# The F1/10 simulator

Real-Time Embedded System - The F1tenth autonomous racing

# Course outline

› Intro course + basics of AD

› Hardware platform

› ROS2: Installation and profiling
  – Ex: ROS2 to HiL, open a bag

› Navigation: FTG, FTW, Pure pursuit
  – EX: navigation HiL

› Perception: scan matching, PF, LIO?
  – Ex: perception (PF with PThreads)

› Build the car

**I do not cover all aspects of AD!!!**

› Systems and control theory => Prof. Falcone

› Platforms and algorithms for autonomous systems => Prof. Sanudo & Prof. Falcone

› High-Performance Computing => Prof. Marongiu (FIM)

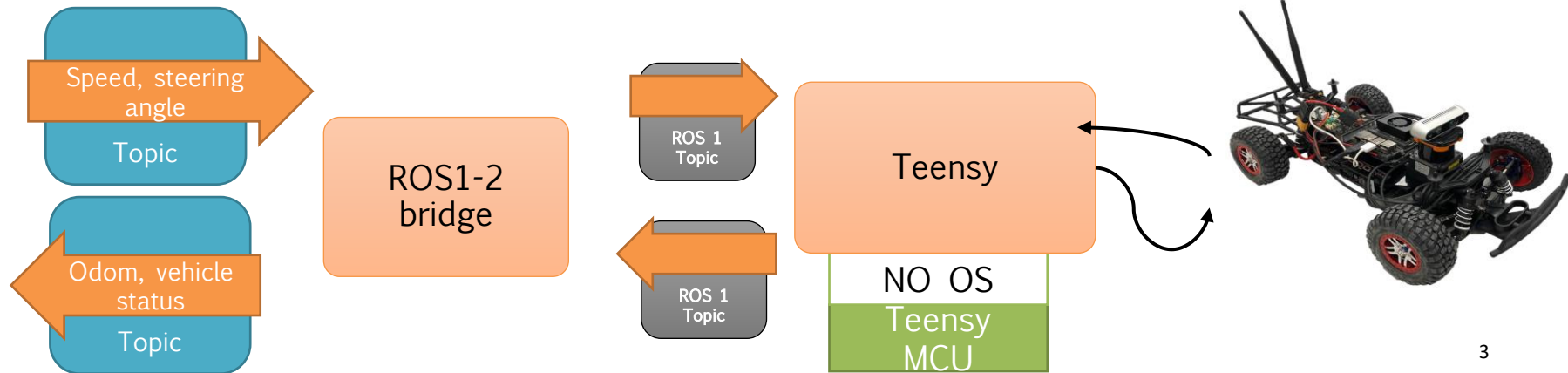› Machine Learning => Cucchiara's

# «Teensy» ROS node

Teensy is the microcontroller that controls the brushless engine

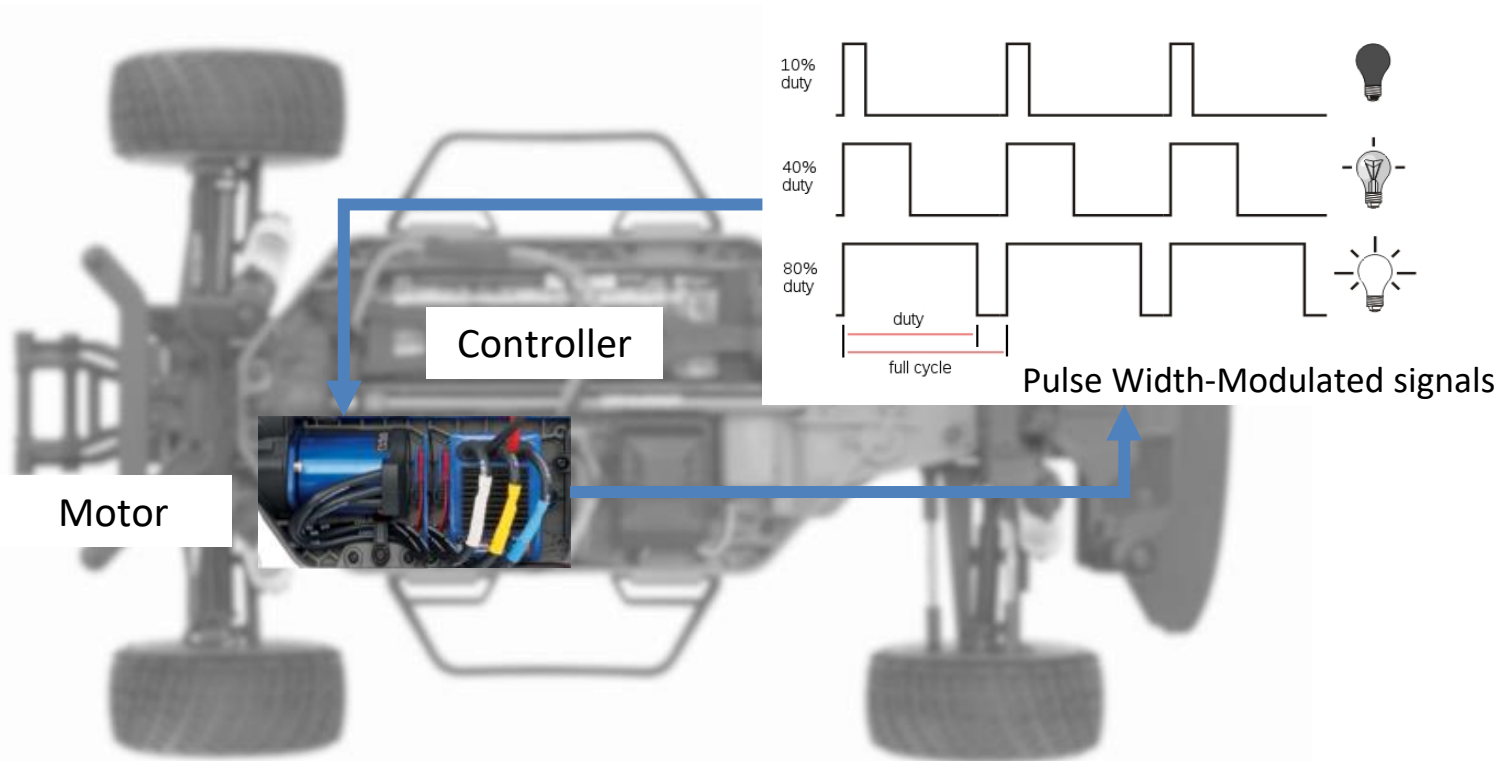› Typical scenario, also real cars have legacy actuator ECUs!

