Seamless TinyML lifecycle management

In Software Engineering Project with Helsinki University CS

16/01/2023

Basic info (1/2)

- Here's the original announcement
- Here's our proposal, Seamless TinyML lifecycle management
- 5 students are assigned to our project.
- 15 working hours / week / student is expected.
- The project duration, 14 weeks (week 3-16), is scheduled.
  - 15 hours * 14 weeks * 5 students
  - = 1050 hours / 7.5 hour
  - = 140 man days / 22
  - = 6.4 man month
  - Which SBI workload unit to use? (e.g. 3 hours, 7.5 hours or 15 hours)

Basic info (2/2)

Milestone

1. MWC is at the end of FEB. on **week 9**.
2. NEXUS Demo day is on 27th March on **week 13**. ## Project duration:
3. Originally, 14 weeks, Week 3-16
4. Preferably, 12 weeks, Week 3-14
5. Ideally, 10 weeks, Week 3-12 ## Weekly workforce

- 15 hours * 14 weeks = 210 hours

1. Originally, 210 hours / 14 weeks = 15 hours/week
2. Preferably, 210 hours / 12 weeks = 17.5 hours/week
3. Ideally, 210 hours / 10 weeks = 21 hours/week

Can UI (Dashboard & Control panel) parts be prioritized to meet some milestones?

Project goal (1/2)

We will reproduce Roberto's demo video, adding its ML training phase, along with TinyML MCU. While this demo uses relatively large hardware which may not belong to TinyML strictly (e.g. TinyML should run on RTOS but not on Linux), we are gradully migrating to TinyML MCUs. There are 4 benefits of starting with the original setting:

1. Jetson nano is a standalone GPU, where we run the following app locally at once before starting pipelining on other nodes.
   - Computer vision app, inc. ML models
   - dashboard on a webserver
   - Jupyter notebook

Project goal (2/2)

2. We could learn from Jetson nano mature tool-stack what kind of tool-stack is still missing to implement TinyMLaaS.
3. We could start with this existing demo immediately with runnable CI, and
4. We are polishing it more fancy gradually towards TinyML as-a-Service.
5. We could gradually migrating to TinyML by adding or replacing a node one by one.
   - For example, we could replace the data acquistion node with:
     a. Camera sensor + Arduino Nano 33 BLE Sense + RPI (for IP)
     b. Camera sensor + RPI pico with WiFi

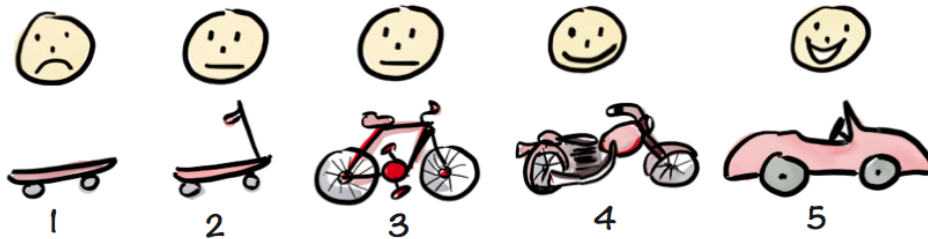Our final goal is to run ML on a mirocontrooler node but it's better to start with the safer configuration at first.

MVP journey

We should always have a runnable MVP automatically generated by CI/CD at every Sprint.

Not like this....

1   2   3   4

Like this!

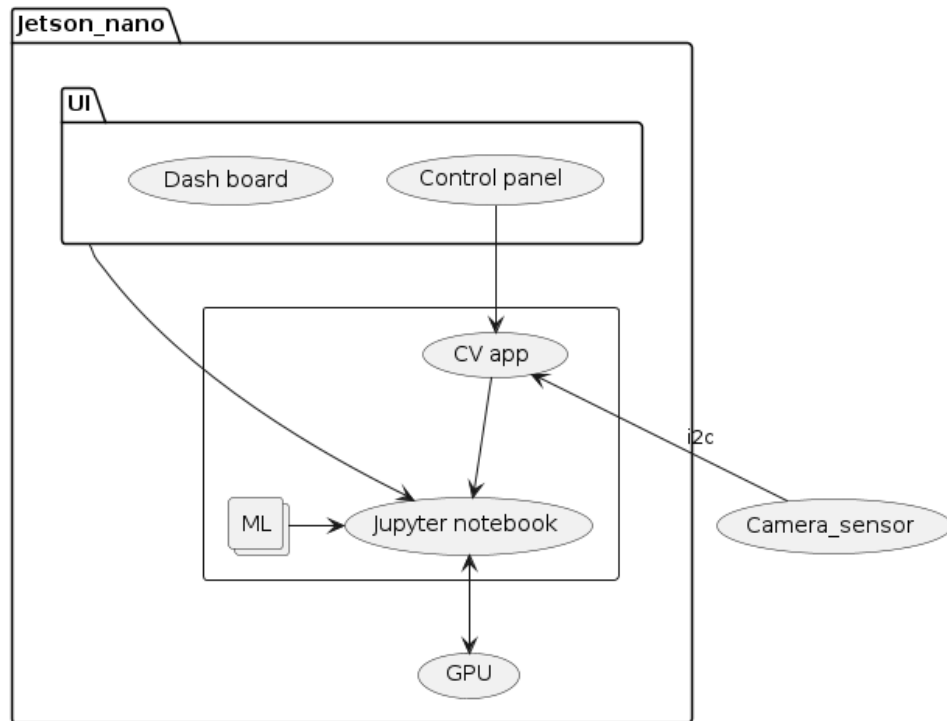1   2   3   4   5
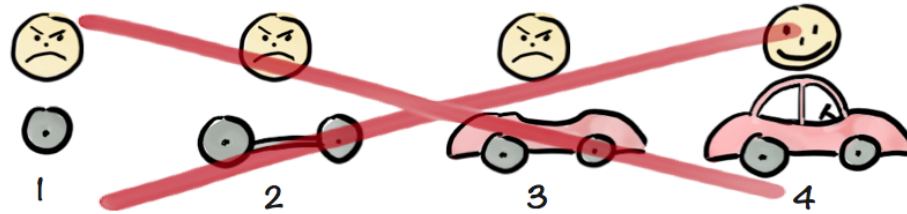
Henrik Kniberg

MVP0

Jetson nano is almost a laptop with GPU so that everything should work
standalone.

