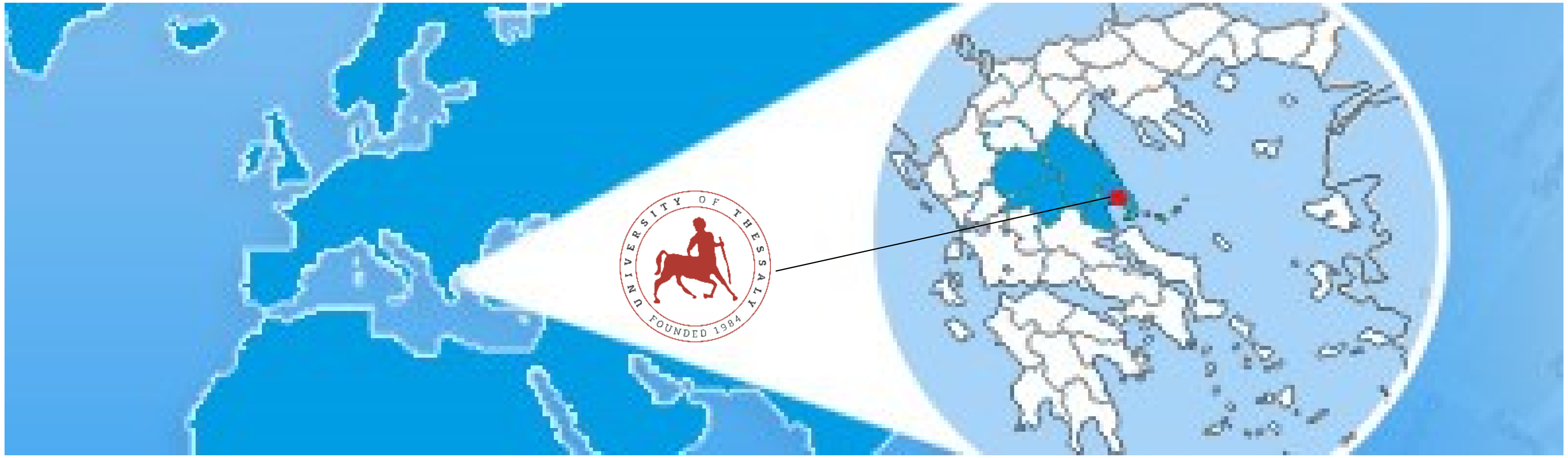
Elevating drones as first-class citizens in the cloud-edge-IoT continuum

Spyros Lalis

Computer Systems Laboratory
University of Thessaly,
Volos, Greece

Electrical & Computer Engineering Dept. @ UTH

- Founded 2000
- 5-year eng. diploma
- 25 faculty members
- 200 students annually



Computer Systems Lab @ ECE.UTH

- Profs + PostDoc

- PhD students

- Alumni

- System software (OS, runtime environments) from embedded to HPC systems
- Approximate computing
- Accelerated & Reconfigurable computing
- Energy-aware computing
- Power / performance optimization
- Distributed & ubiquitous computing

- Significant EC & national funding
- Several international collaborations

Special acks for the work that follows

Manos Koutsoubelias

Nasos Grigoropoulos

Giannis Badakis

Giorgos Polychronis

Foivos Pournaropoulos

Alexandros Patras

Auto-Scout

MLSysOps

European Union
European Regional Development Fund

HELLENIC REPUBLIC
MINISTRY OF ECONOMY & DEVELOPMENT
SPECIAL SECRETARY FOR ERDF & CF
MANAGING AUTHORITY OF EPAnEK

EPAnEK 2014-2020
OPERATIONAL PROGRAMME
COMPETITIVENESS
ENTREPRENEURSHIP
INNOVATION

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη
Partnership Agreement 2014 - 2020