Note written in ROS1 (teensy has no OS!)

› ROS1-to-ROS2 bridge

# Actuation circuit

10% duty

40% duty

80% duty

duty

full cycle

Controller

Motor

Pulse Width-Modulated signals

Reference Signal:
"60 RPM"

# PWM - D/A conversion (recap..?)

*Generate a tension corresponding to a digital value stored in a register*

› Not easy! Use Pulse-Width Modulation (PWM)
  – (Almost) fully implementable in SW!

How it works

$$DC = \tau / T$$

1. Generate a periodic signal of amplitude 1 whose **duty cycle** is proportional to the digital value we want to convert

2. Give it to a low-pass filter, to average (such as Resistor-Capacitor - RC circuit)

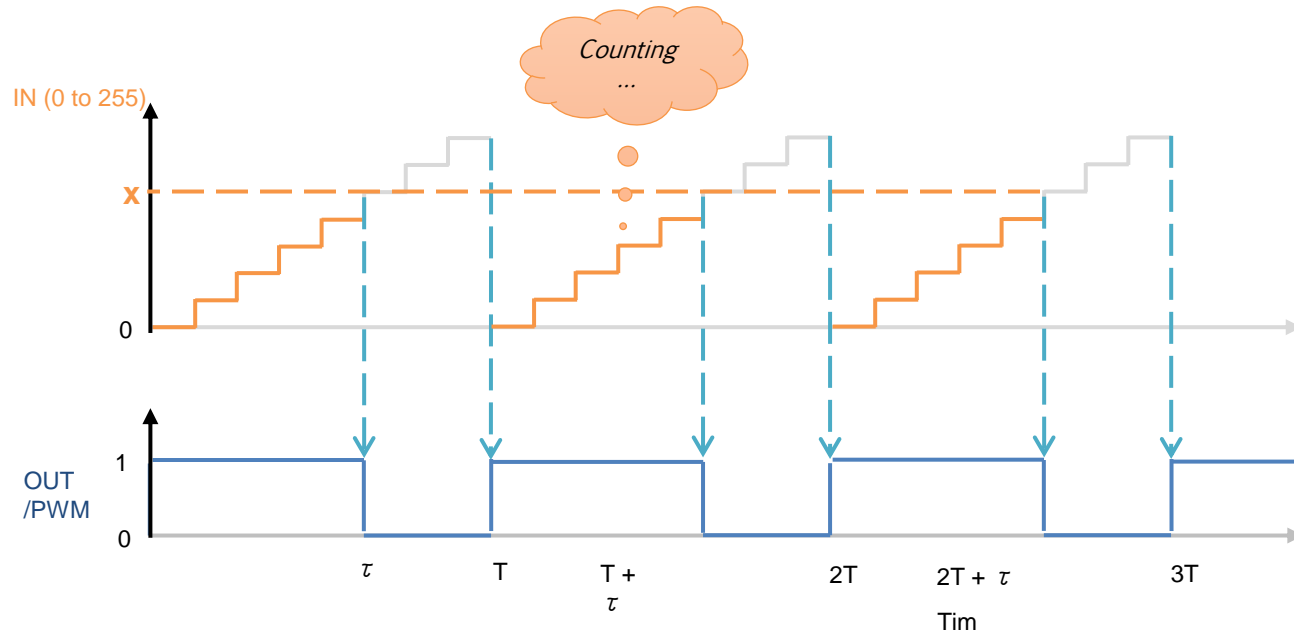3. ..and enjoy your analog signal! ☺

$\tau$ is 60% of T => DC = 0.6

Period T

1

0

$\tau$    T    T + $\tau$    2T    2T + $\tau$    3T

Tim
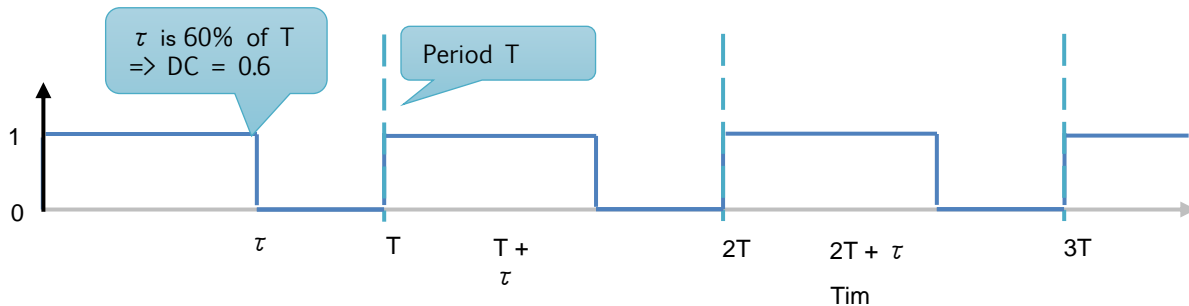
# PWM (recap..?) – step 1

We need

› A register that counts from 0 to 255 (8-bit)

› An output port (bit) set to '1' and becoming '0' when input value matches the one of the register
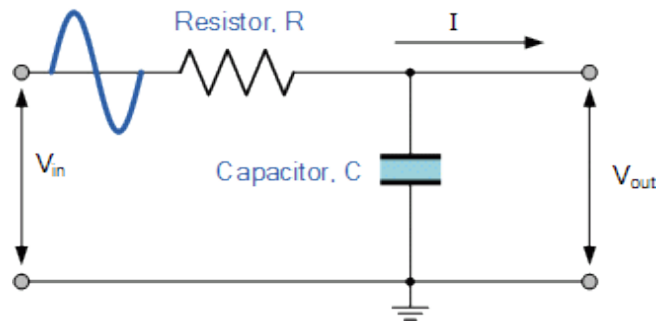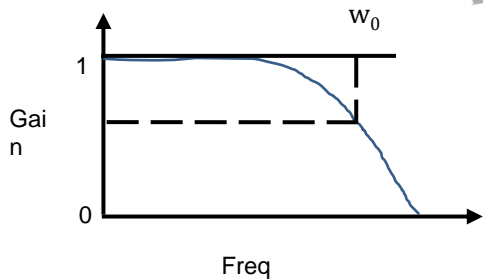
# RC low-pass filter (recap..?)

