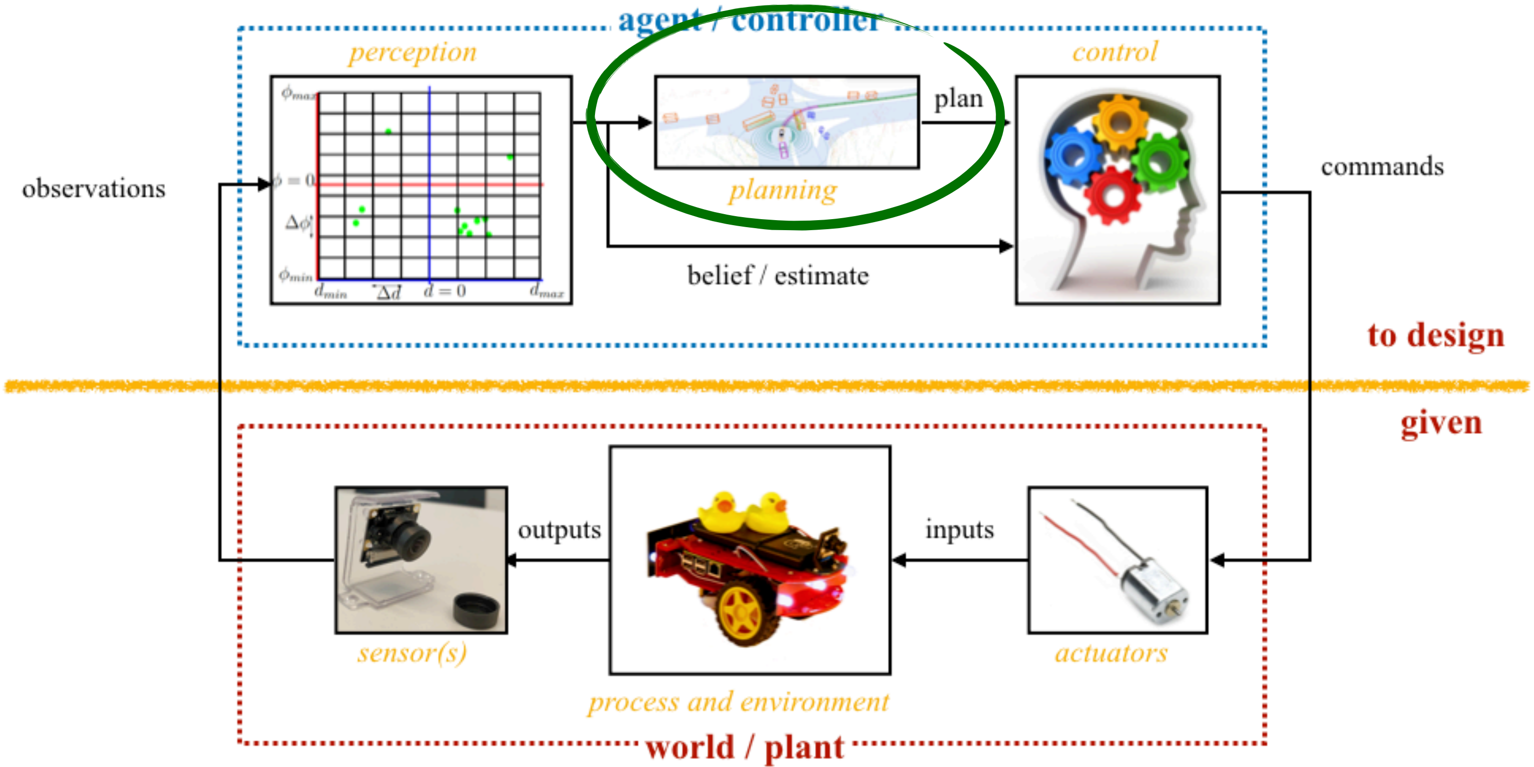
https://www.youtube.com/watch?v=OksuVuNY5o0

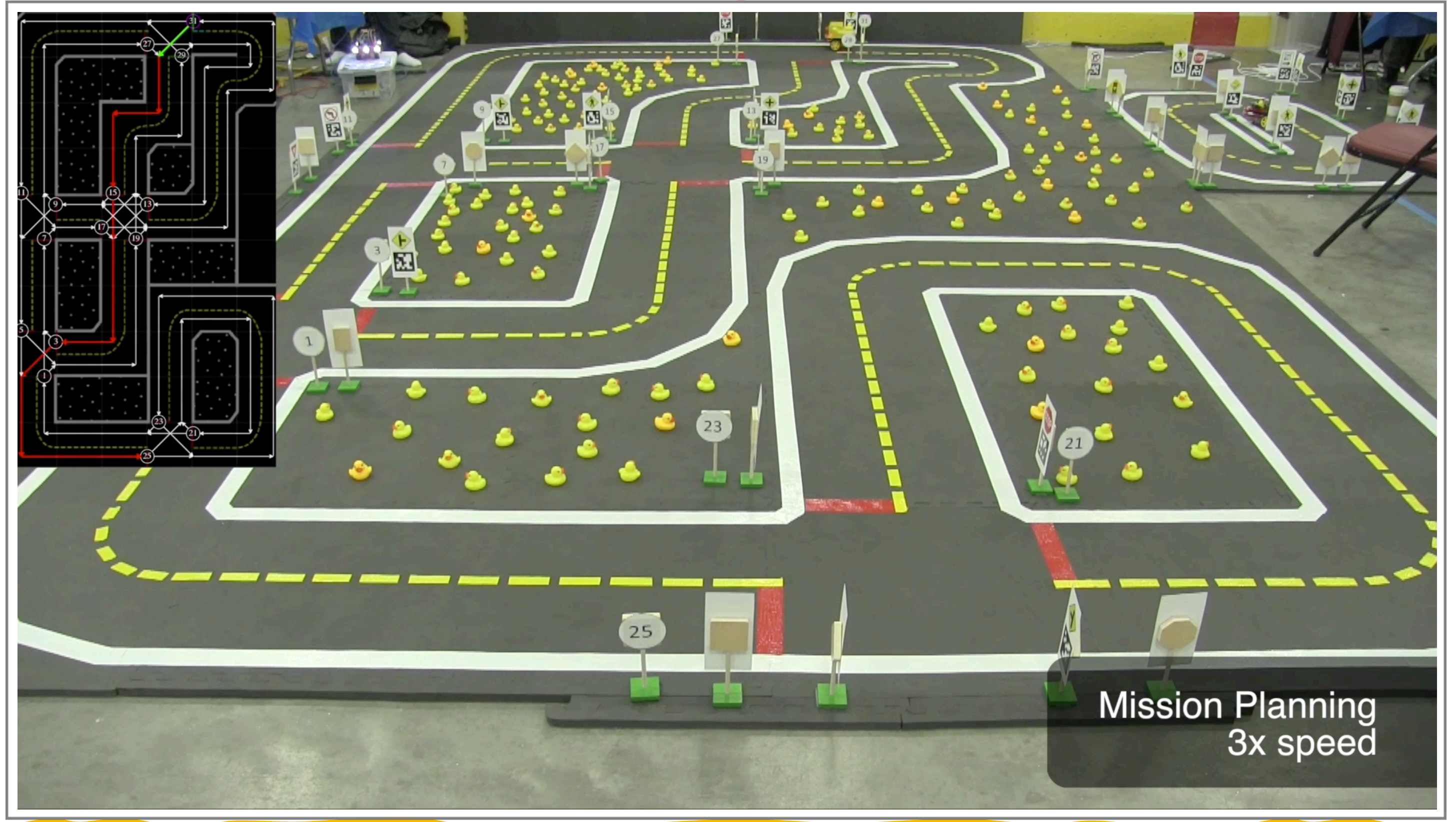
Duckiebot - actionable information

- Duckiebots
- Duckies
- ...