Co-financed by Greece and the European Union

European Commission | HORIZON EUROPE

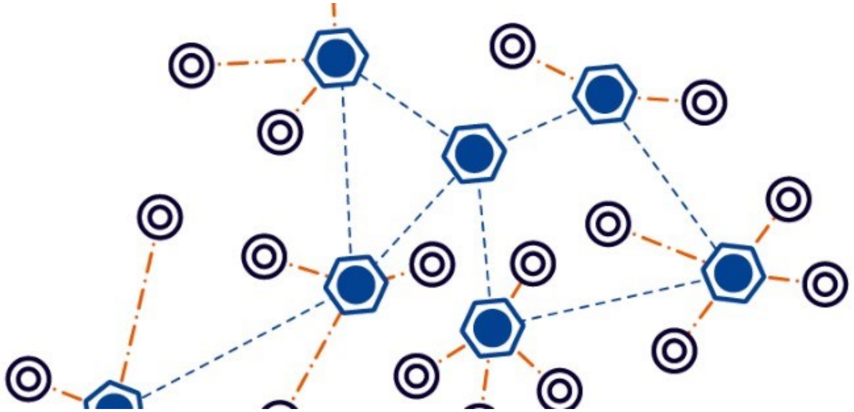
Overview

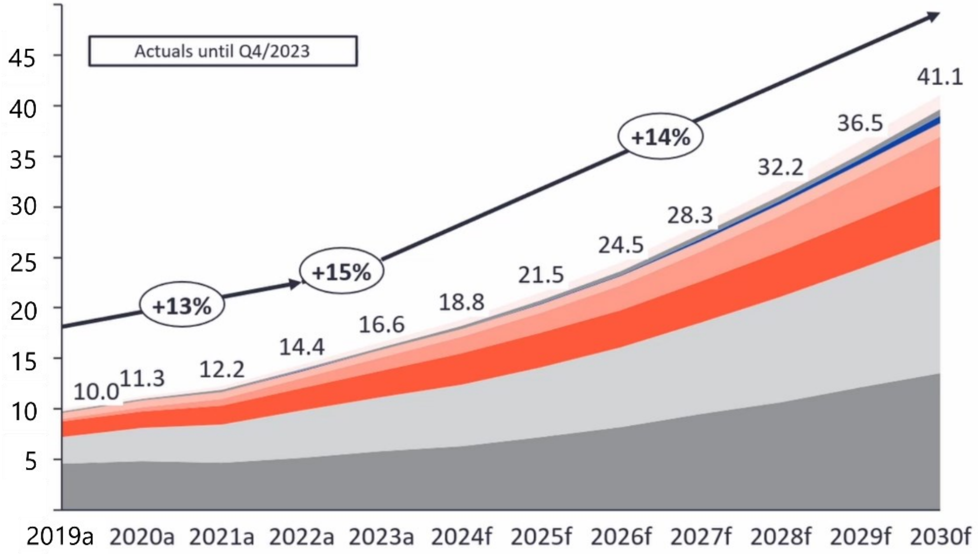
- Motivation
- Selected Topics
  - Precision landing
  - Drone-based remote sensing architecture
  - Flexible application deployment & orchestration
  - Drone usage in the MLSysOps project
- Wrap-up

Motivation

IoT

- Billions of connected devices
- Producing huge amounts of data
  - Some estimate 80 ZB by 2025
- The trend will increase
- Not possible (or desirable) to transmit & process everything in the cloud





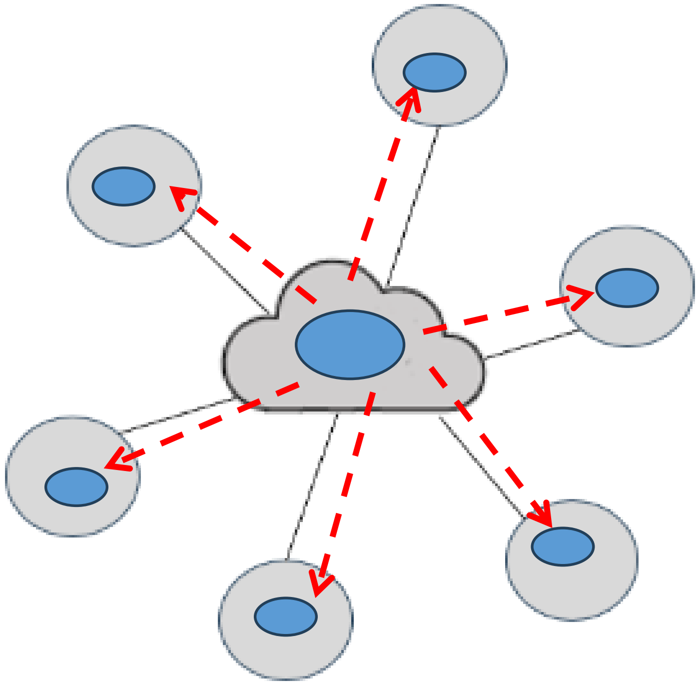
https://iot-analytics.com/wp/wp-content/uploads/2024/09/Global-IoT-market-forecast-Number-of-connected-IoT-devices-Sep-2024.mp4

Edge computing

- Move data processing out of the cloud
- Towards the so-called edge
- Close(r) to the data sources

- Regional or on-premise data centers
  - smaller clouds

- Standalone base stations / servers or the IoT devices themselves
  - resource-constrained, heterogenous



Drones become increasingly popular

- More affordable
- Can be equipped with different compute/communication HW
- As well as different sensors & actuators