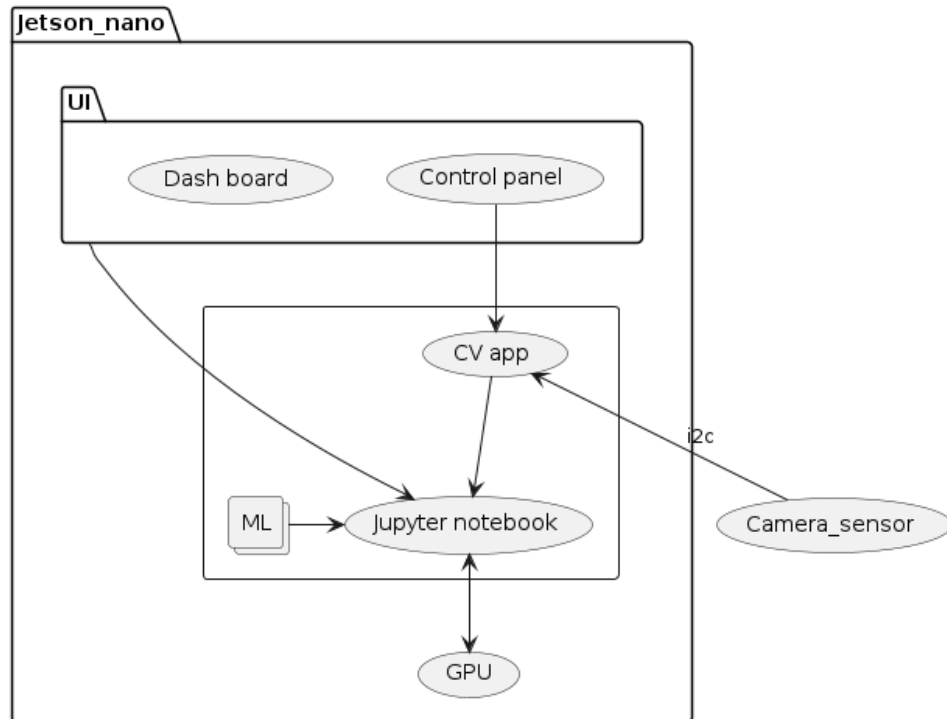### ML pipeline

1. Object detection
2. -> face detection
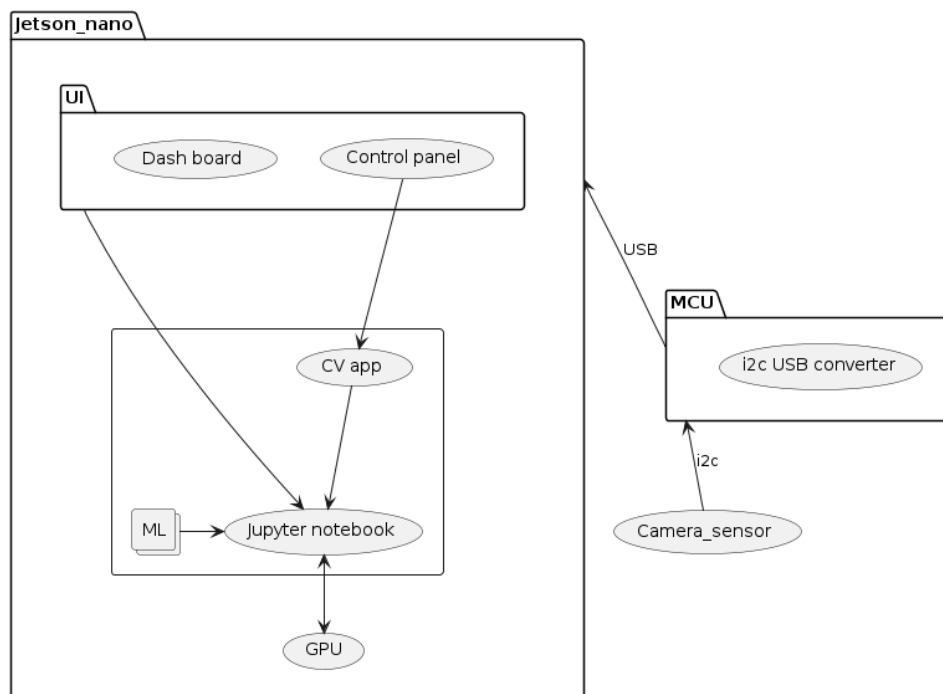3. -> Person identificaion pipeline

MVP0: Standalone ML processing

Jetson_nano

UI

Dash board   Control panel

CV app

i2c

ML → Jupyter notebook

Camera_sensor

GPU

4

MVP0 -> MVP1

MVP0: Standalone ML processing

MVP1: Add MCU via I2C to collect data

Jetson_nano

UI

Dash board   Control panel

USB

MCU

i2c USB converter

CV app

i2c

ML → Jupyter notebook

Camera_sensor

GPU