Simple electric circuit with a Capacitor and a Resistance

› "Averages" the IN value

$$|V_{out}| = |V_{in}| \times \frac{1}{\sqrt{1 + w^2 R^2 C^2}}$$

Resistor, R

I

$V_{in}$

Capacitor, C

$V_{out}$

Frequency domain

$w_0$
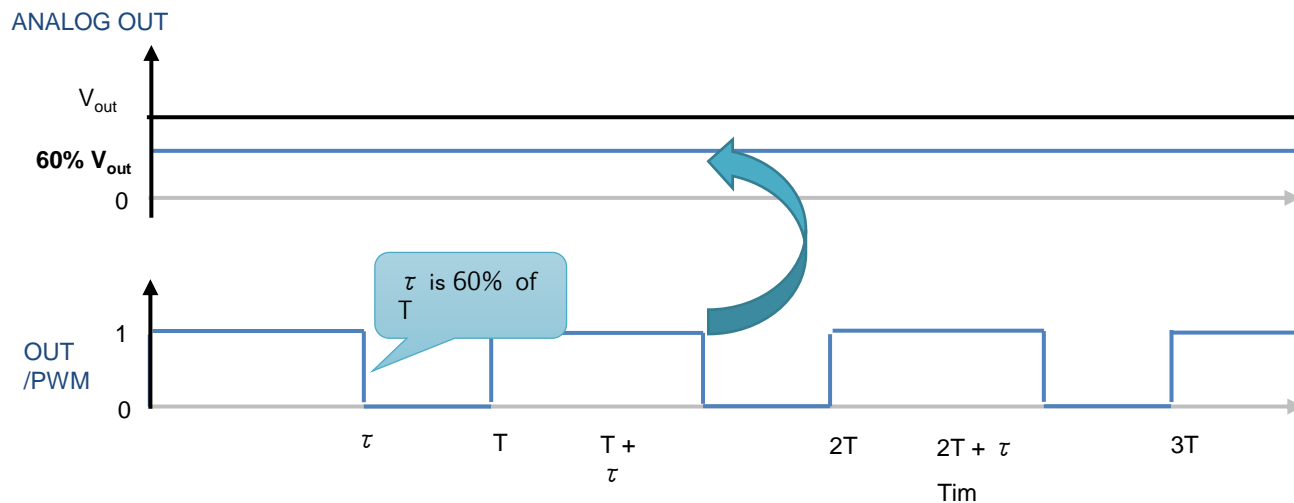
1

Gain

0

Freq

**Cutoff freq** $w_0 = \dfrac{1}{RC}$

# PWM (recap..?) – step 2

Now, compute the average for every T

› Using a low-pass filter

› Plug it to output port

› *Et voilà*

Extremely useful in engine controls

ANALOG OUT

$V_{out}$

**60% $V_{out}$**

0

τ is 60% of T

OUT /PWM

1

0

τ      T      T + τ      2T      2T + τ      3T

Tim

# Electronic Speed Controller – (V)ESC

Manages the brushless engines of the car
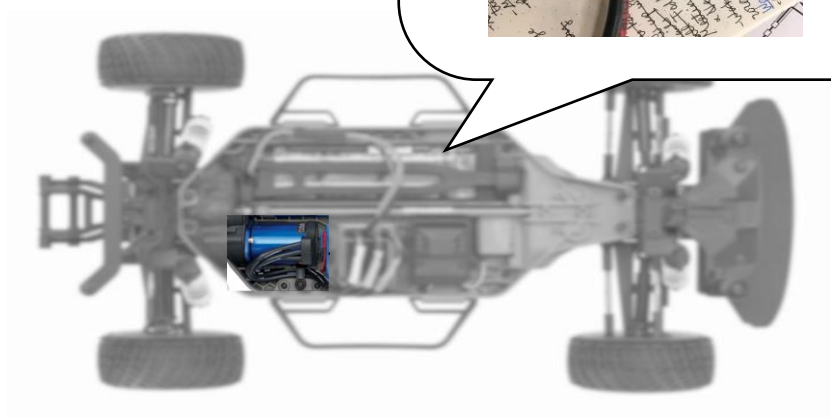
Programmed via a tool
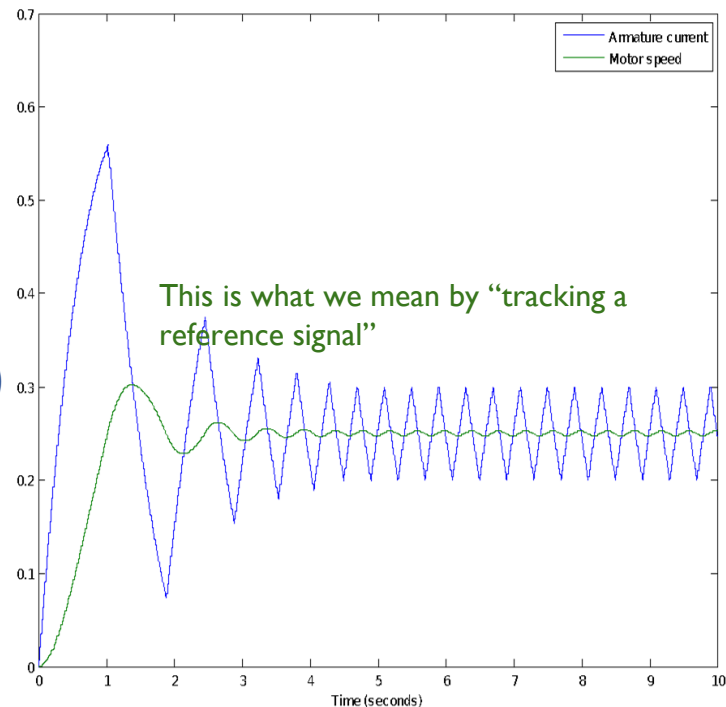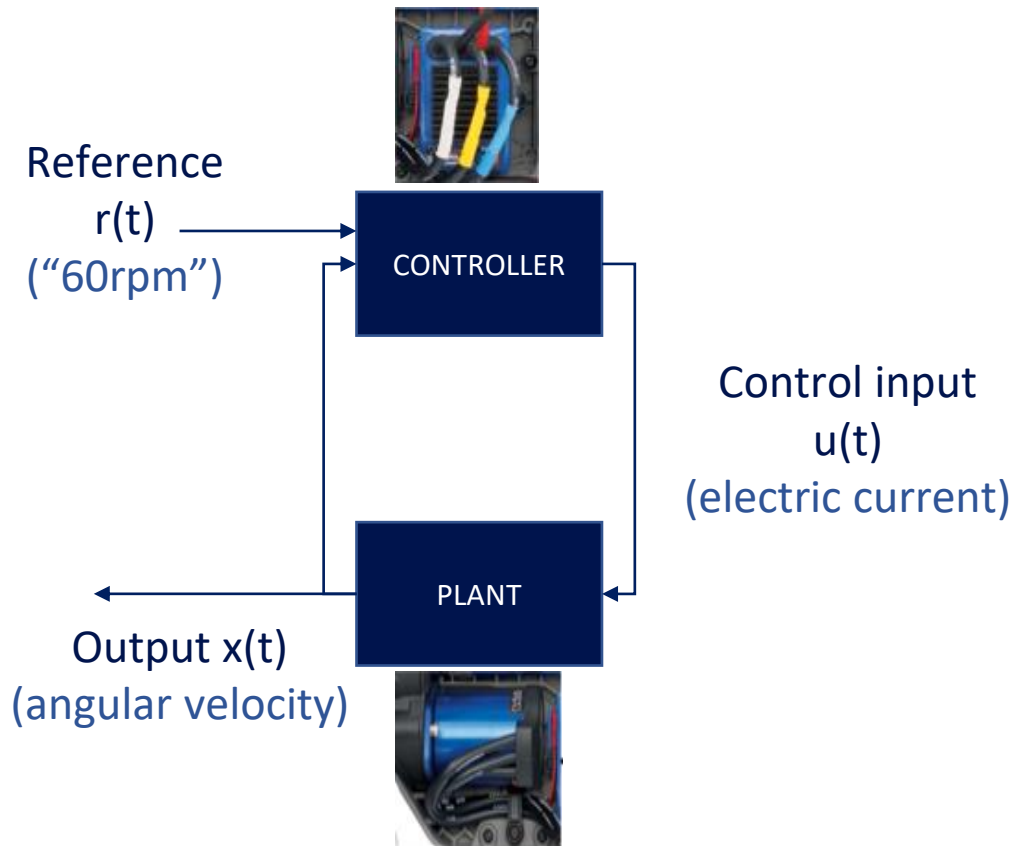
› Once-for-all