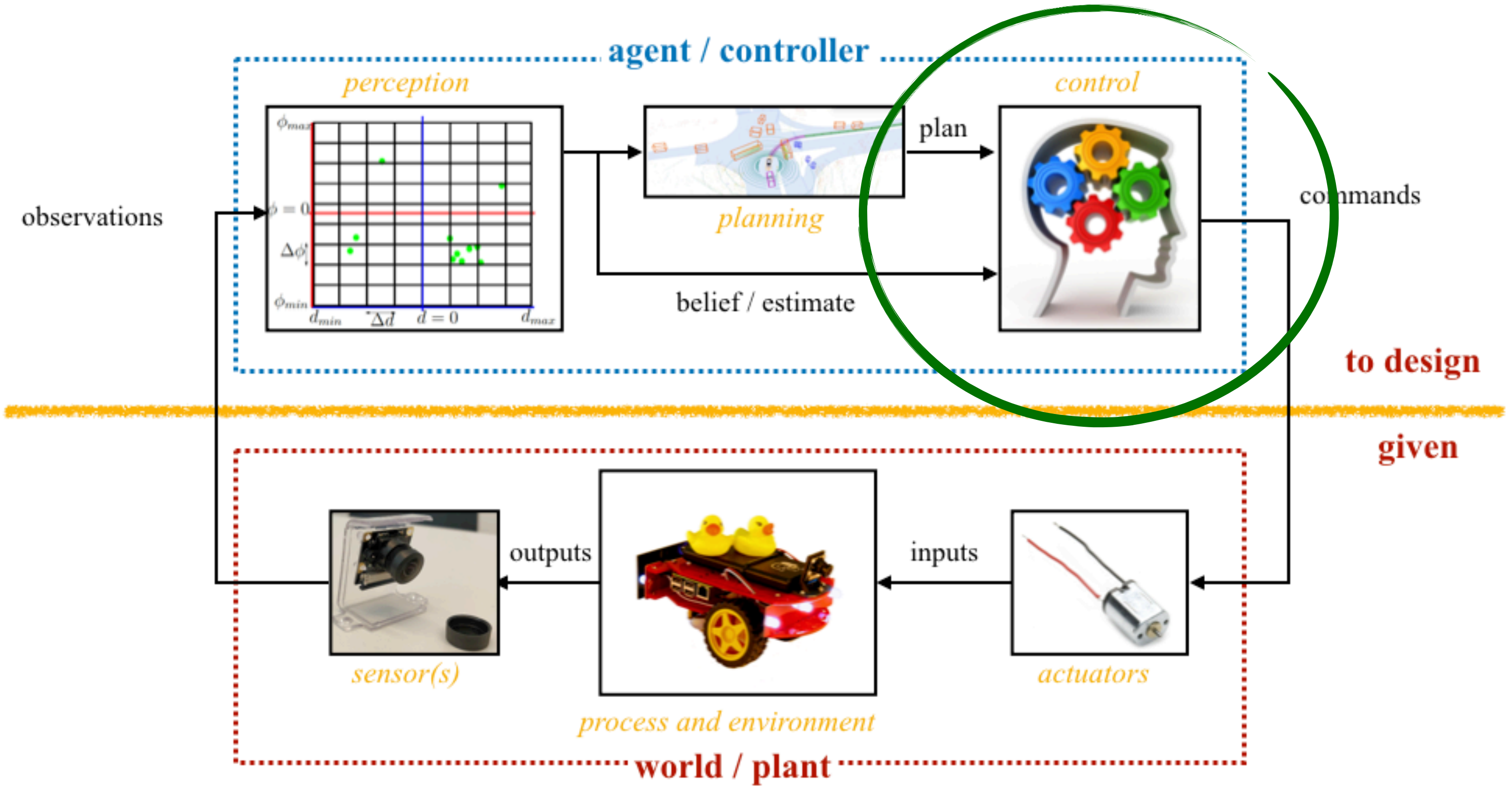


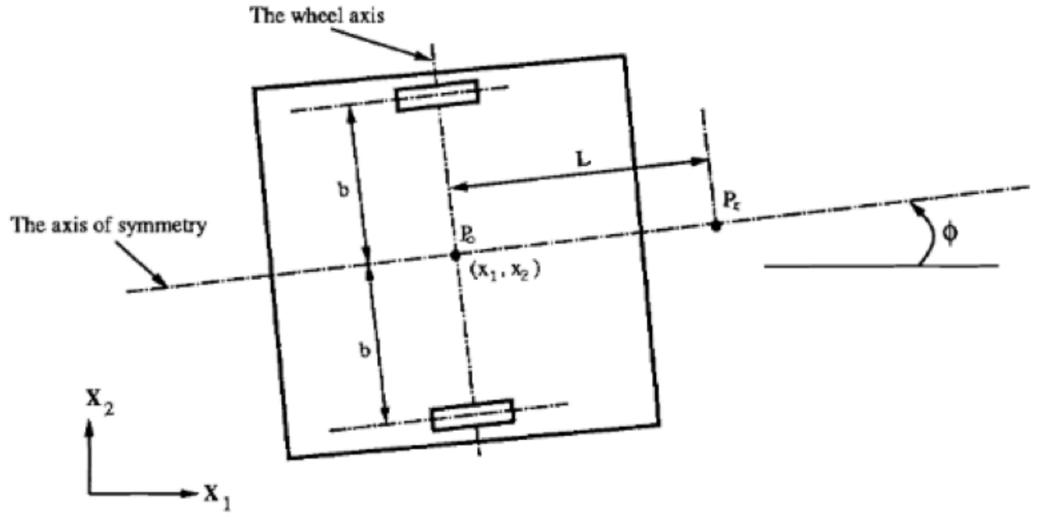
Planning

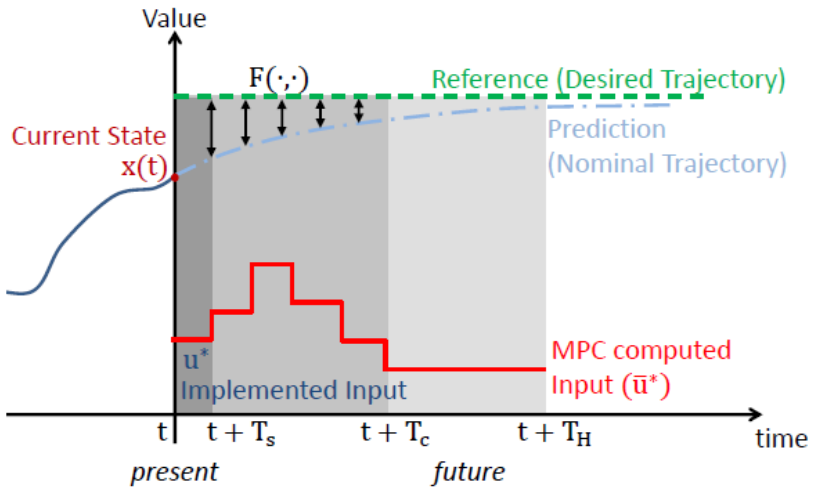

agent / controller
perception
control
observations
plan
planning
commands
belief / estimate
ϕ_{max}
$\phi = 0$
$\Delta\phi$
ϕ_{min}
d_{min}
Δd
$d = 0$
d_{max}
to design
given
outputs
inputs
sensor(s)
actuators
process and environment
world / plant

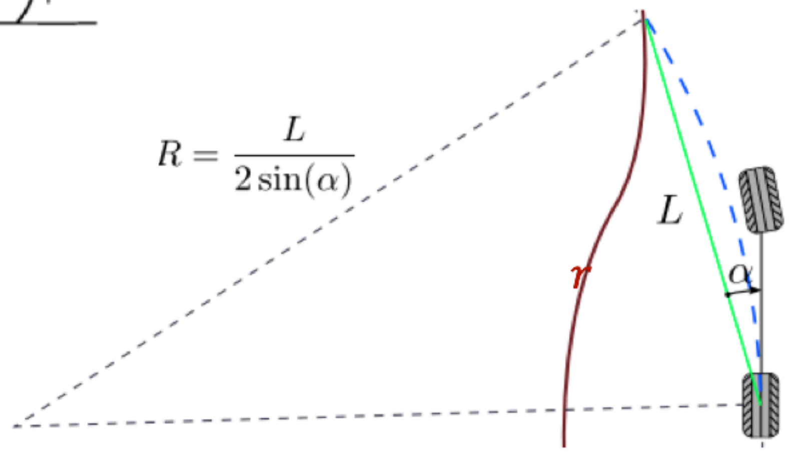

25
23
21
Mission Planning
3x speed

Control


agent / controller
perception
planning
plan
control
observations
belief / estimate
commands
to design
given
outputs
inputs
sensor(s)
process and environment
actuators
world / plant