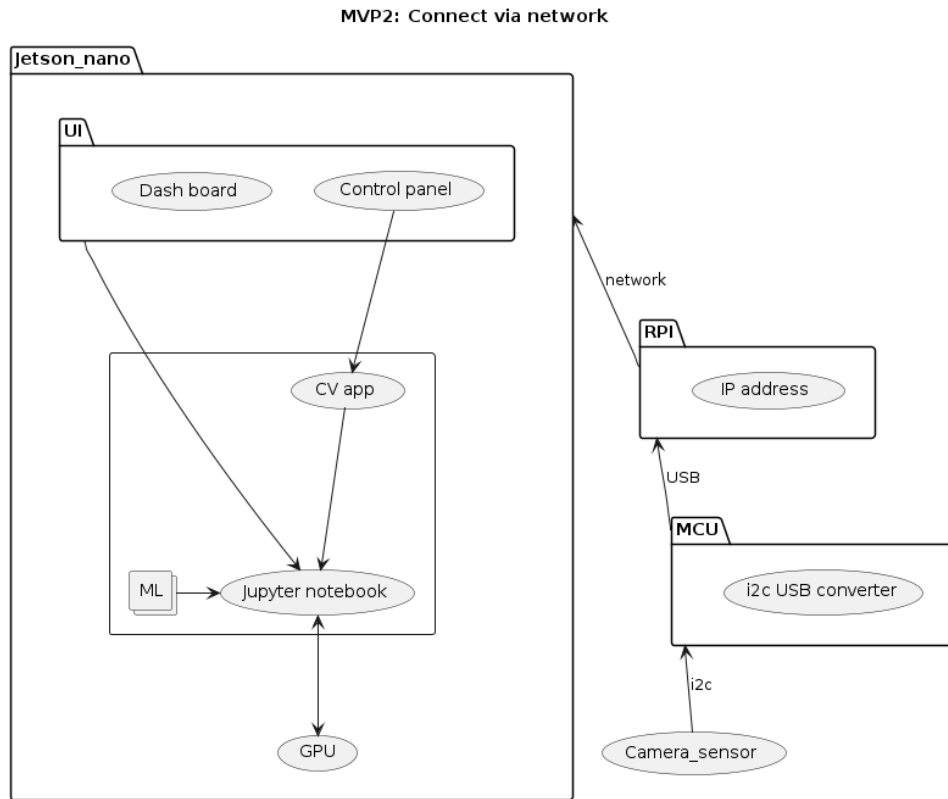
At first, we could push sensor part out via I2C.

MVP1 -> MVP2

MVP1: Add MCU via I2C to collect data

MVP2: Connect via network

Jetson_nano

UI

Dash board   Control panel

CV app

ML → Jupyter notebook

GPU

network

RPI

IP address

USB

MCU

i2c USB converter

i2c

Camera_sensor
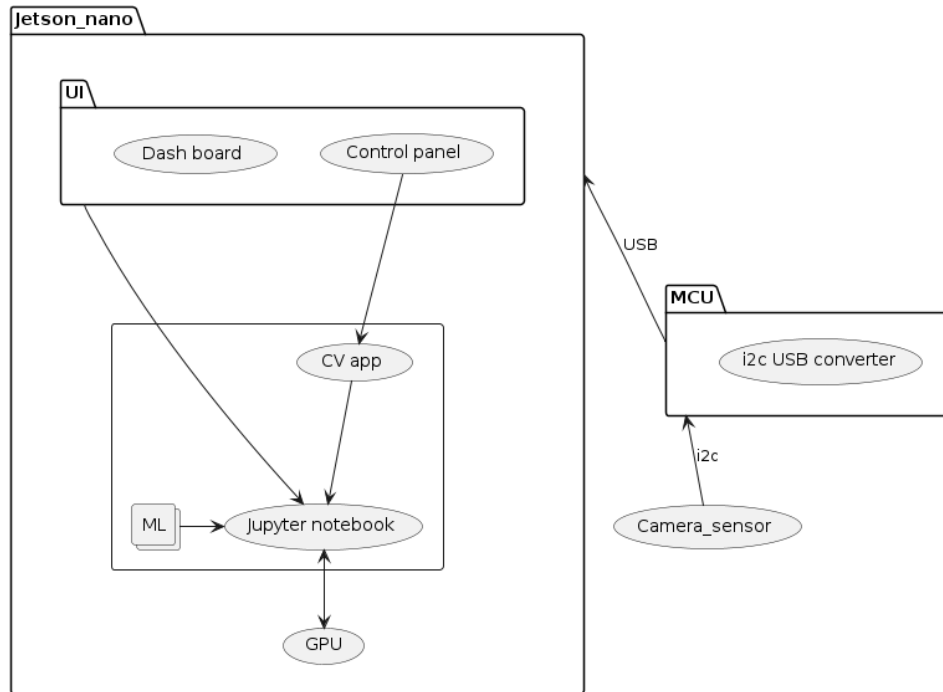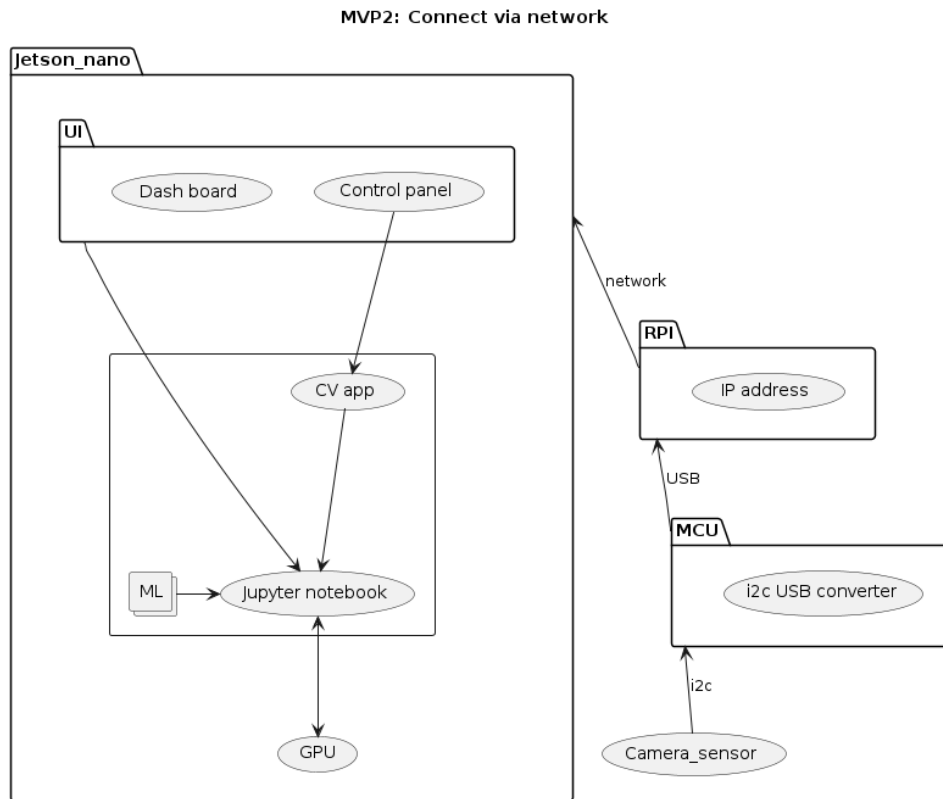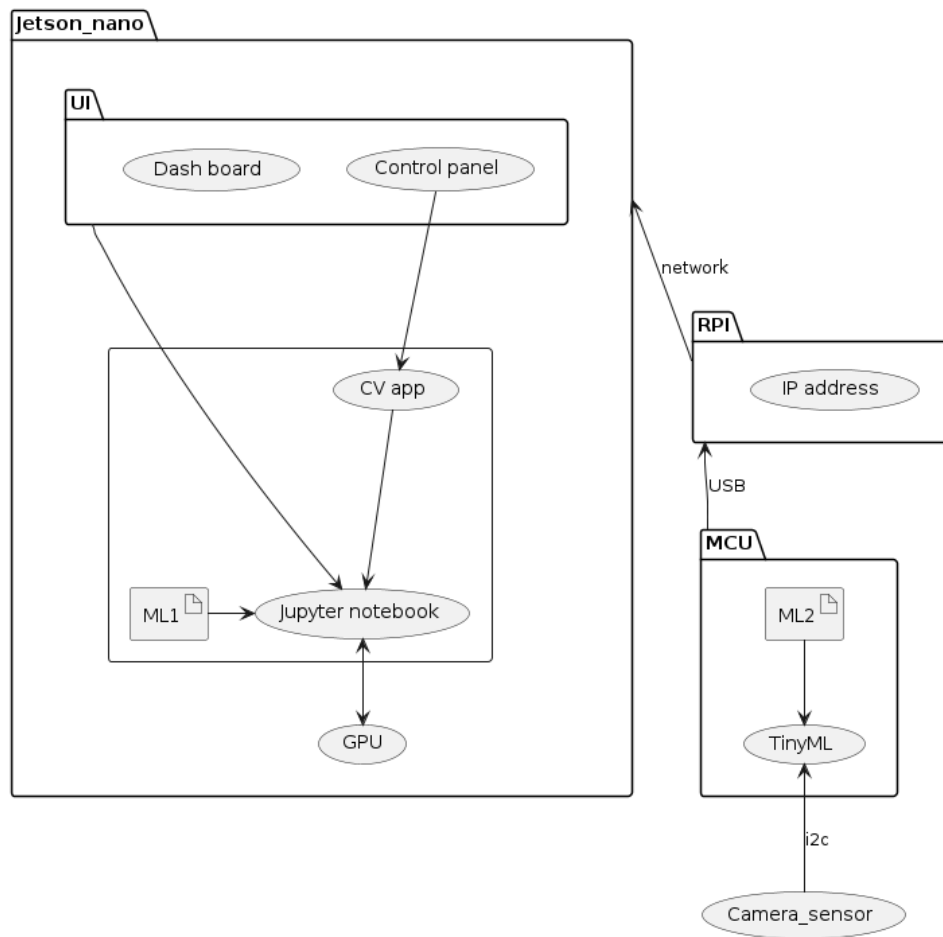
To orchestrate ML, IP connection is convenient so that we insert RPI between
Jetson and MCU.