Accepts ROS1 commands for lat/lng control

› Acceleration & steering

https://vesc-project.com/

# Actuation system W/VESC



Reference
r(t)
("60rpm")

CONTROLLER

Control input
u(t)
(electric current)

PLANT

Output x(t)
(angular velocity)

This is what we mean by "tracking a reference signal"
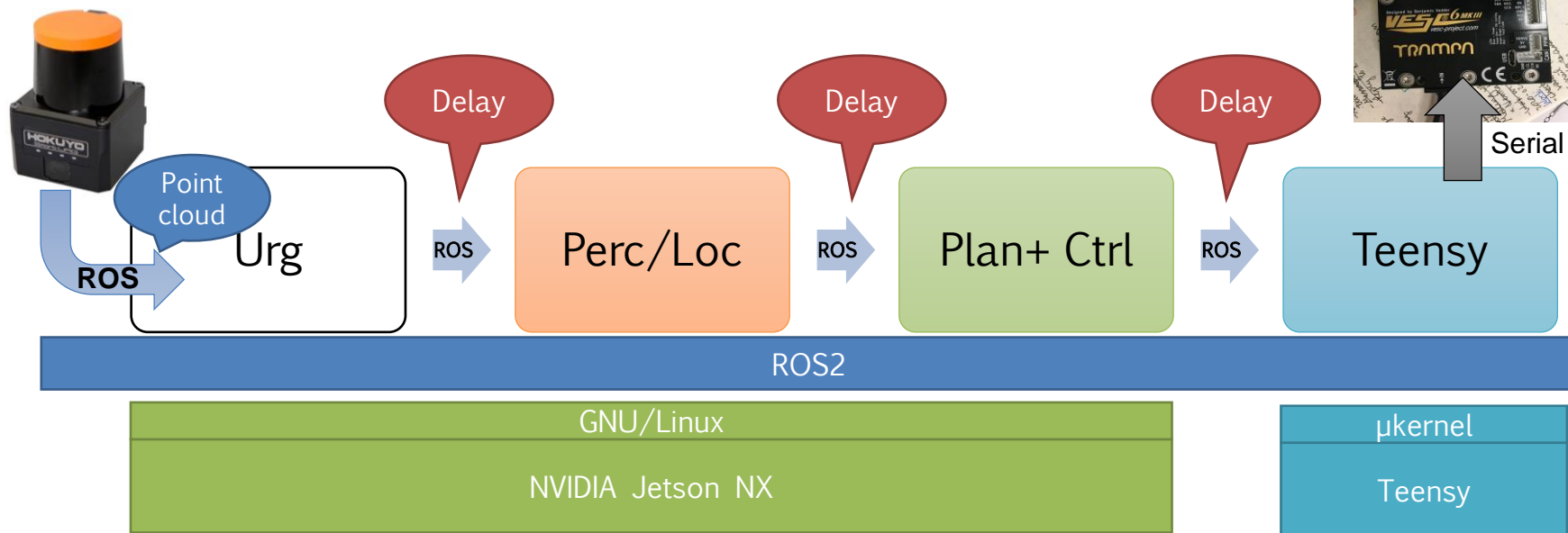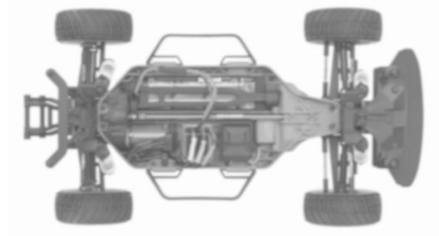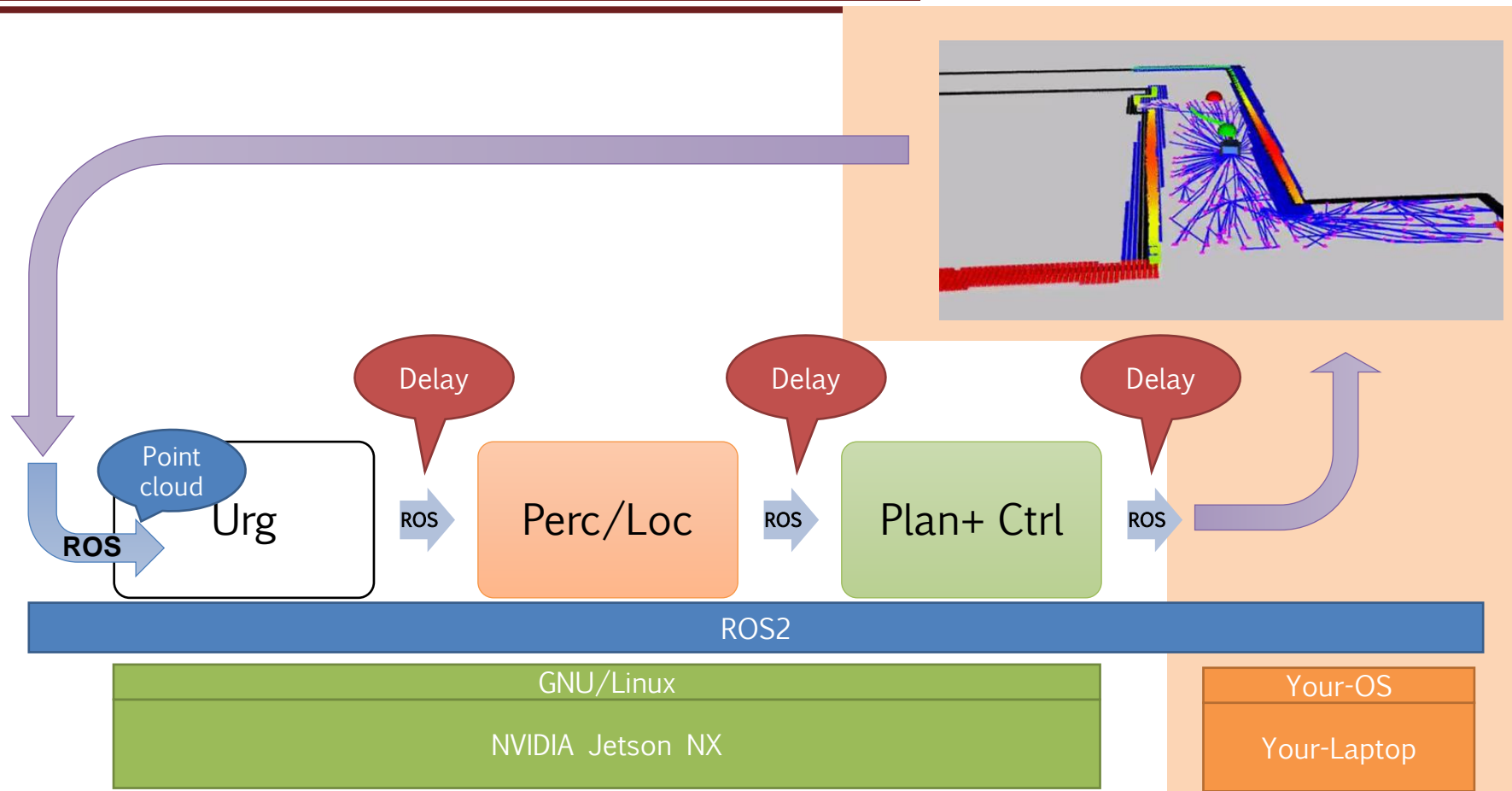
# Actuation timings

› LiDAR publishes @40Hz (25ms)

› Teensy loop @?? Hz

    – We don't care! If we don't send a signal, it holds the previous one

› Our pipeline must meet 25ms deadline...incl. ROS delays!