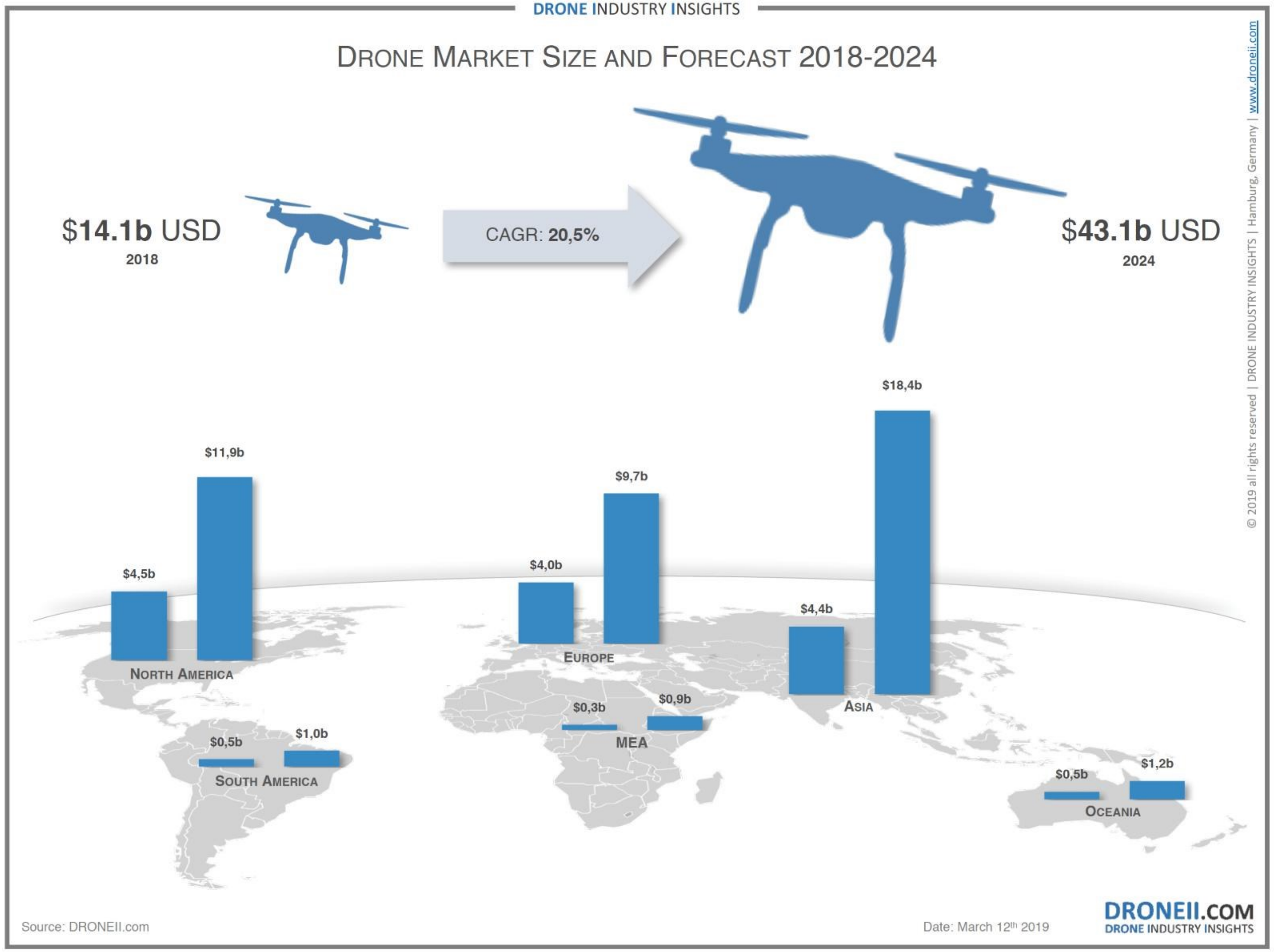
- Wide range of civilian applications

- Polycopters
  - Easy to operate
  - Vertical take-off/landing
  - Hovering over position of interest