MVP2 -> MVP3

MVP2: Connect via network

Jetson_nano

UI
- Dash board
- Control panel

CV app

ML → Jupyter notebook

GPU

network

RPI
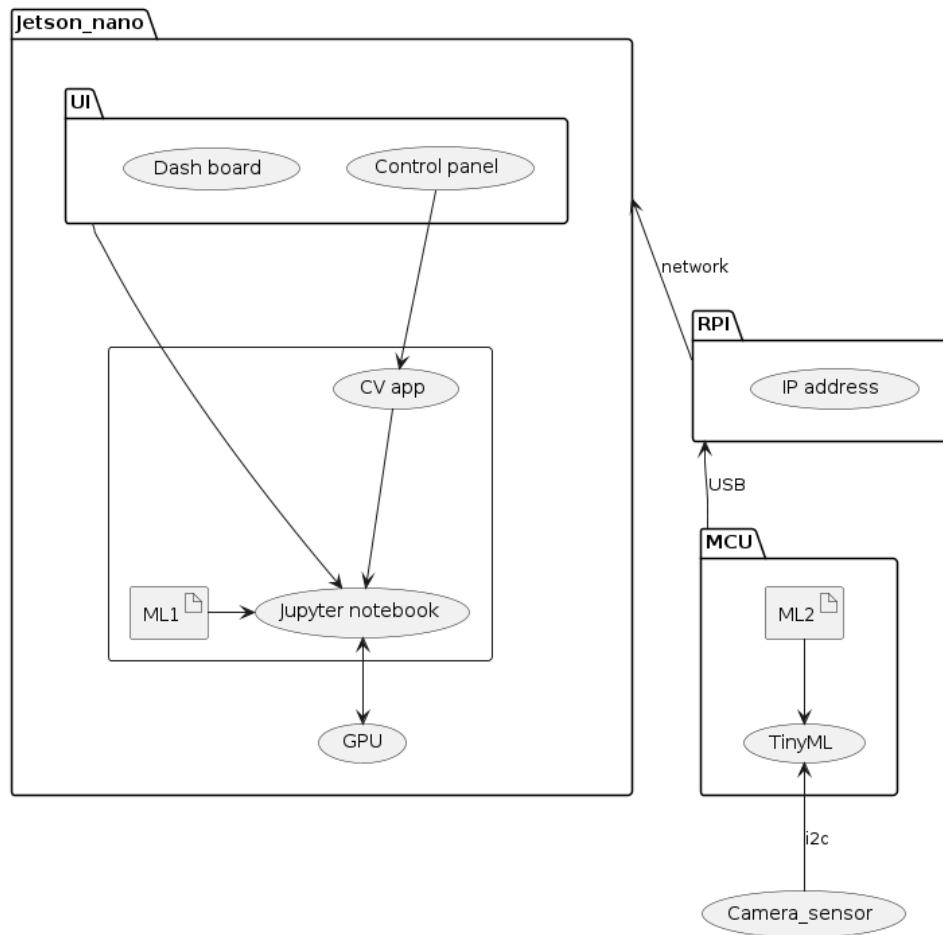- IP address

USB

MCU
- i2c USB converter

i2c

Camera_sensor

MVP3: Offloading with TinyML

Run a small part of ML processing (ML2) as TinyML on MCU.