Delay      Delay      Delay

Point cloud

ROS

Urg    ROS    Perc/Loc    ROS    Plan+ Ctrl    ROS    Teensy

Serial

ROS2

GNU/Linux

μkernel

NVIDIA  Jetson  NX

Teensy

# Actuation in simulator (HiL)



**Delay**     **Delay**     **Delay**

**Point cloud**

**ROS**    Urg    **ROS**    Perc/Loc    **ROS**    Plan+ Ctrl    **ROS**
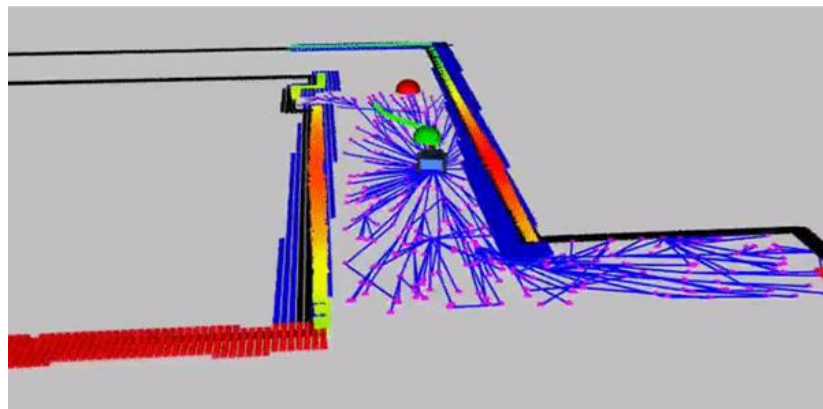
ROS2

GNU/Linux

NVIDIA Jetson NX

Your-OS

Your-Laptop

# F1/10 simulator

## F1TENTH Gym:

- Lightweight 2D simulator built in Python

- Asynchronous

- Faster than real-time execution (30x realtime)