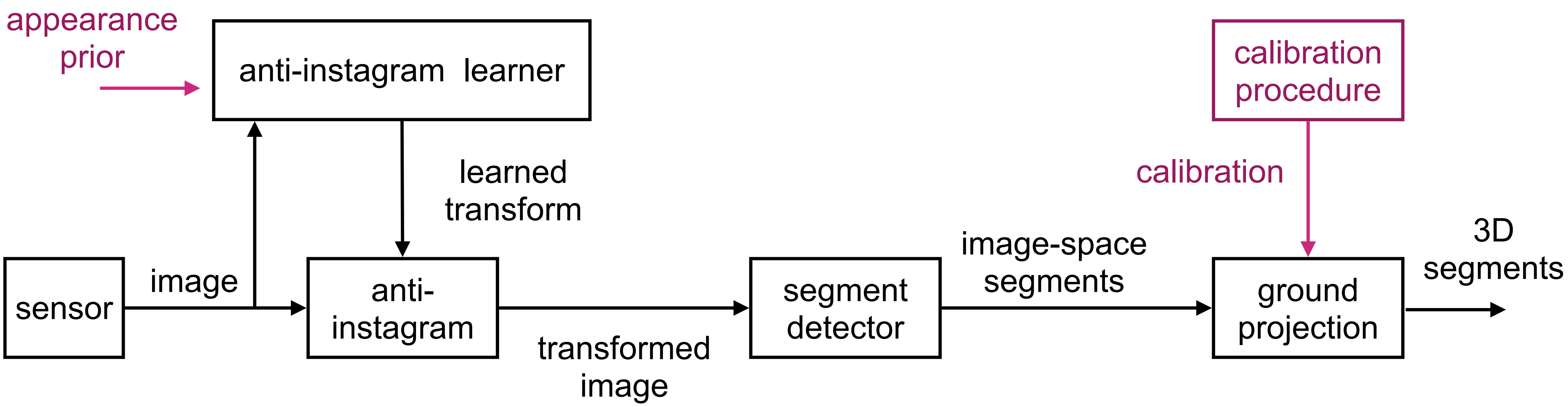

The wheel axis
The axis of symmetry
b
b
L
P_r
P_o
(x_1, x_2)
ϕ
X_2
X_1


Value
F(·,·)
Reference (Desired Trajectory)
Current State
x(t)
Prediction
(Nominal Trajectory)
u*
Implemented Input
MPC computed
Input ($\bar{u}^*$)
t
$t + T_s$
$t + T_c$
$t + T_H$
time
present
future

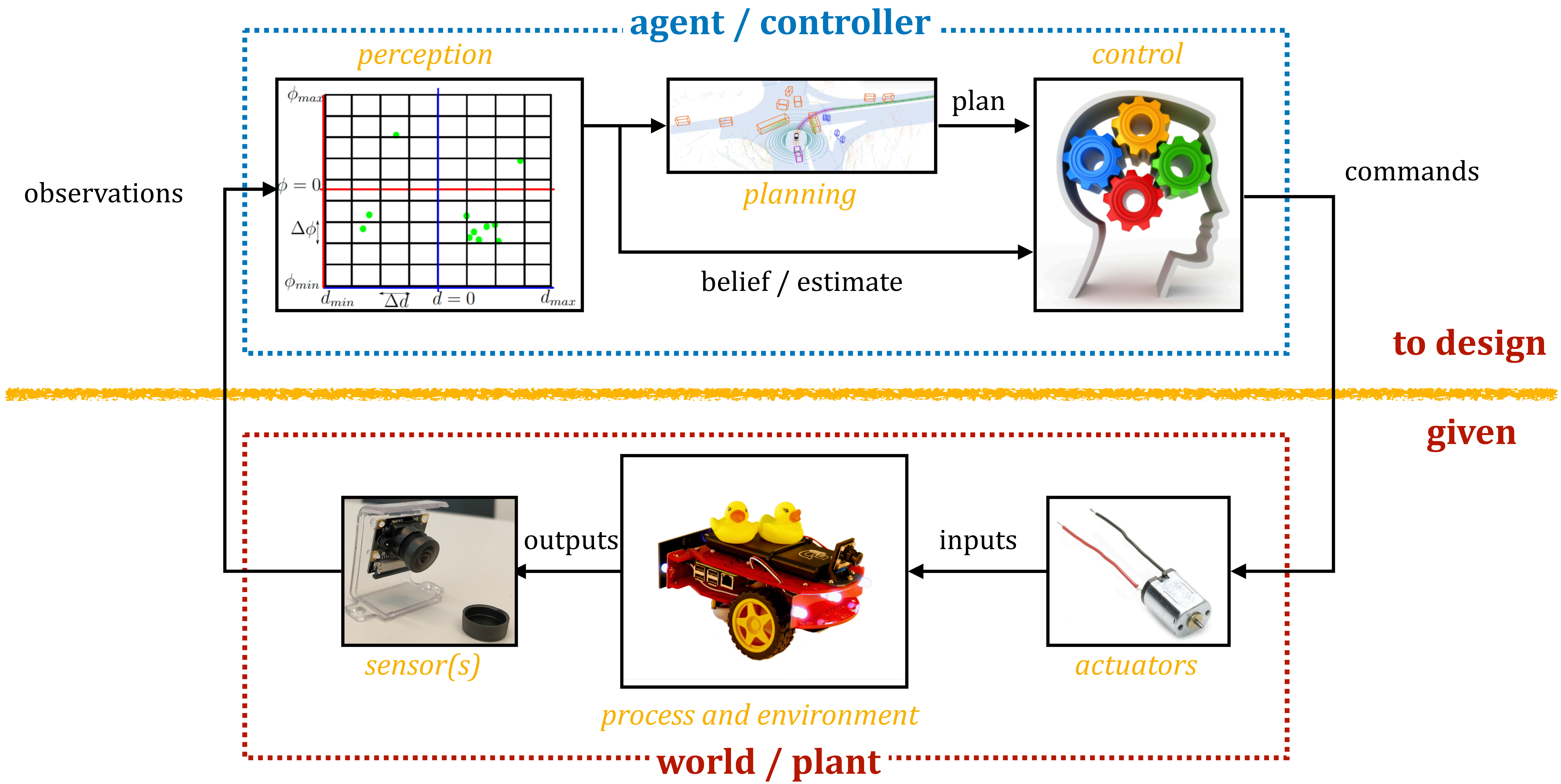

$R = \dfrac{L}{2\sin(\alpha)}$
L
α
r

Part of logical architecture for Duckietown

- Part of the main perception pipeline.
- Purple refers to “static” information.
- The diagram does not show how things are implemented.
- It does show who-knows-what, and who-tells-what-to-whom.