MVP3 -> MVP4

MVP3: Offloading with TinyML

MVP4: Add Cloud control UI
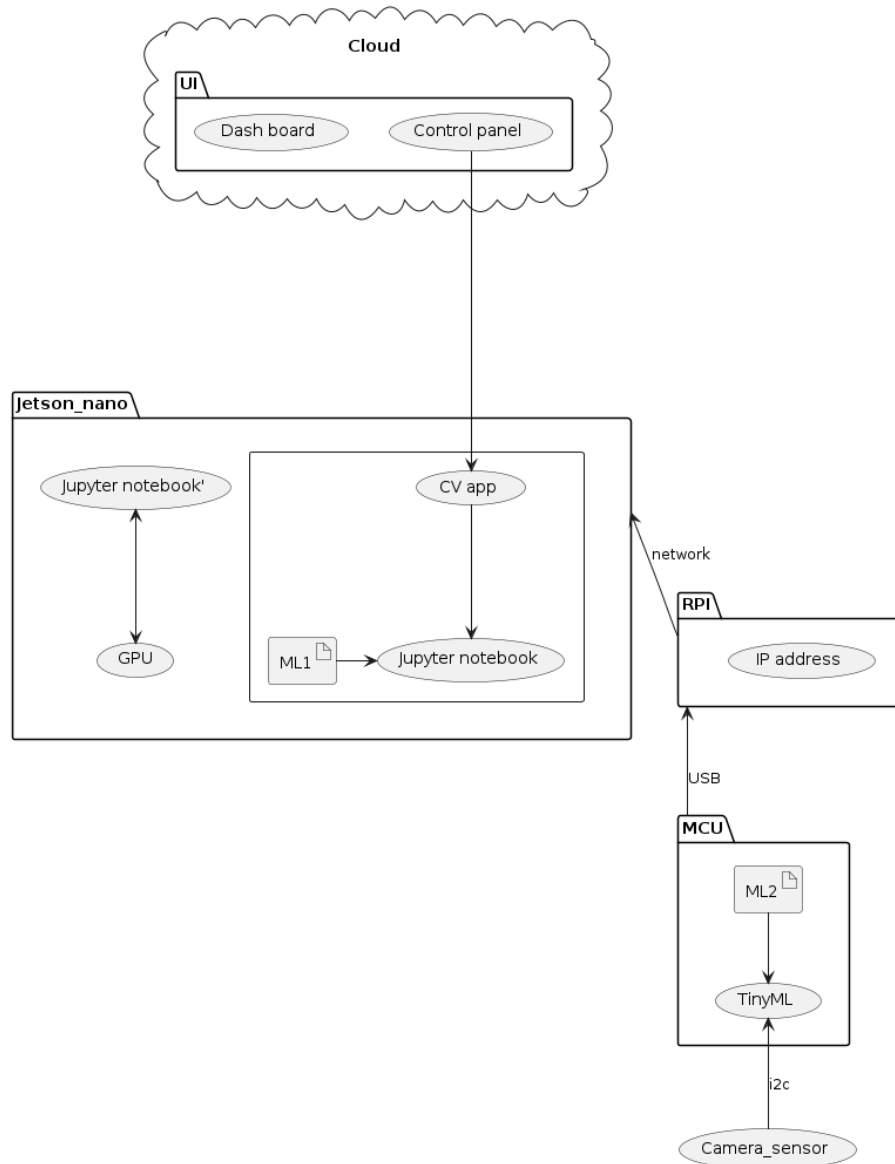
Factor out UI on Cloud. This part could be done independently from TinyML.

MVP4 -> MVP5

MVP4: Add Cloud control UI

Cloud

UI

Dash board   Control panel

Jetson_nano

Jupyter notebook'

CV app

GPU

ML1 → Jupyter notebook

network

RPI

IP address

USB

MCU

ML2

TinyML

i2c

Camera_sensor

Cloud

UI

Dash board   Control panel

network

CV app'

ML0 → Jupyter notebook'