- Realistic vehicle simulation and collision

- Runs multiple vehicle instances

- Publishes laser scan and odometry data
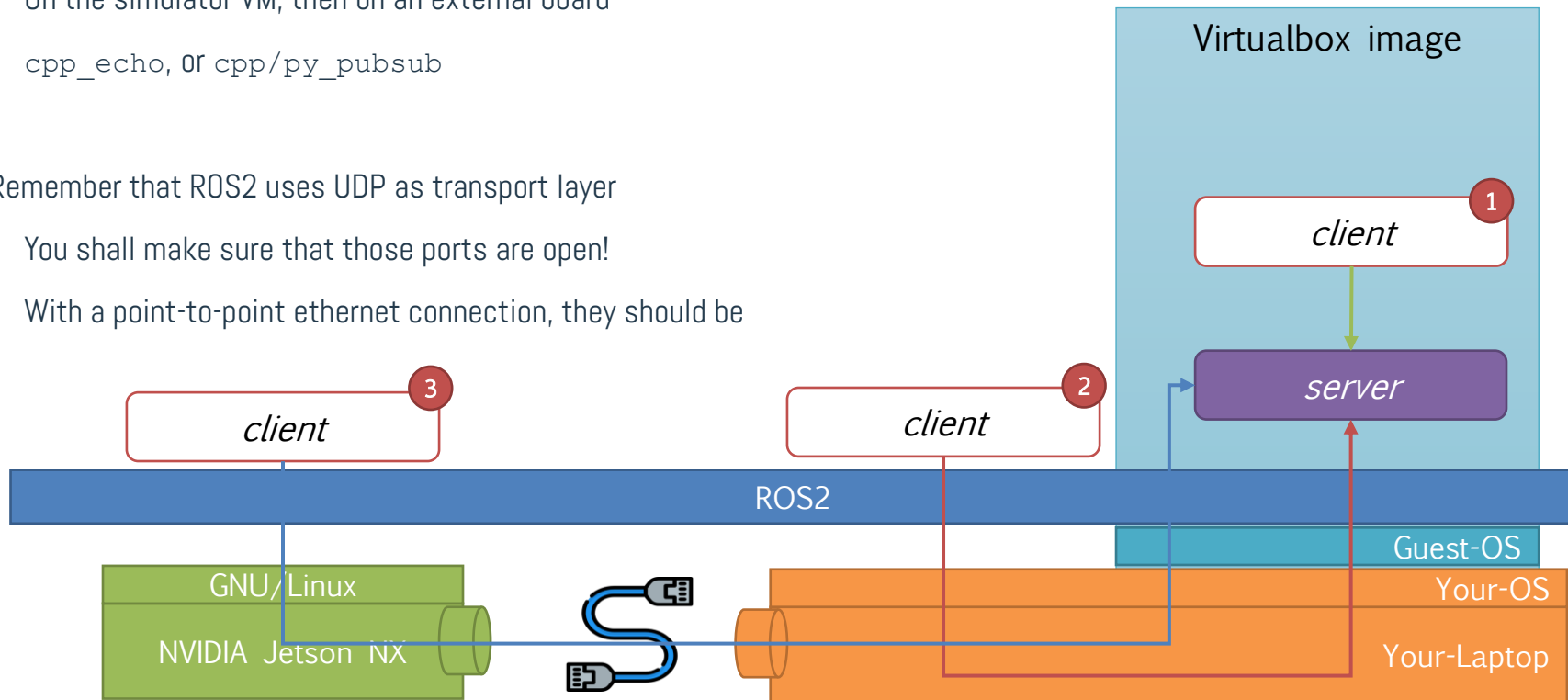
- Built for fast prototyping

# Exercise

Let's code!

Build and run the pub/sub examples that come with in ROS2

› On the simulator VM, then on an external board

› `cpp_echo`, or `cpp/py_pubsub`

Remember that ROS2 uses UDP as transport layer

› You shall make sure that those ports are open!

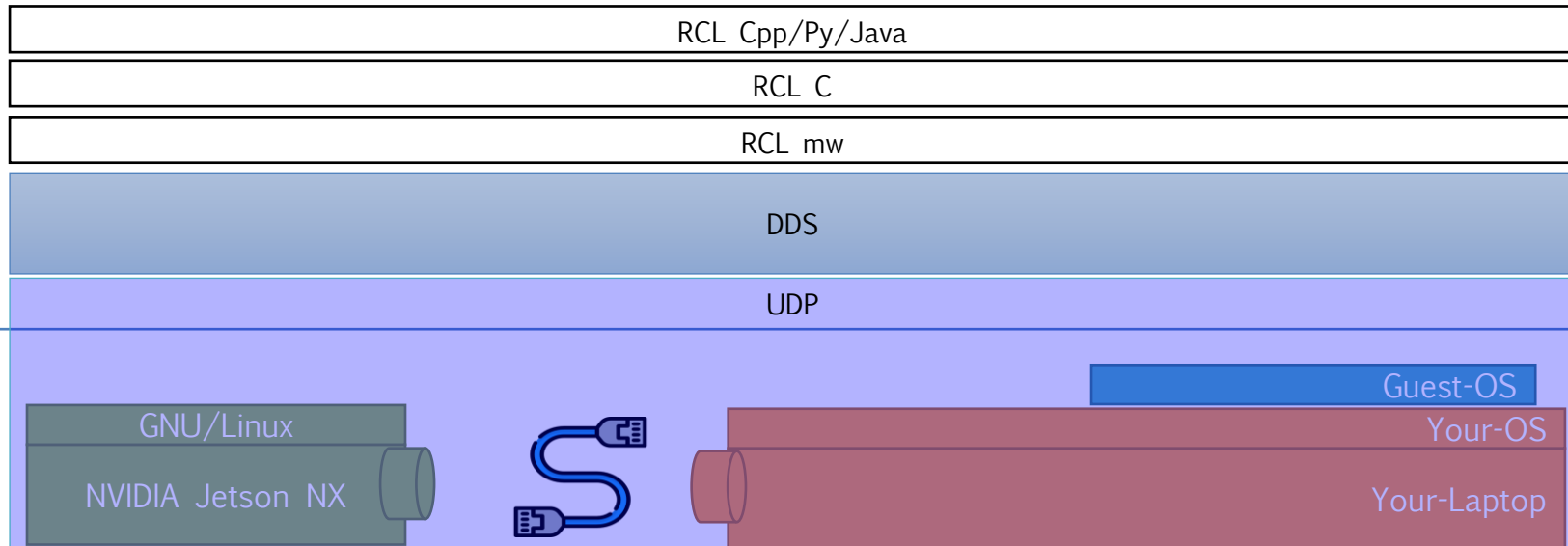› With a point-to-point ethernet connection, they should be