Logical Architecture



agent / controller

perception

planning

plan

control

belief / estimate

observations

commands

to design

given

outputs

inputs

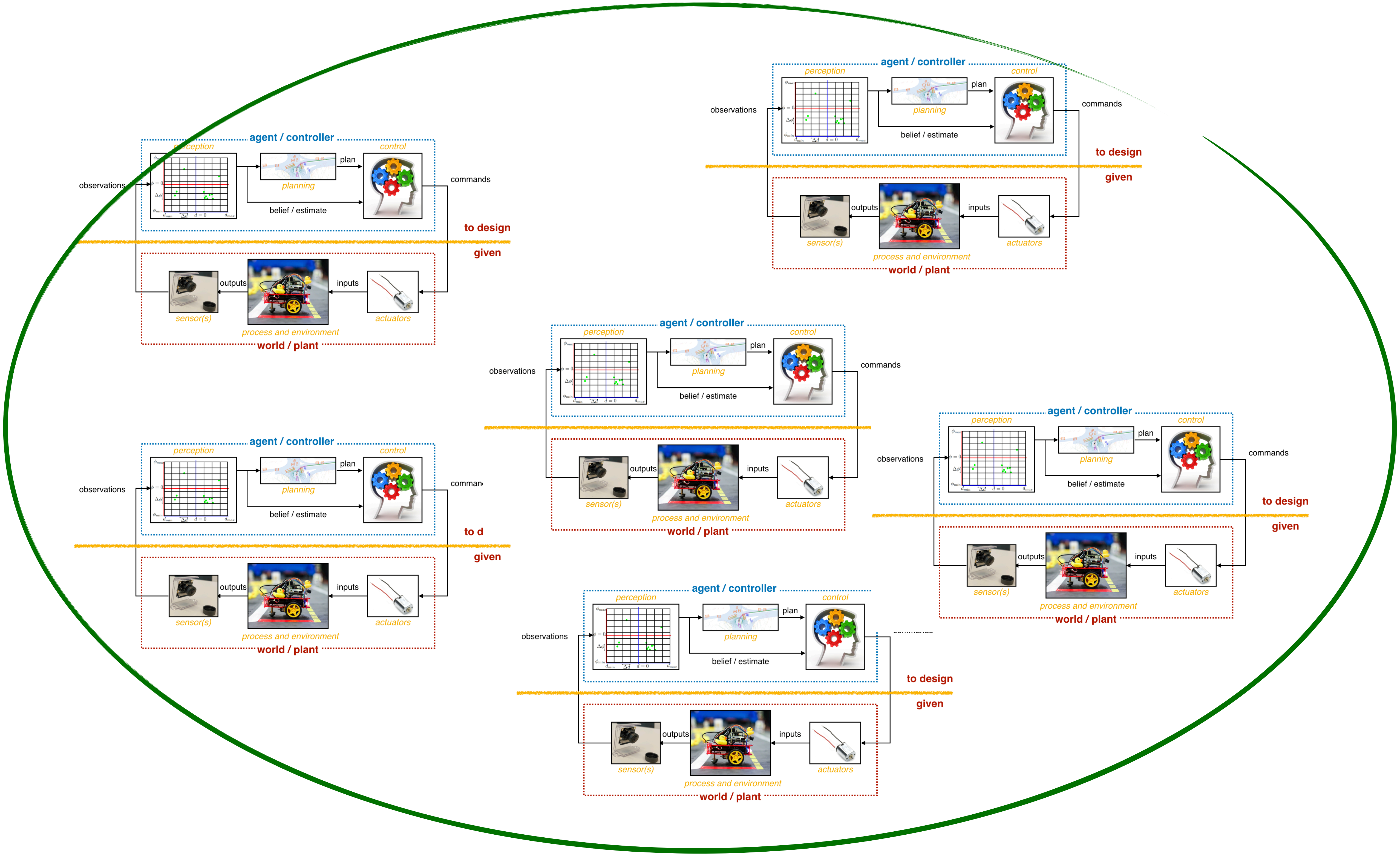
sensor(s)

process and environment

actuators

world / plant

... for a fleet

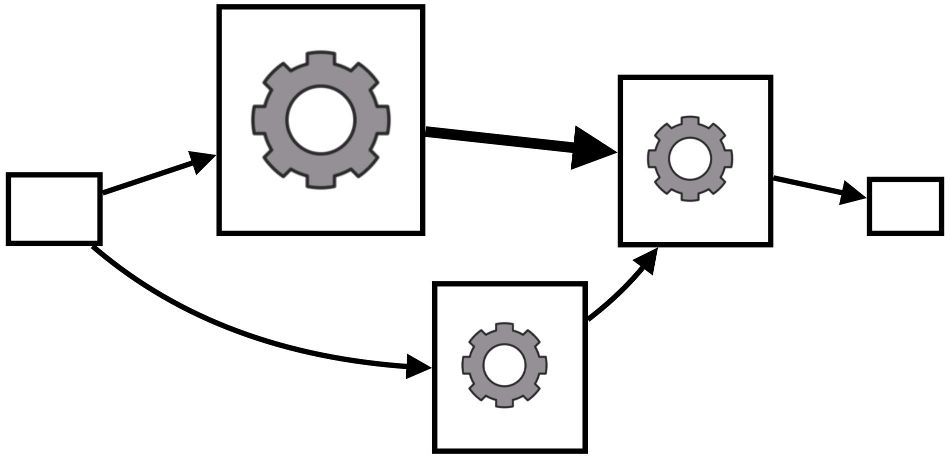

agent / controller
perception
planning
plan
control
belief / estimate
observations
commands
to design
given
outputs
inputs
sensor(s)
process and environment
actuators
world / plant

Physical Architecture

- “Physical” architecture: **how** things are implemented.
- The physical architecture includes:
    - **Which processor** runs what process?
    - **How is the data communicated** (TCP, UDP, etc.)
    - **Where is the data stored**
    - Protocols, formats, etc.

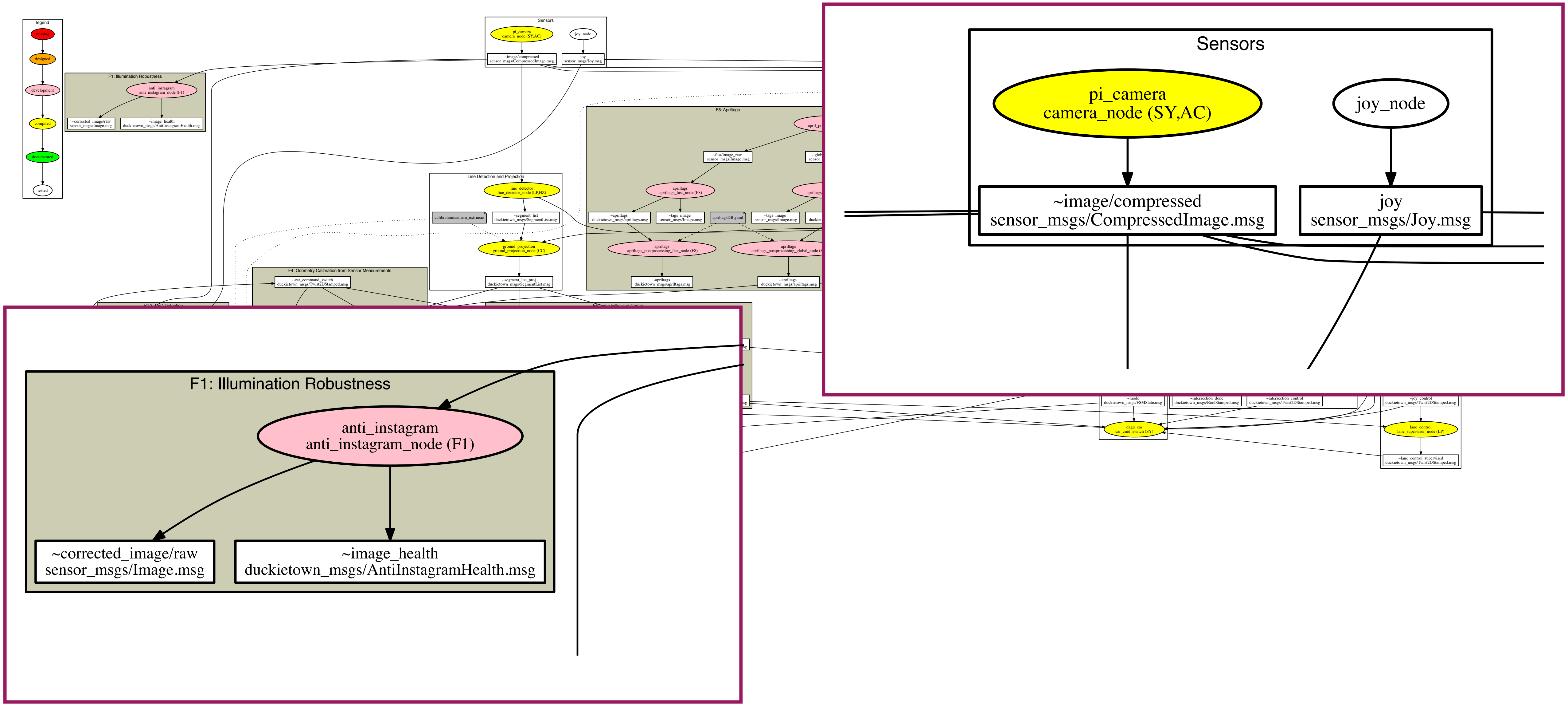
Computation graph

- **Computation graph**
  - nodes = components
  - edges = signals
  - node weight = flops required
  - edge weight = size in bytes