GPUs

Jetson_nano

CV app

ML1 → Jupyter notebook

GPU

network

RPI

IP address

USB

MCU

ML2

TinyML

i2c

Camera_sensor
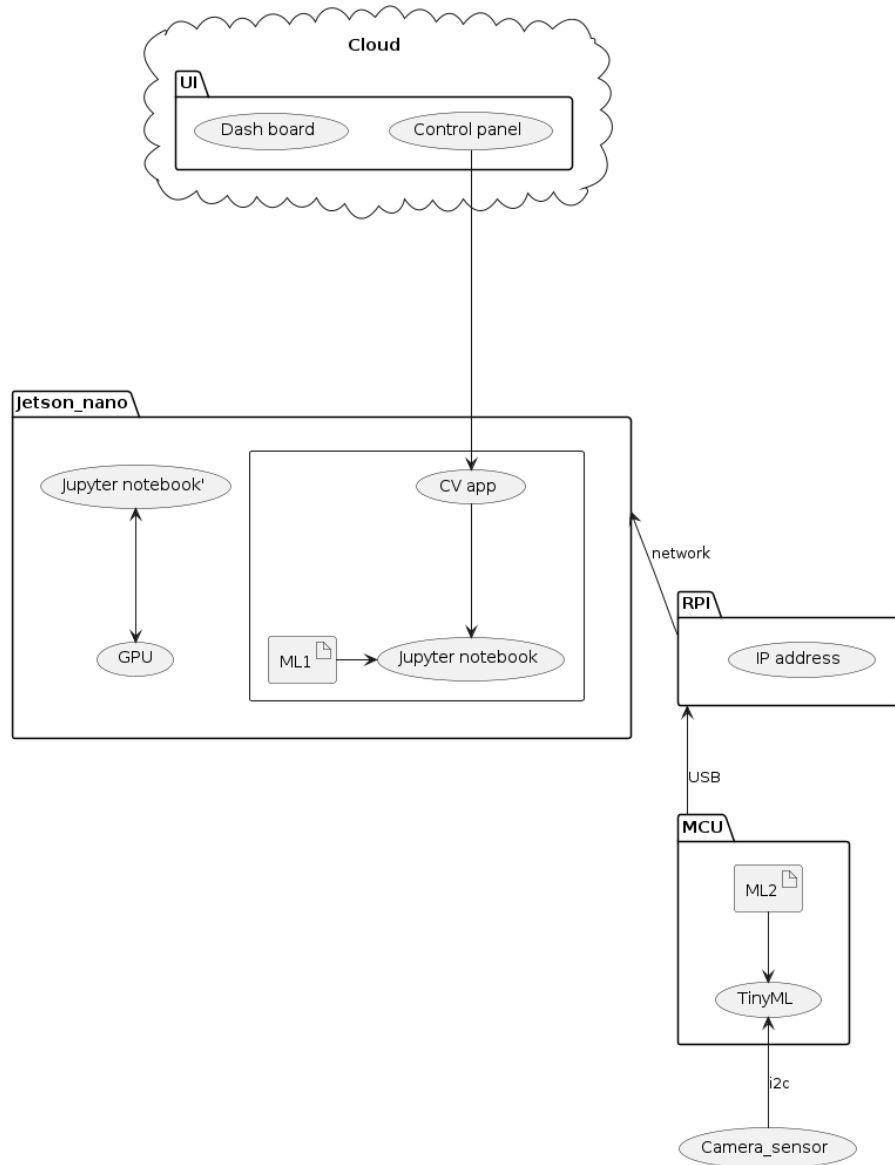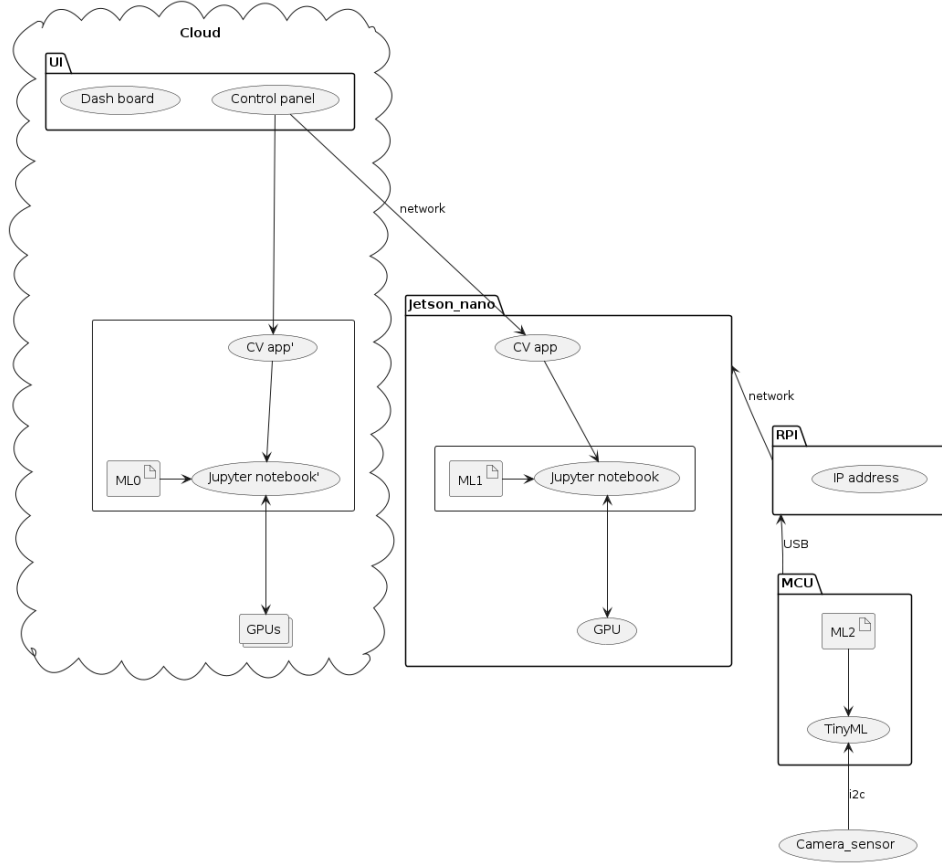
Run some part of ML processing (ML0) on Cloud.

MVP5 -> MVP6

MVP5: Add Cloud ML

Cloud

UI
- Dash board
- Control panel

CV app'

ML0 → Jupyter notebook'

GPUs

network

Jetson_nano

CV app

ML1 → Jupyter notebook

GPU

network

RPI
- IP address

USB

MCU

ML2 → TinyML

i2c

Camera_sensor

Cloud

UI

Dash board   Control panel

network   network

CV app'

Jetson_nano

CV app

RPI

IP address

ML0 → Jupyter notebook'