# ROS_DOMAIN_ID and ROS2 ports

› ROS_DOMAIN_ID groups communication onto sub-nets [0-101] – Default 0

› Each domain has an associated port for discovery and (2x) communication

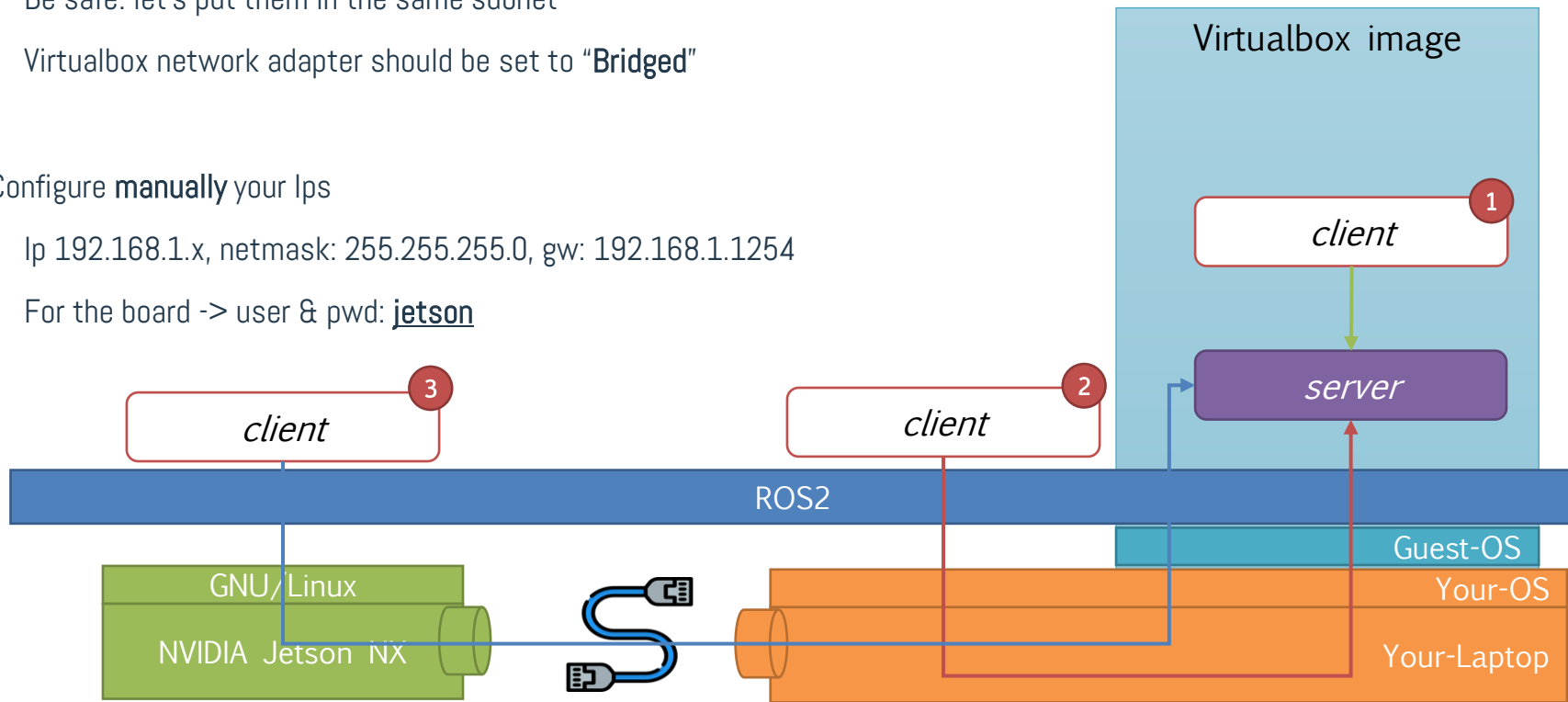› Example: Domain ID 1 uses port 7650 and 7651 for multicast, then 7660/1 on for comms

# Networking

The three machines should be able to "speak" to each other

› Be safe: let's put them in the same subnet

› Virtualbox network adapter should be set to "**Bridged**"

Configure **manually** your Ips

› Ip 192.168.1.x, netmask: 255.255.255.0, gw: 192.168.1.1254

› For the board -> user & pwd: <u>jetson</u>

# Network map



Let's code!

We already set up reserved addresses for the boards
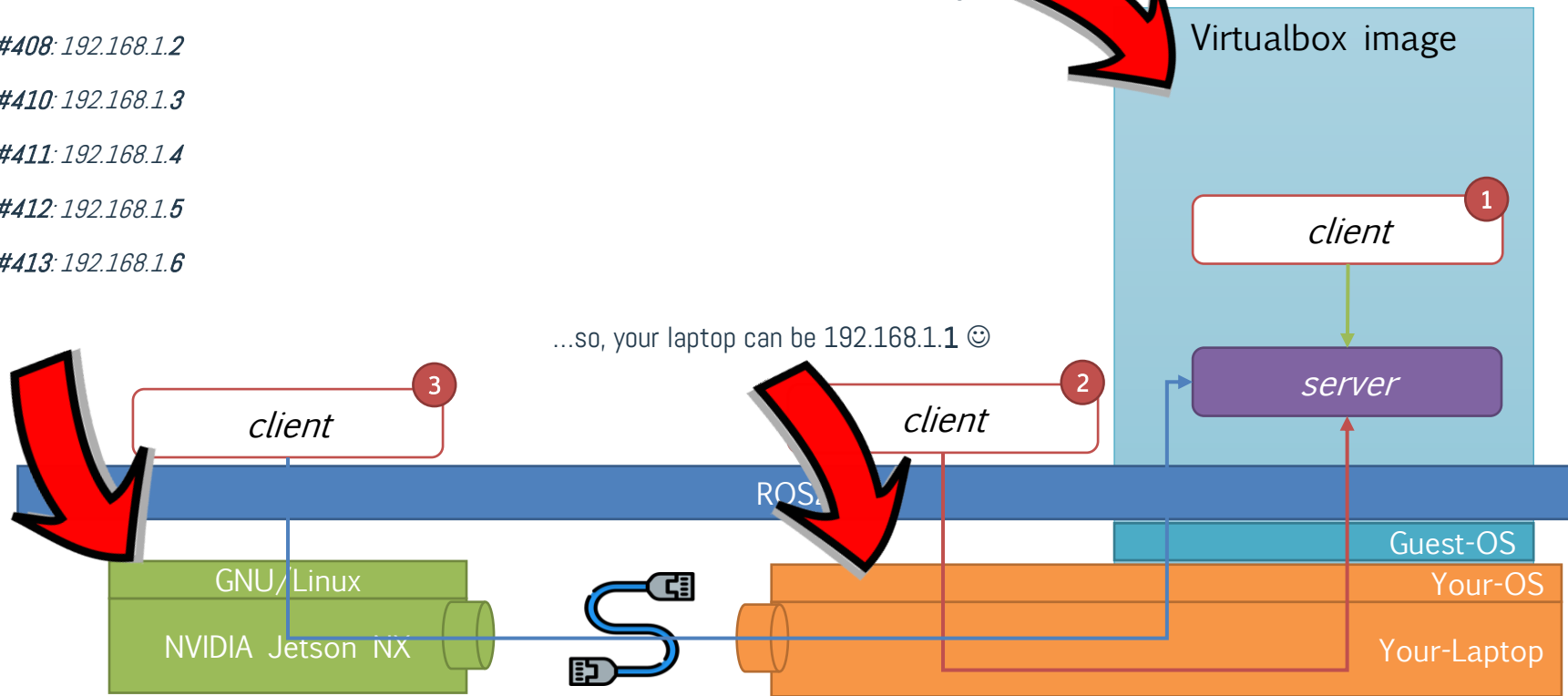
*#408*: *192.168.1.2*