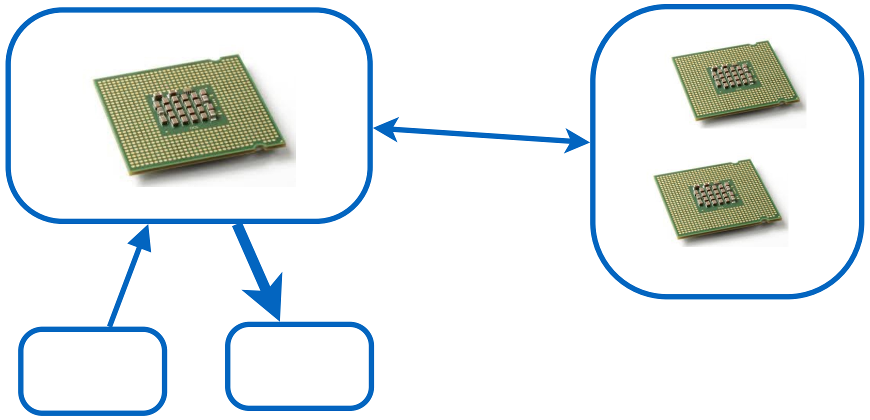
Part of computation graph of Duckietown

- This is the Duckietown ROS graph
- It lists the nodes and the signals.


legend
Sensors
F1: Illumination Robustness
Line Detection and Projection
F8: Apriltags
F4: Odometry Calibration from Sensor Measurements
Sensors
pi_camera
camera_node (SY,AC)
joy_node
~image/compressed
sensor_msgs/CompressedImage.msg
joy
sensor_msgs/Joy.msg
F1: Illumination Robustness
anti_instagram
anti_instagram_node (F1)
~corrected_image/raw
sensor_msgs/Image.msg
~image_health
duckietown_msgs/AntiInstagramHealth.msg

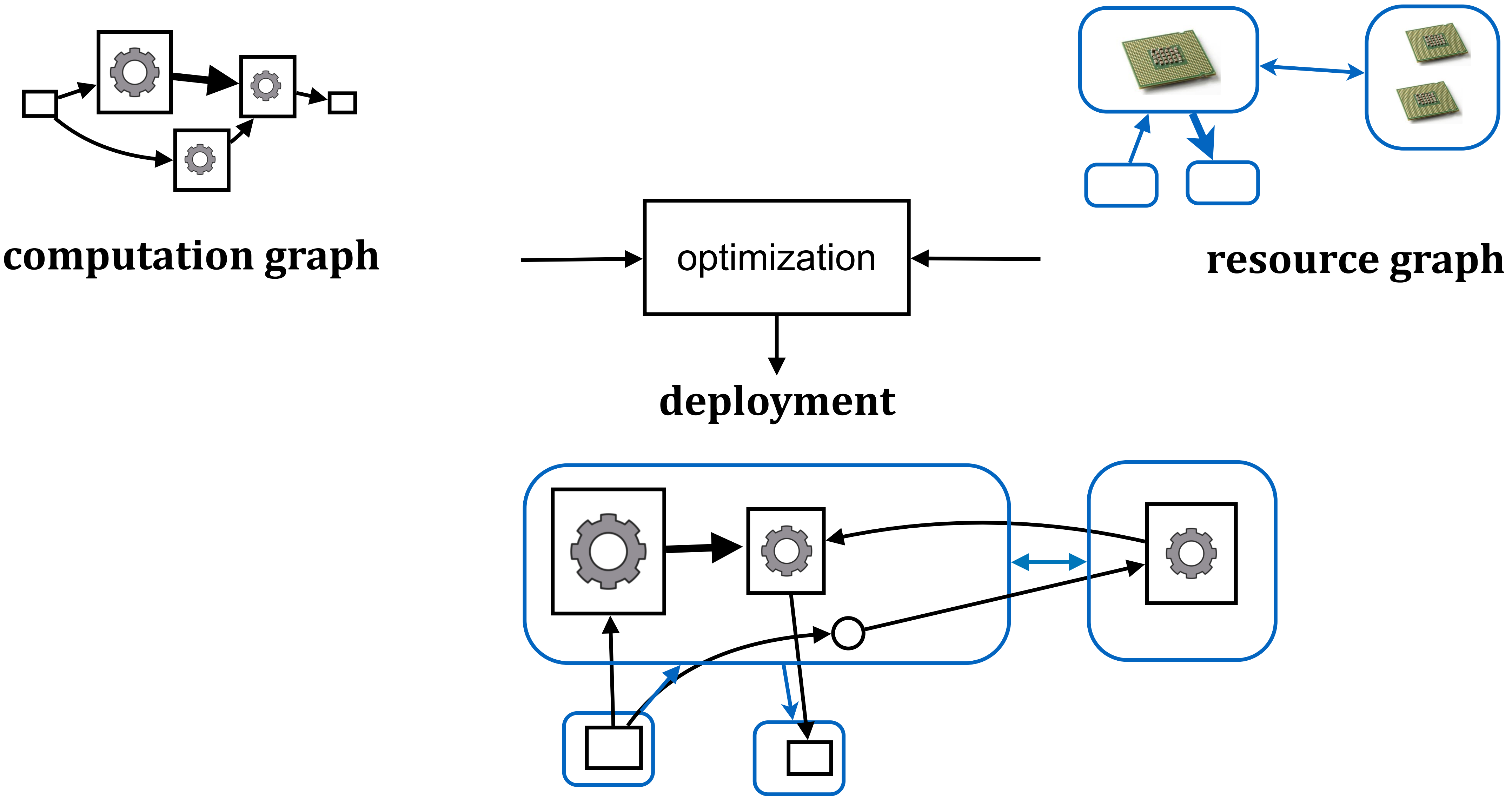
Physical architecture: Resource graph

- **Resource graph:**
  - nodes = processors
  - edges = network links
  - node weight = processor power
  - edge weight = bandwidth



Deployment: Mapping logical architecture onto physical

- We need to map the computation graph onto the resource graph.
- Different choices will have different properties for latency, frequency, etc.