ML1 → Jupyter notebook

USB

MCU

ML2

GPUs

GPU

TinyML

i2c

Camera_sensor
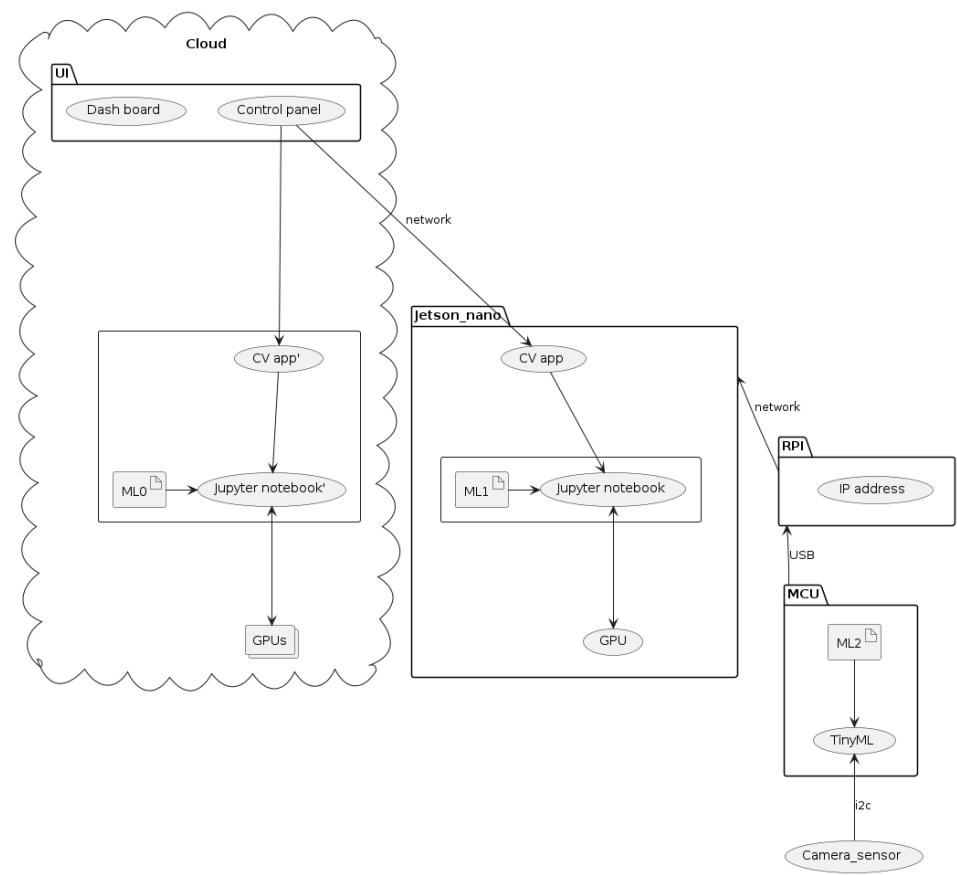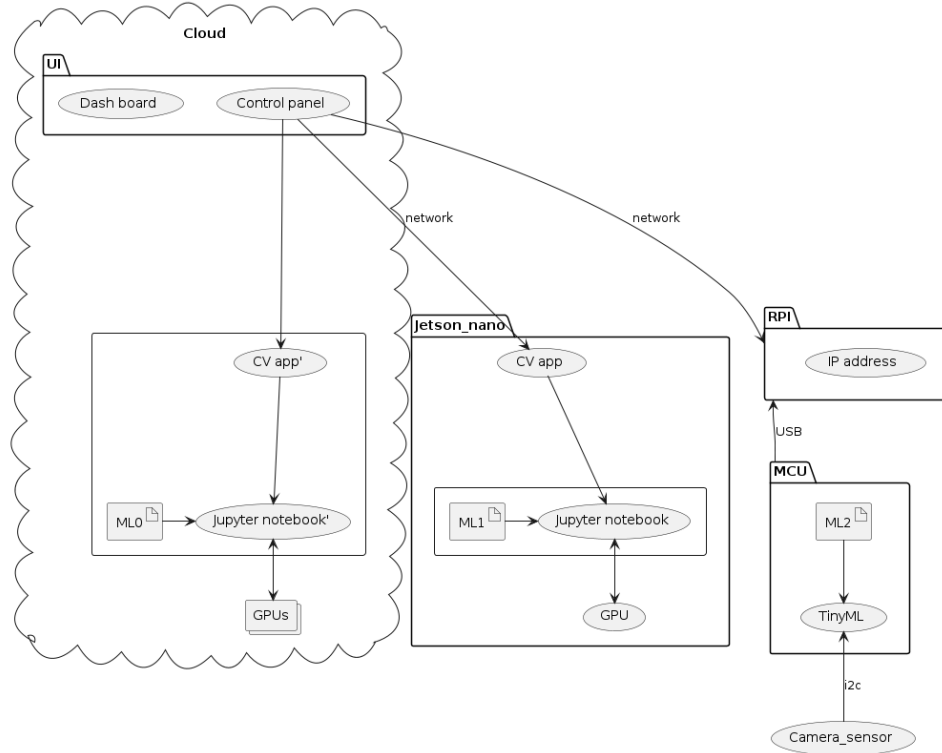
No cascading MLs but parallerizing with nodes.