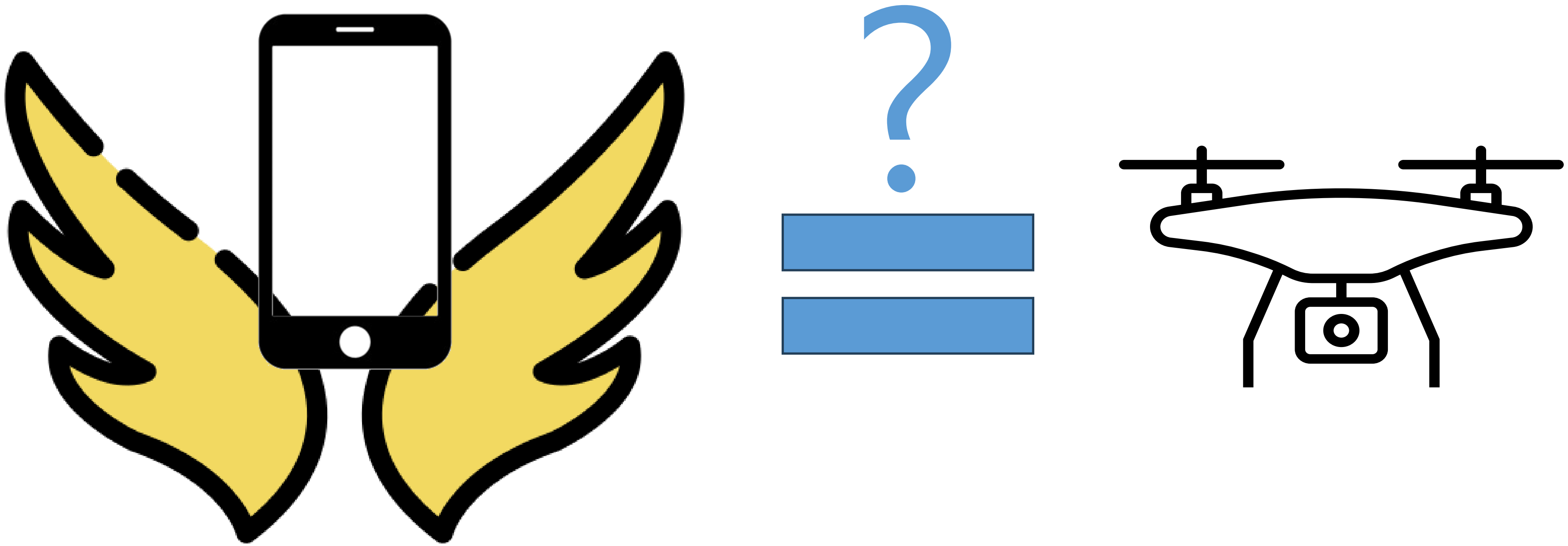




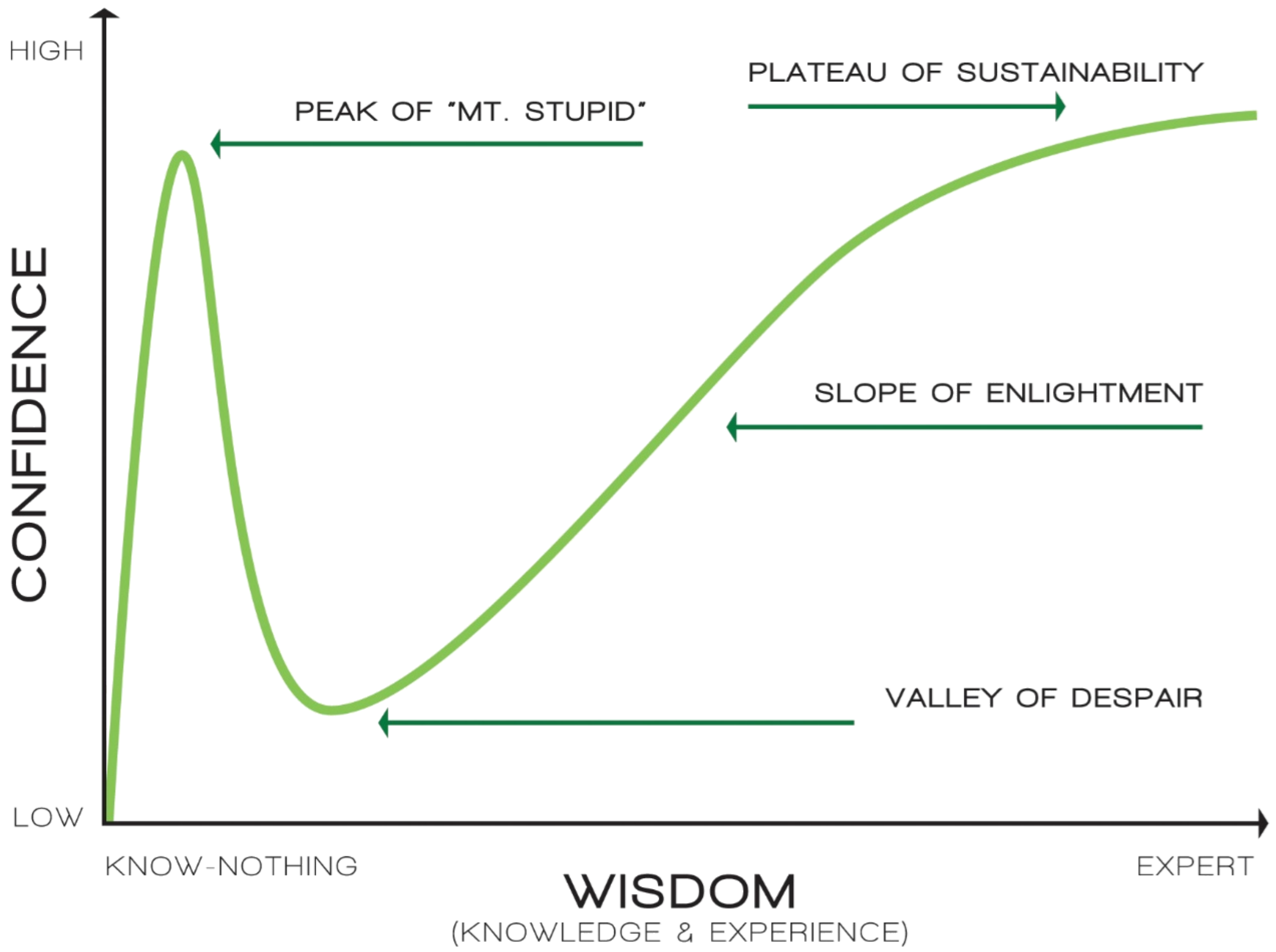
Growing business sector



The next ubiquitous IoT device?



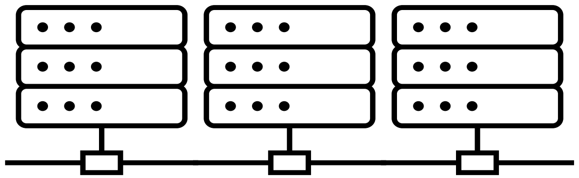
Maybe …



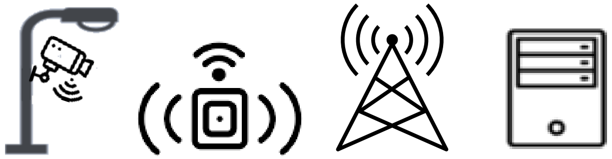
HIGH
LOW
CONFIDENCE
PEAK OF "MT. STUPID"
PLATEAU OF SUSTAINABILITY
SLOPE OF ENLIGHTMENT
VALLEY OF DESPAIR
KNOW-NOTHING
EXPERT
WISDOM
(KNOWLEDGE & EXPERIENCE)

Drones as part of the system infrastructure

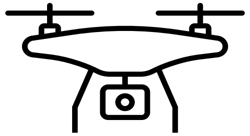
- Server machines in datacenters
  - computing/storage resources



- IoT devices & standalone machines
  - sensing/actuation resources
  - computing/communication resources



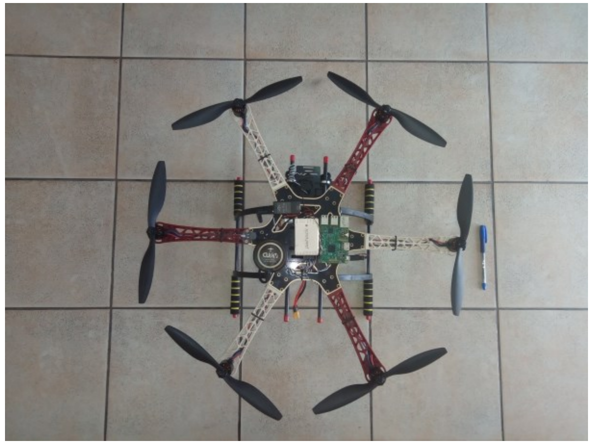
- Drones
  - sensing/actuation resources
  - computing/communication resources
  - can fly directly above the region/points of interest
  - significant flexibility & coverage