computation graph

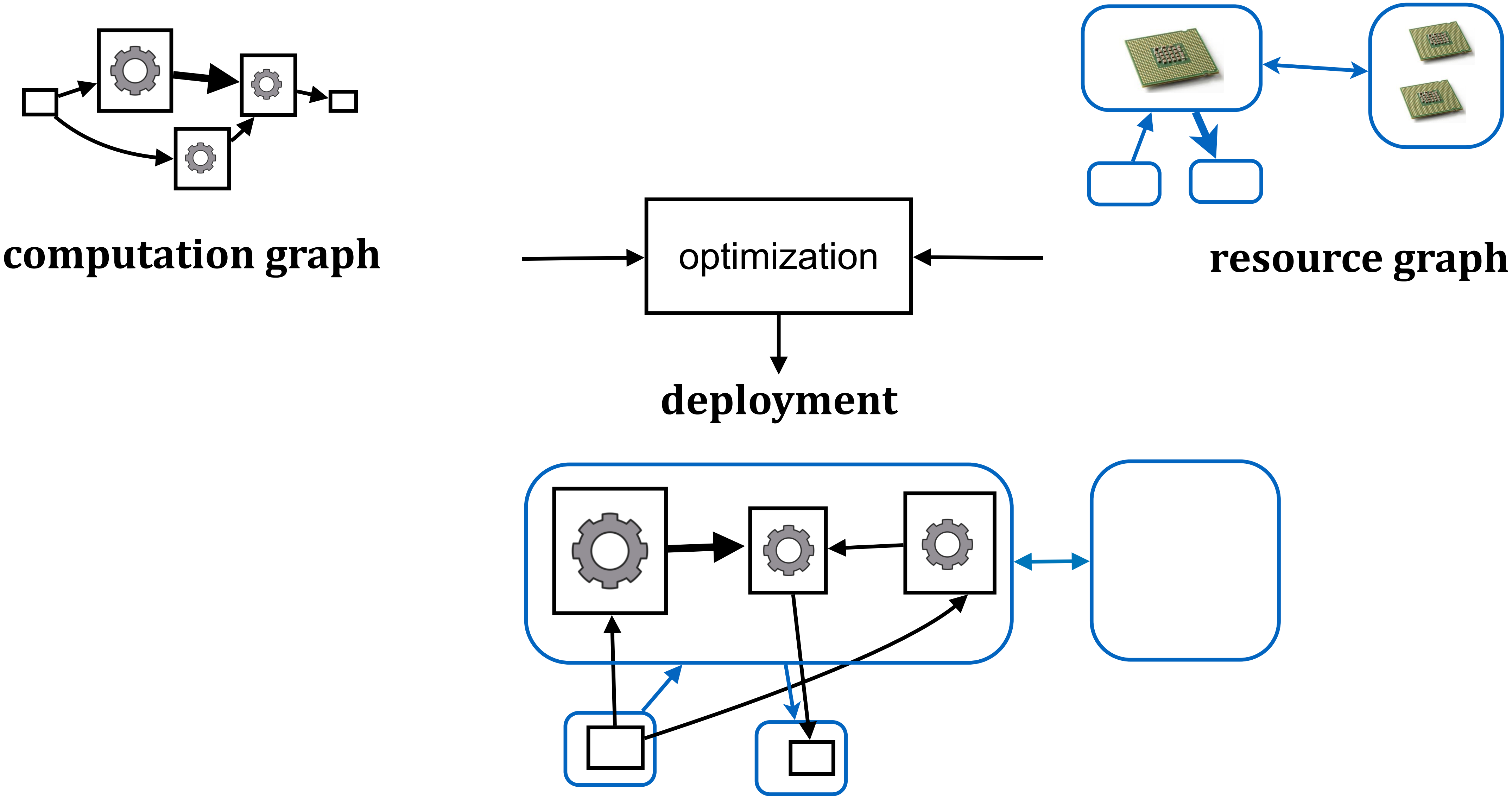
optimization

resource graph

deployment

Deployment: Mapping logical architecture onto physical

- We need to map the computation graph onto the resource graph.
- Different choices will have different properties for latency, frequency, etc.



computation graph

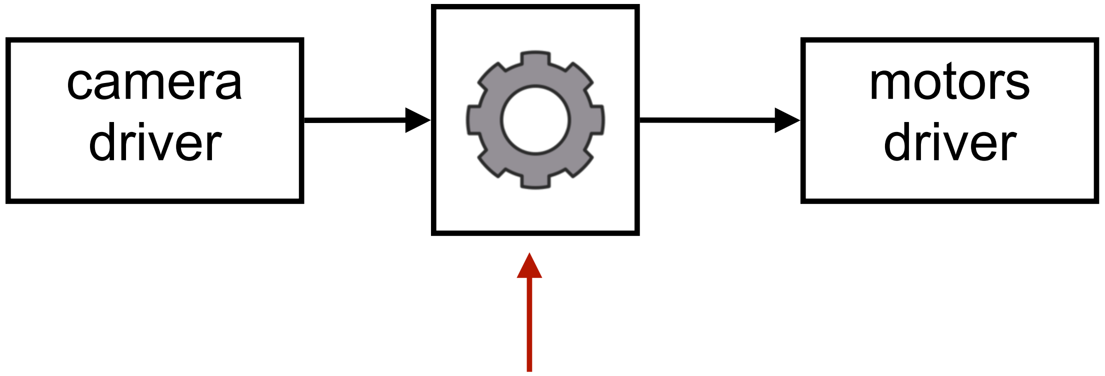
optimization

resource graph

deployment

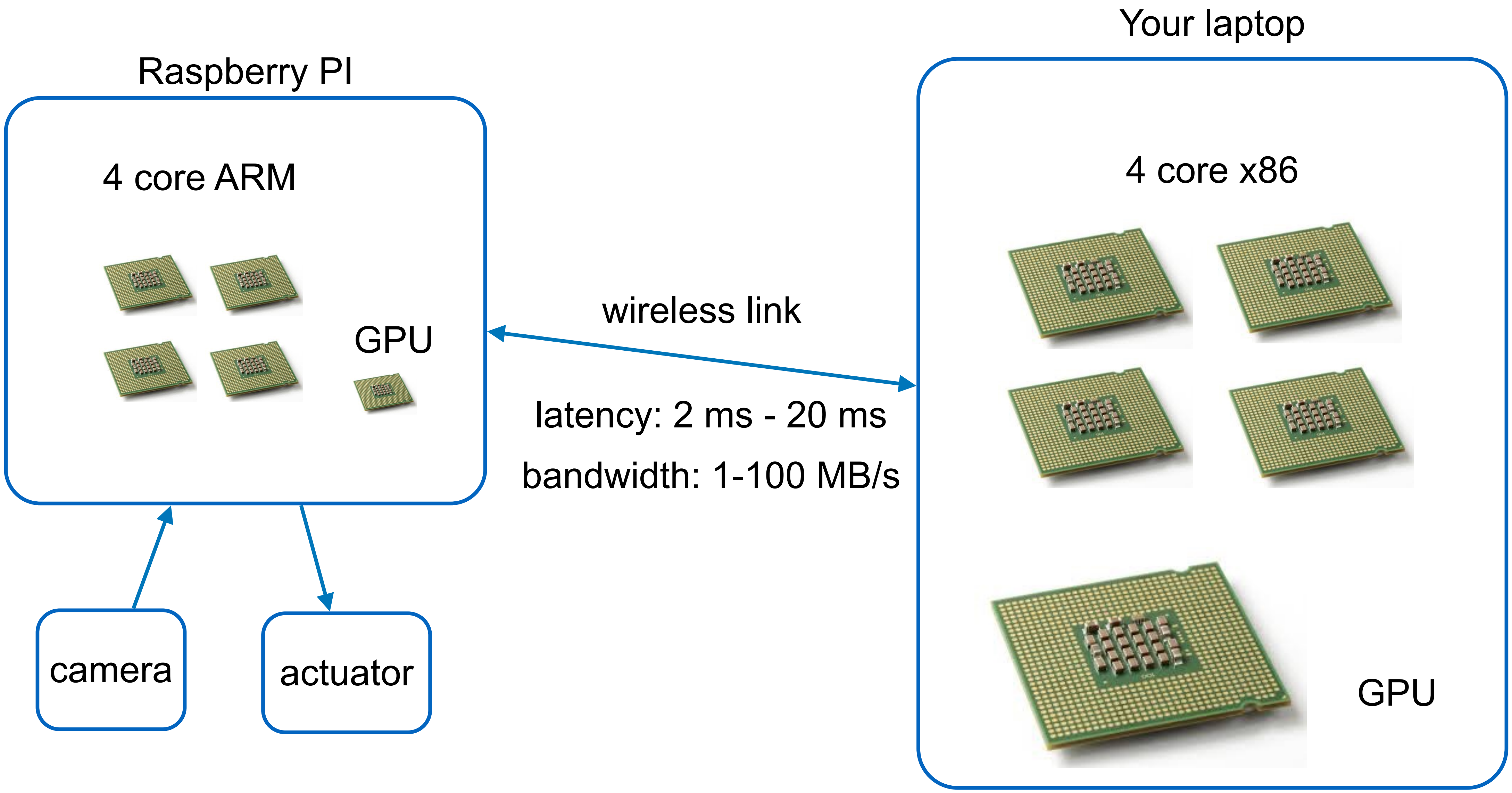
Example in Duckietown: Computation graph