MVP7

MVP6: Parallerized ML processing

Cloud

UI

Dash board   Control panel

network   network

CV app'   Jetson_nano   RPI
CV app   IP address

ML0 → Jupyter notebook'   ML1 → Jupyter notebook   USB

GPUs   GPU   MCU
ML2

TinyML

i2c

Camera_sensor
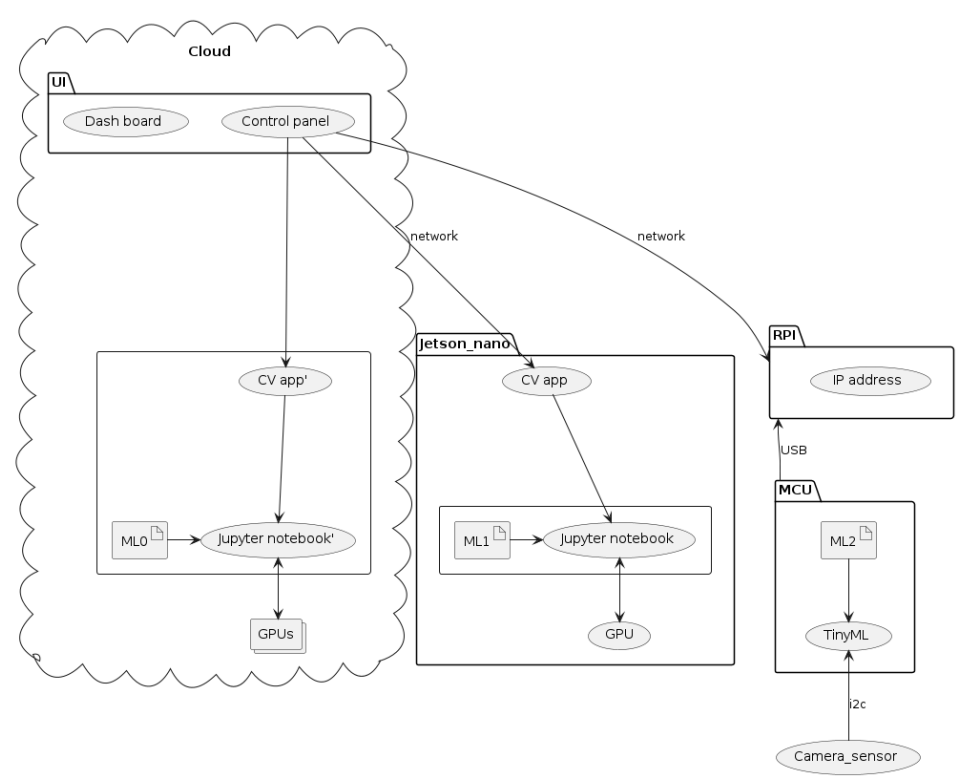
MVP7: TinyML as-a-Service
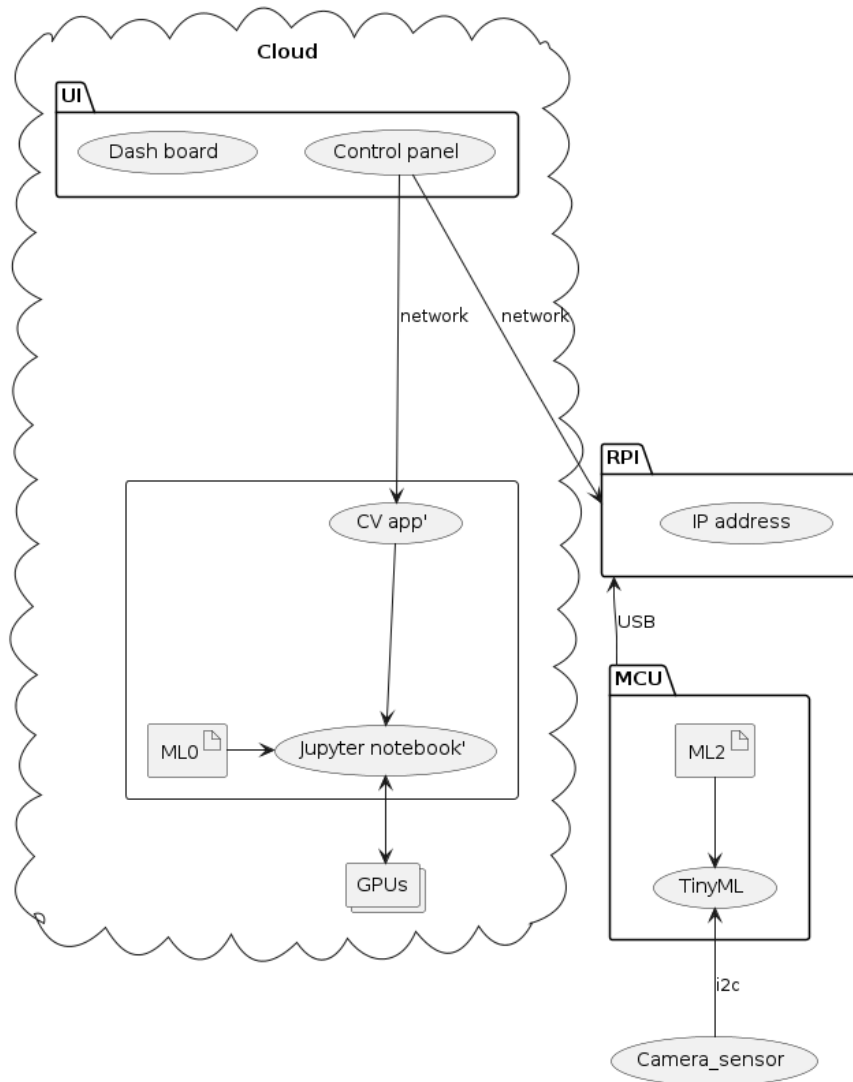
Get rid of Jetson nano but only with TinyMLaaS.

Kick-off meeting Agenda (1/3)

- Scheduled on 16th JAN (MON)

Get familiar with all participants. Everyone introduces oneself

1. What one can do
2. What one wants to do
3. How one sees this project

Will explain Project goal

- We should present demo video1 & demo video1

Set up a SCRUM team (e.g. specify each role in the project)

- SM: HU instructor + Michihito Mizutani
- PO: Roberto Morabito
- Developer: 5 students
- ML support: Hiroshi Doyu
- Customer: Perttu & Samuli would review the outcome of increment in bi-weekly meeting.

Kick-off meeting Agenda (2/3)

User story mapping

- Specify PBIs always as GitHub issues, which need to be a PR and it automatically runs CI/CD as acceptance tests.
- Estimate PBI effort
- Specify acceptance tests
- 1 increment == 2 sprint
- 1st sprint planning should be done on 16th.

Kick-off meeting Agenda (3/3)

Agree on WoW in SCRUM

- Use Github project KANBAN
- Use Discord channel to communicate or Slack?
- Agree on scheduling a Daily meeting day & time
- 1 increment == 2 sprint
- 1st sprint should have some Architecture investigation to find out which components are reusable.

- 1st sprint should have a ZFR (Zero Feature Release) to make sure that CI/CD works on Github workflow (action) without any features (or just with existing components)
- We should run CI/CD to reproduce the current Roberto's demo story at first, without a training part. If HW is not available, it could be simulated.

Architecture investigation @Roberto?

- How should we understand Roberto's demo architecture?
    - Any architecture document?
    - Any architecture block diagram?
    - Any flow diagram?
    - Any list of components used? which could be reused and which not, since the outcome of this project would be opensource'ed.
    - Any list of software frameworks each components uses?
- Should we map each Seamless TinyML lifecycle management's phase on this demo scenario?
- Should we make sure which phases are still missing? (e.g. "2. Model training")

Mapping 6 Seamless TinyML lifecycle to this project

Phase #	Name	Early phase	Demo
1	Data collection	Simulated	RPI pico + Cam
2	Model training	Missing	On Cloud VM?
3	Model squeezing	ML compiler	ML compiler
4	Model splitting	Standalone	Pipelining
5	Model deployment	Standalone	TBI
6	Model update	Dashboard	Control panel

Remaining questions

- What's the main purpose of this project from students' perspective? (e.g. experience Agile development)
- How long can the 16th meeting be allocated? (e.g. 3-4 hours with lunch break)

- What kind of competency are students generally expected? (e.g. Python, JS, Java, Frontend, Embedded)
- Can student's weekly working hours be negotiable? (e.g. 15 hours with 14 weeks -> 17.5 hours with 12 weeks)

Contact information

Origami

https://Origami-TinyML.github.io/blog/about.html