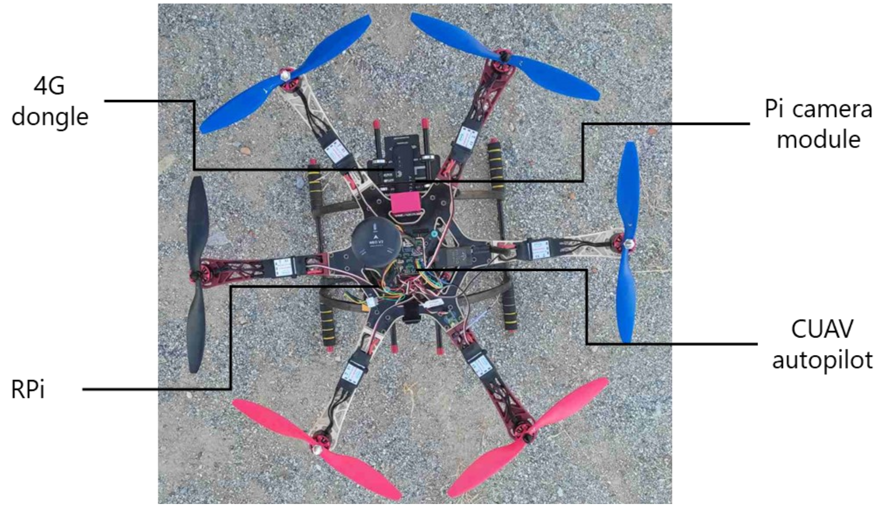
Main goals of this talk

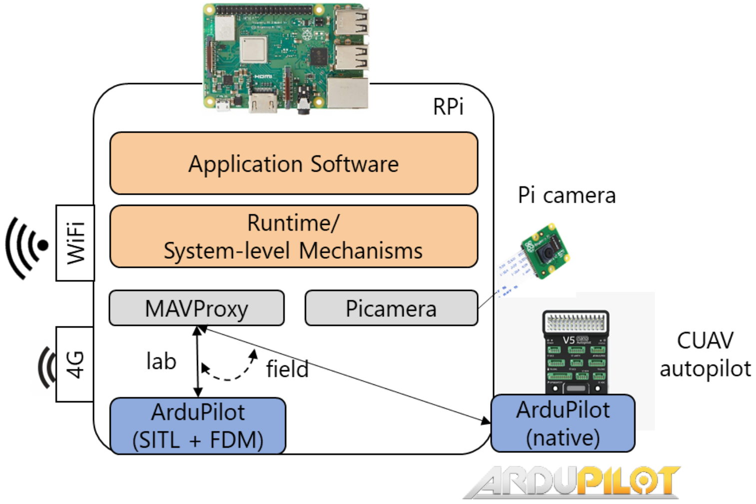
- Convince you that civilian drones are interesting IoT devices
- Show that these are different “animals” requiring extra care
- Indicate how these can be exploited as part of a greater system infrastructure
- Tease you to consider these animals in your research for edge-centric systems

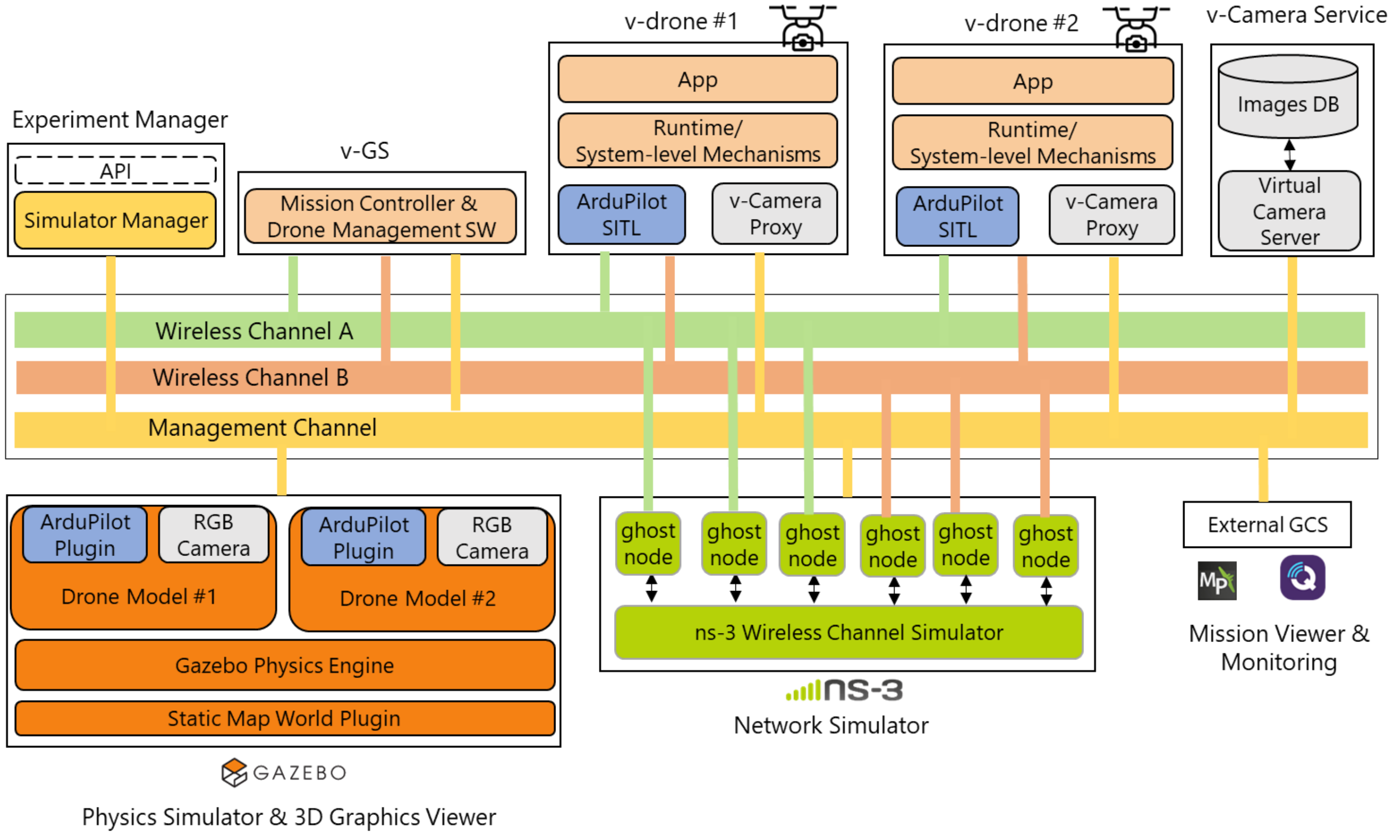
The players



Experimentation







Topics

- Precision landing
- Drone-based sensing architecture
- Flexible application deployment & orchestration
- Drone usage in the MLSysOps project

Precision Landing

Landing pads & hangars

- Recharge the drone's batteries
- House the drone between missions

- Important element for making drones part of the system infrastructure

- Support multiple (consecutive) takeoff-sense-land-recharge cycles
- Reduce/eliminate human intervention
- Potential for remote installations