- Computation graph: we collapse all computation in one node

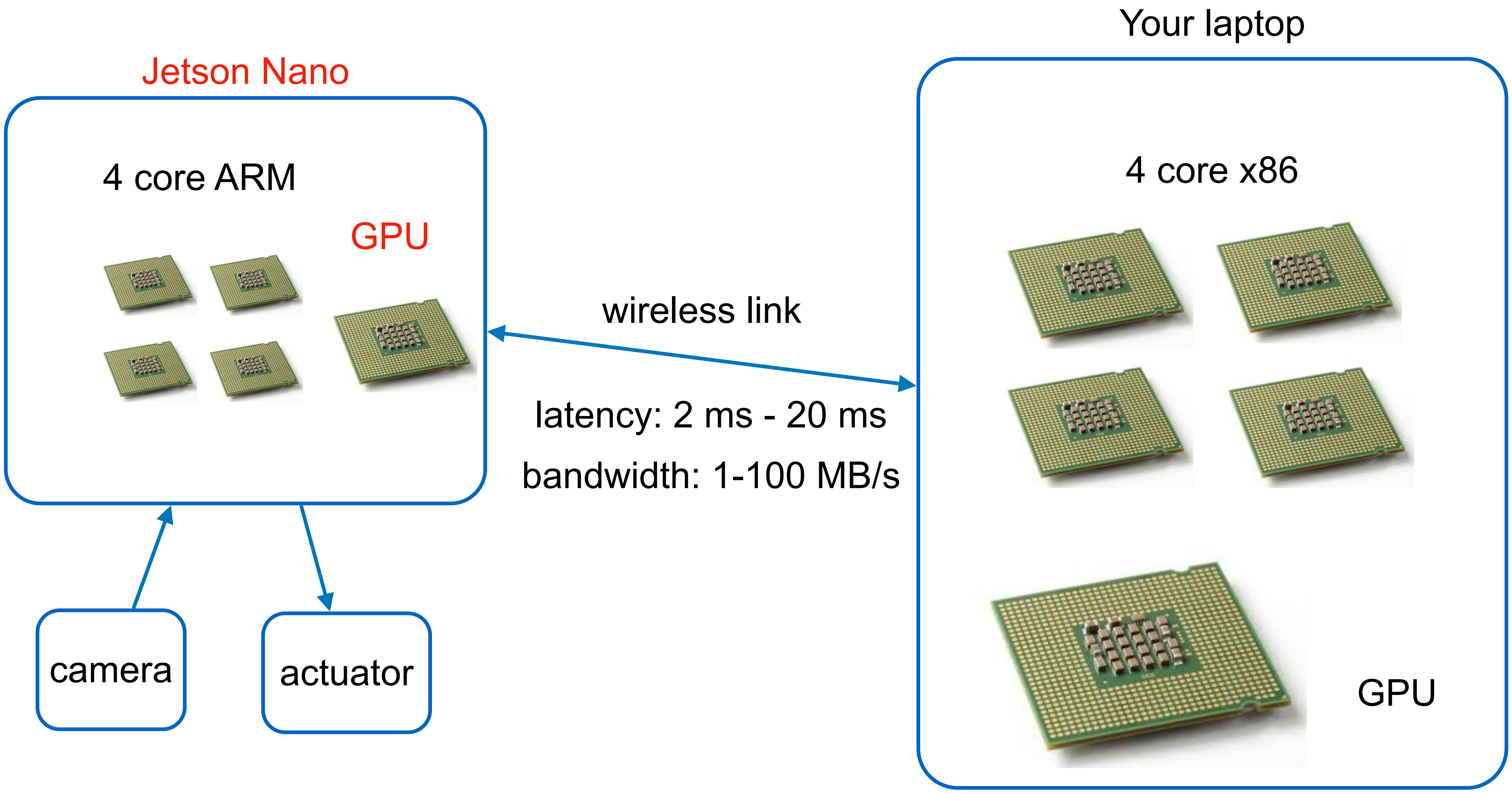


(abstracting all computation as one node)