*#410*: *192.168.1.3*

*#411*: *192.168.1.4*

*#412*: *192.168.1.5*

*#413*: *192.168.1.6*

Also, remember to set the address of your VM guest OS (192.168.1.**something**)

Virtualbox image

...so, your laptop can be 192.168.1.**1** ☺

*client* 1

*server*

*client* 3

*client* 2

ROS.

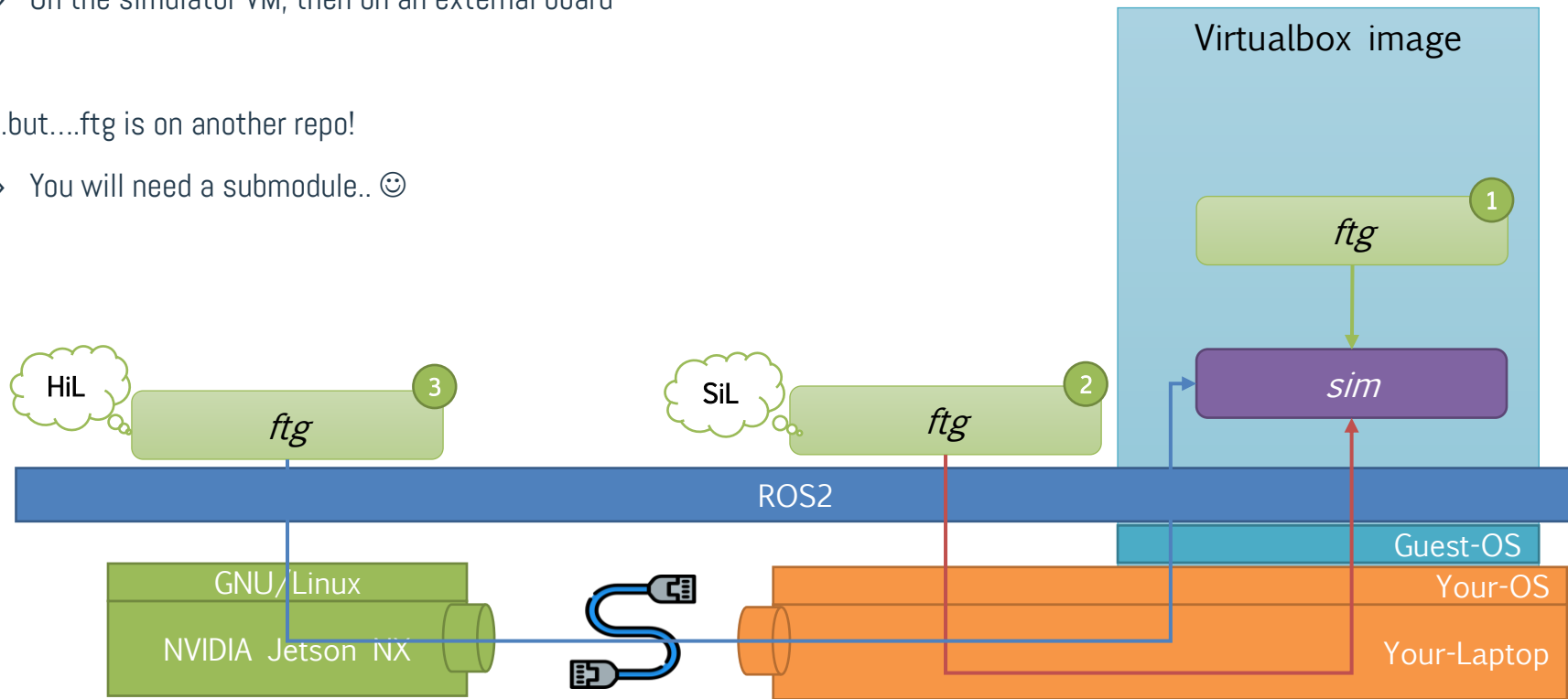Guest-OS

Your-OS

Your-Laptop

GNU/Linux

NVIDIA Jetson NX

# Exercise



Build and run the Follow-the-Gap *ftg* node and the F1tenth-Gym simulator

› On the simulator VM, then on an external board

..but....ftg is on another repo!

› You will need a submodule.. ☺

# References

## Course website

› http://personale.unimore.it/rubrica/contenutiad/markober/2023/71846/N0/N0/10005

› https://github.com/HiPeRT/F1tenth-RTES
  – Online resources/preview

## My contacts

› paolo.burgio@unimore.it

› http://hipert.mat.unimore.it/people/paolob/

## Resources

› https://github.com/HiPeRT/F1tenth-RTES/blob/master/Code/ros2/LAB_CHEAT_SHEET.md

› A "small blog"
  – http://www.google.com