KDDI smart drone – Skycharge Source

Skycharge hangar solution

Need precision landing

- The drone must land inside a small area
- Conventional GPS can be (very) inaccurate

- RTK GPS, infrared beacons (IRLock)
- May have issues due to interference

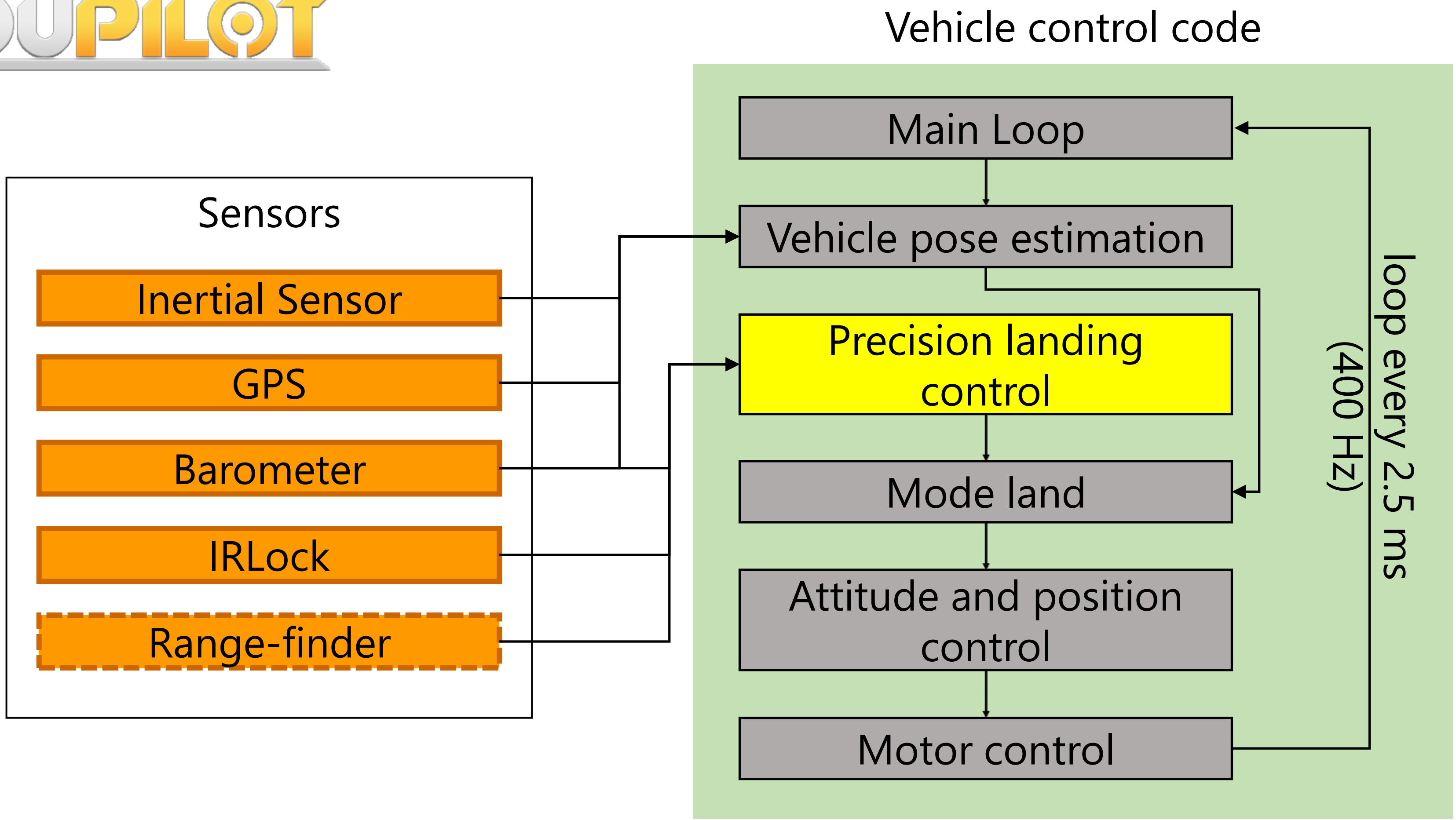
- Any single precision landing sensor may fail

Our work

- As one more option, we investigate visual markers
- Use the drone’s conventional camera to detect a specific tag
- Run the detection code on the drone’s onboard computing platform (RPi)
- Integration with the autopilot framework

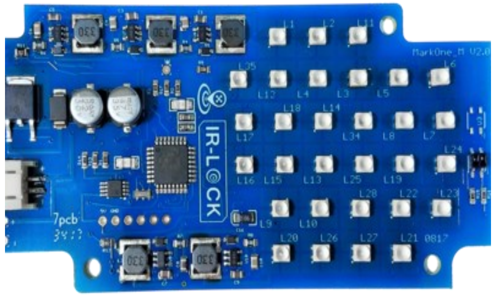
- Combine/fuse with another precision landing sensor (IRLock)
- Fault tolerance to individual sensor failures
- New landing modes to exploit new capability