Example in Duckietown: Resource graph



Example in Duckietown: Resource graph



Jetson Nano

4 core ARM

GPU

camera

actuator

wireless link

latency: 2 ms - 20 ms

bandwidth: 1-100 MB/s

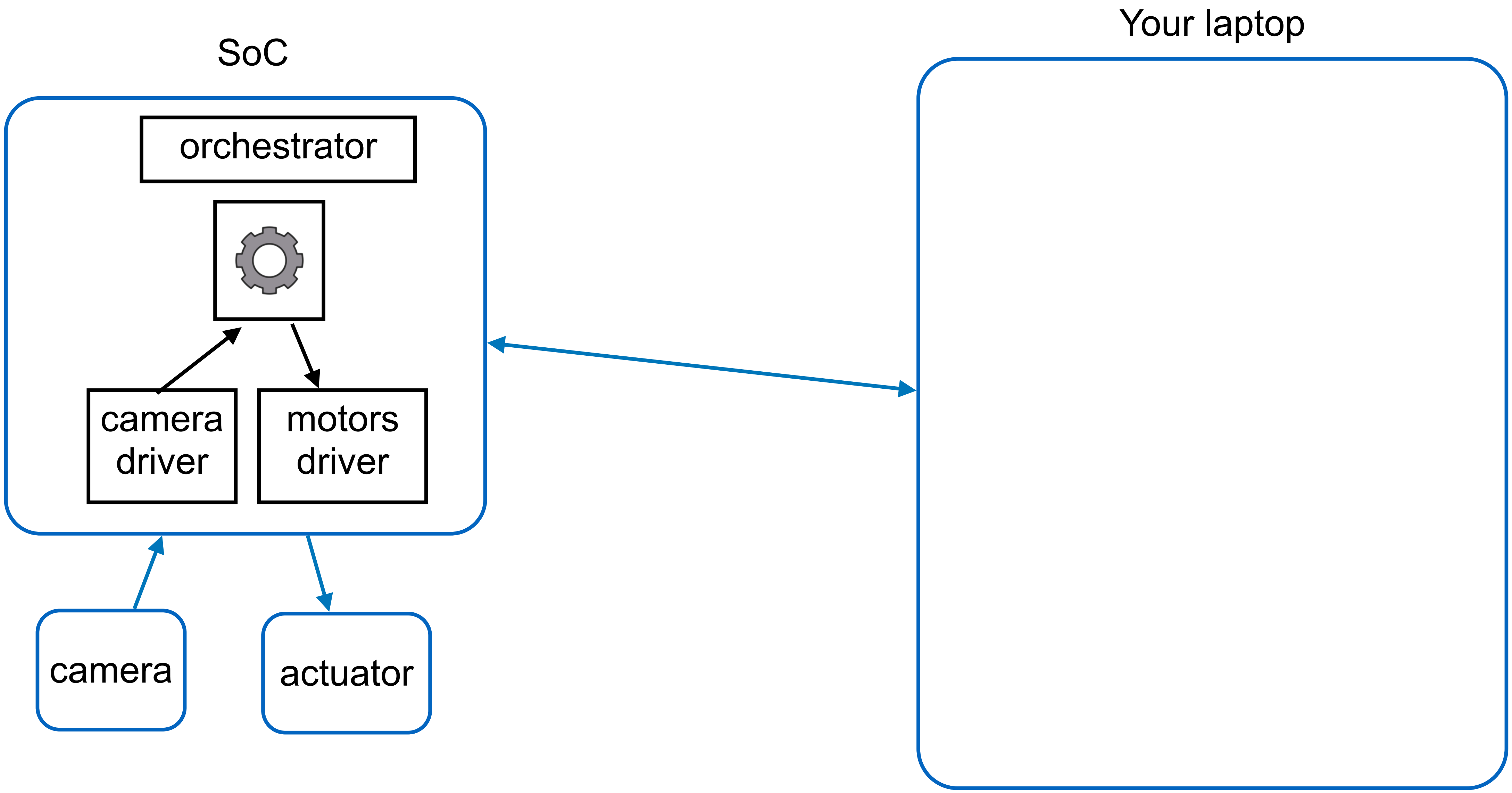
Your laptop

4 core x86

GPU

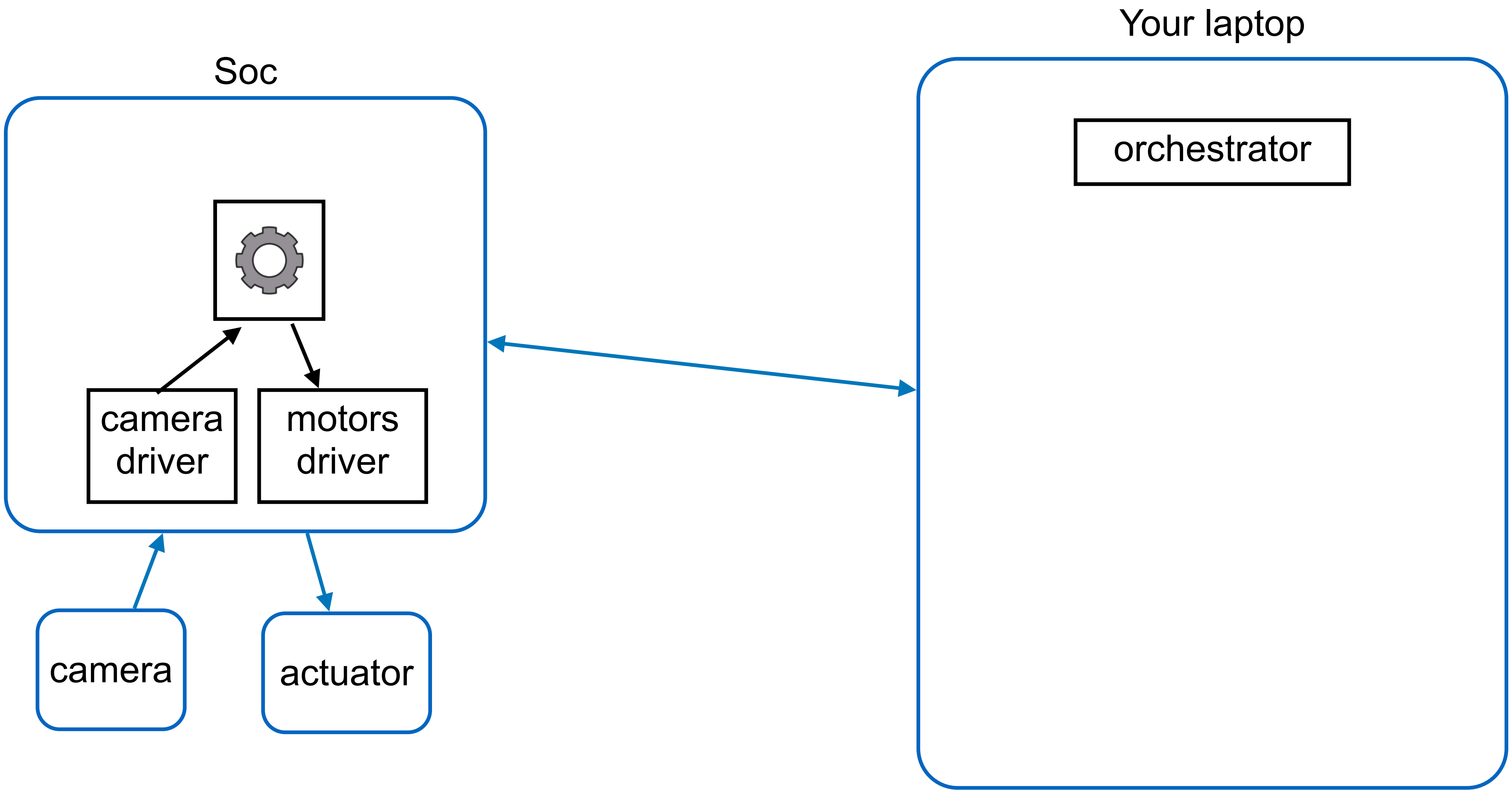
Example in Duckietown: Deployment 1

- Option 1: Run everything on the SoC (PI/Nano)



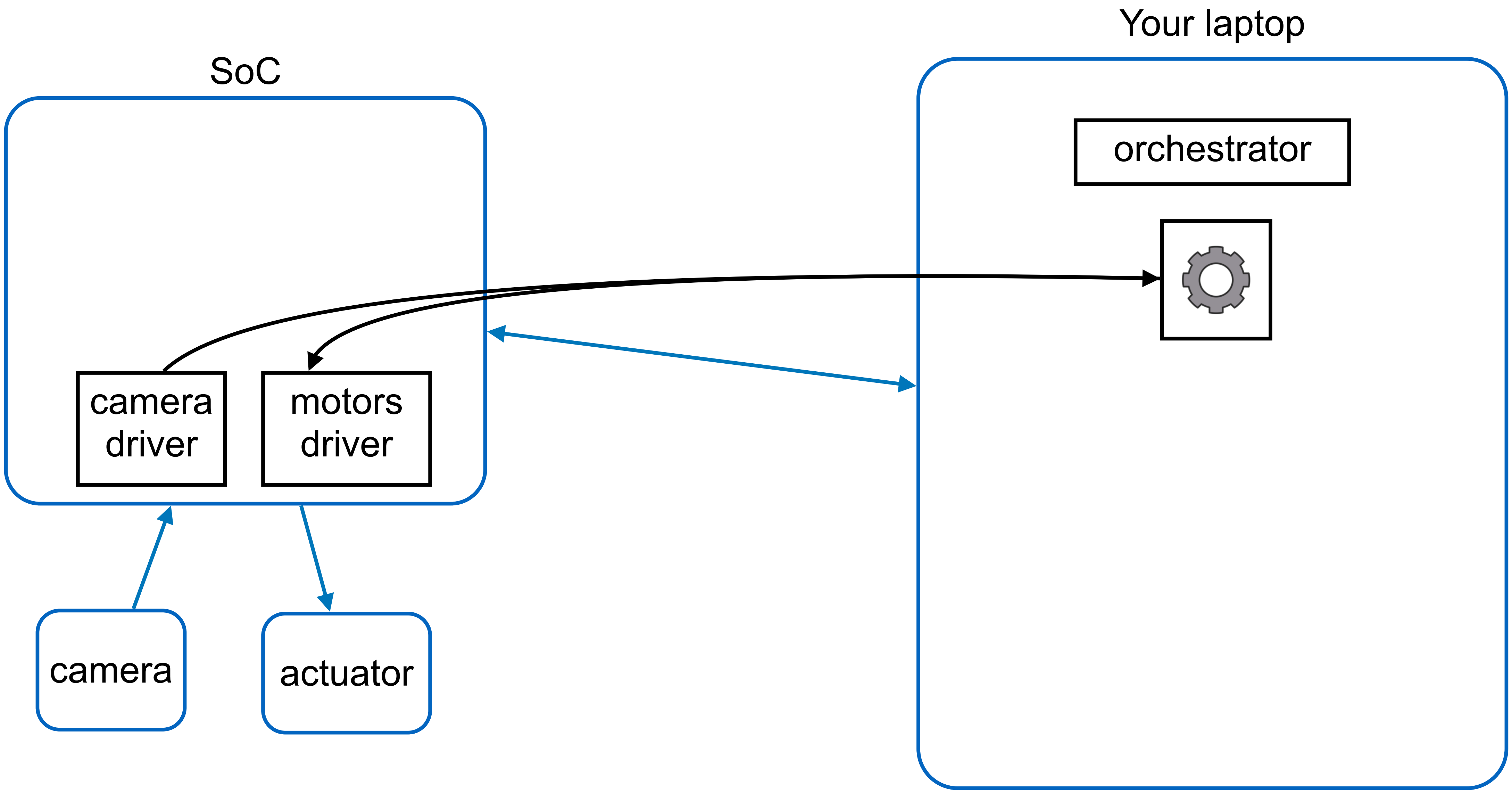
Example in Duckietown: Deployment 2

- Option 2: Run from your laptop



Example in Duckietown: Deployment 3

- Option 3: Run heavy processing on the laptop



An interesting read

- E. A. Lee and S. A. Seshia, *Embedded Systems -- A Cyber-Physical Systems Approach*