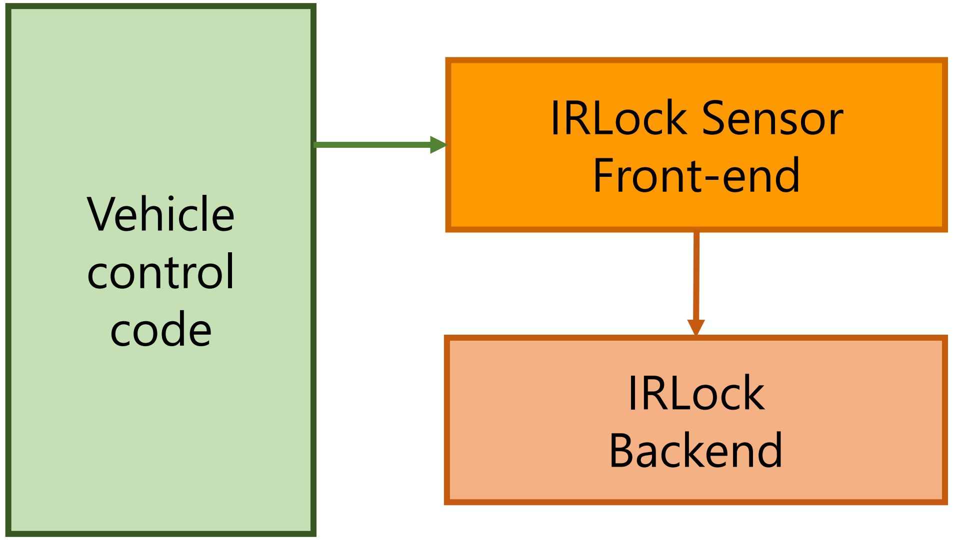
Control flow with precision landing enabled



IRLock subsystem

- I2C protocol
- 1 frame every 20ms (50Hz)
- Detection range 12-15 meters



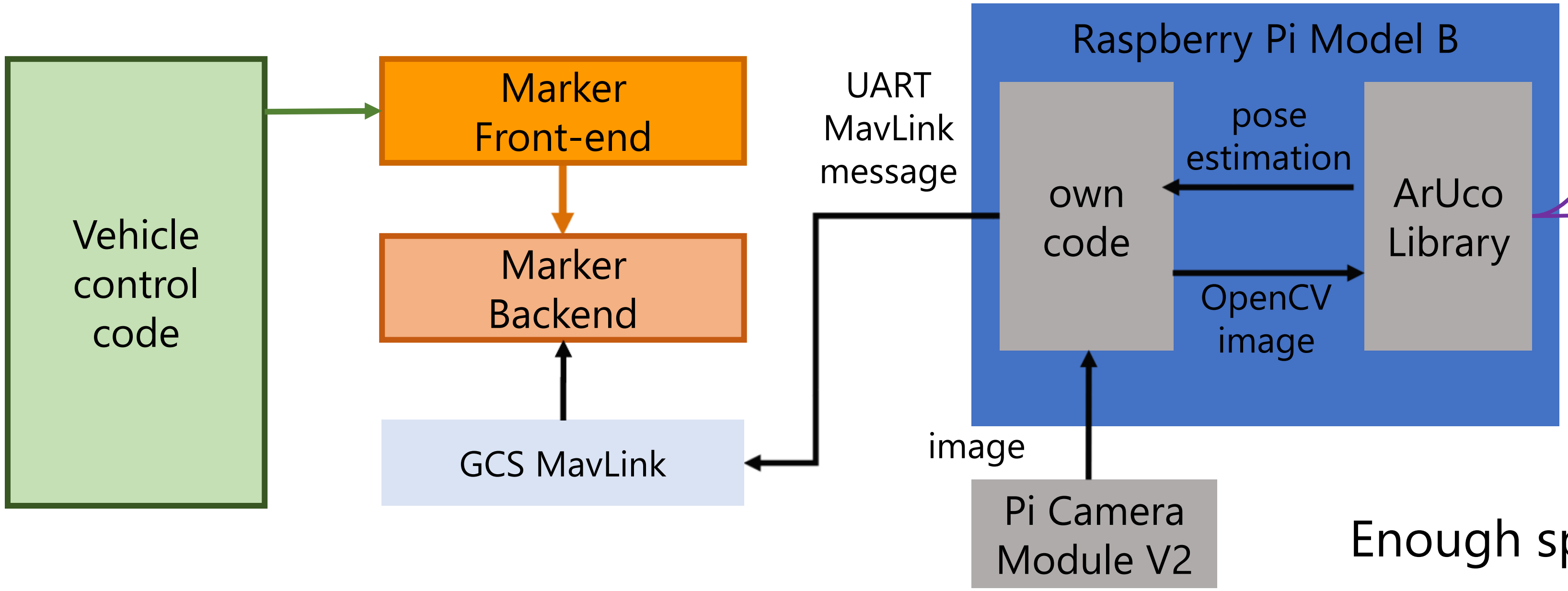


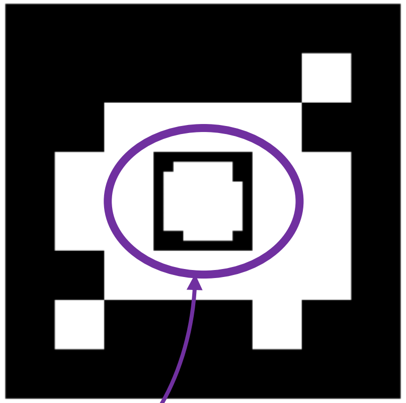
Bytes	16-bit words	Description
0, 1	0	Sync (0xaa55)
2, 3	1	Checksum (sum of all words 2 – 6)
4, 5	2	Signature number
6, 7	3	X center of object (pixel)
8, 9	4	Y center of object (pixel)
10, 11	5	Width of object (pixels)
12, 13	6	Height of object (pixels)

Marker sensor subsystem

- Pi camera configured at 640x480
- Pose estimation every ~50ms (20 Hz)
- Send information using the MavLink protocol over serial/UART

- ArUco custom fractal marker
- Enables detection from different heights

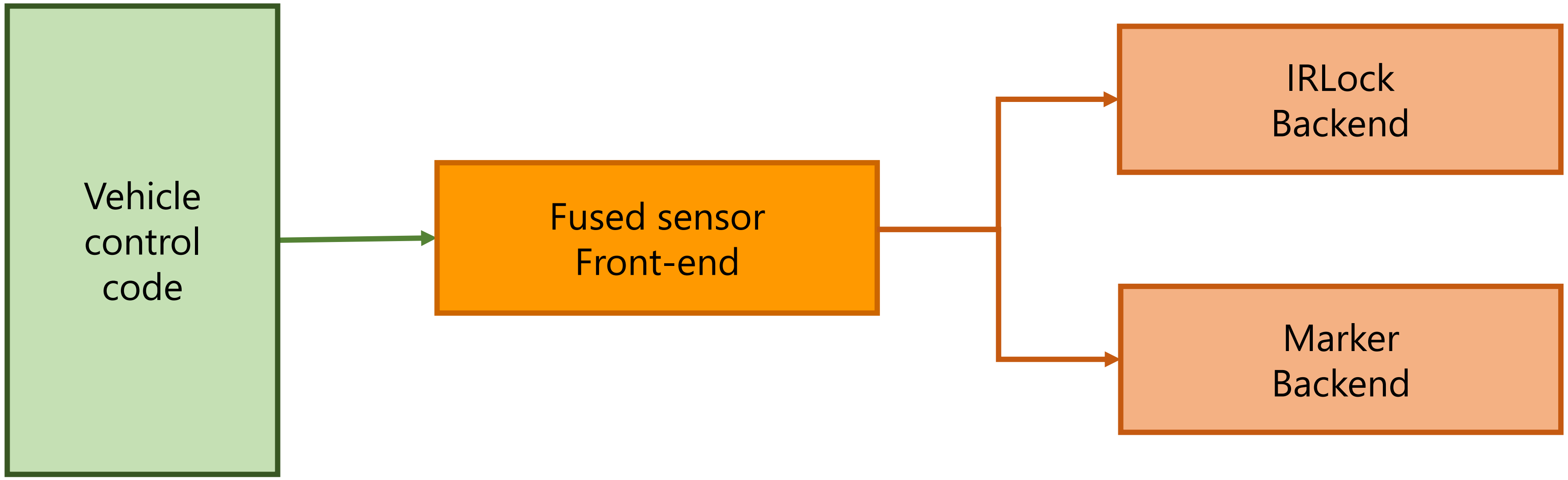




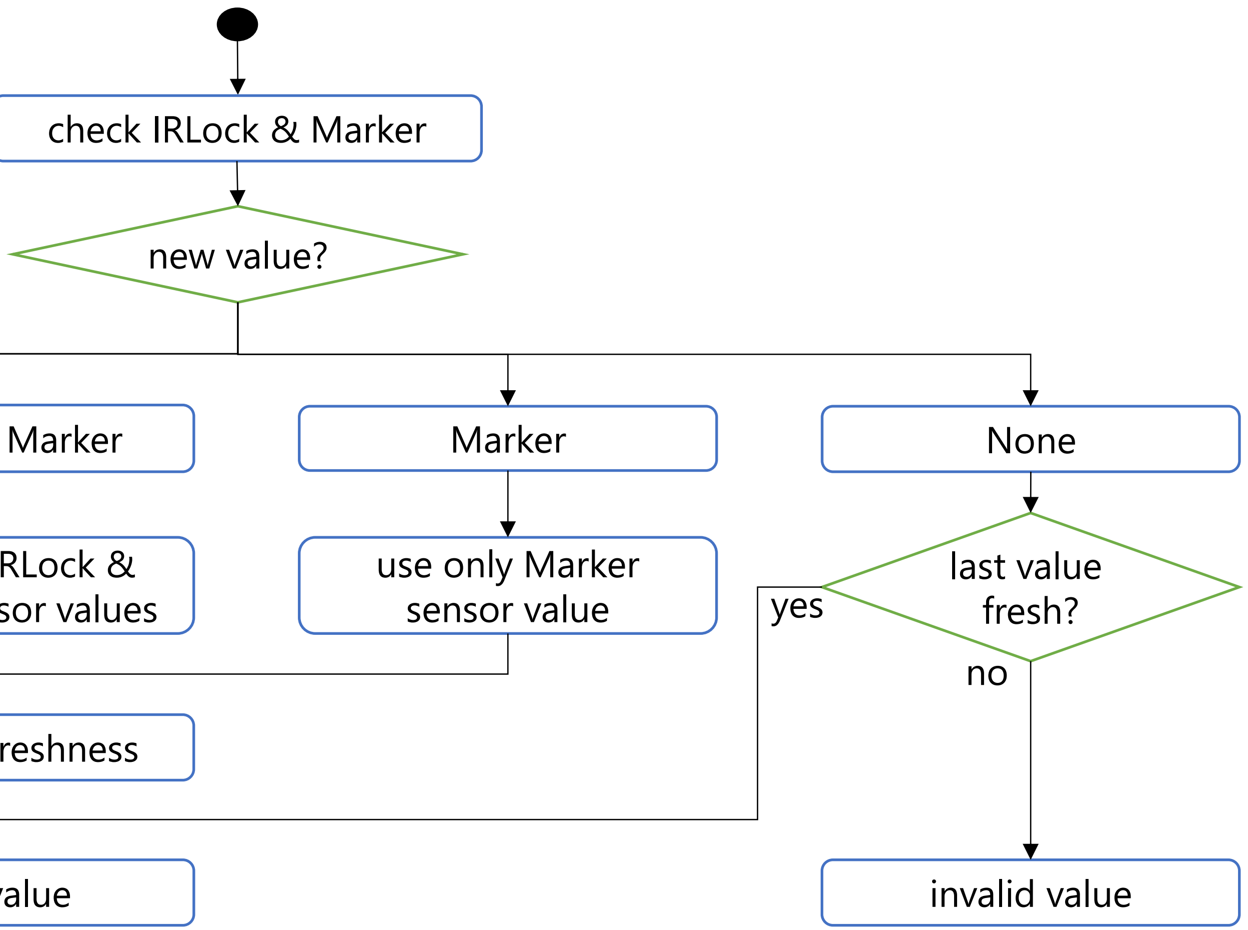
Enough space for the IR beacon

Fused sensor

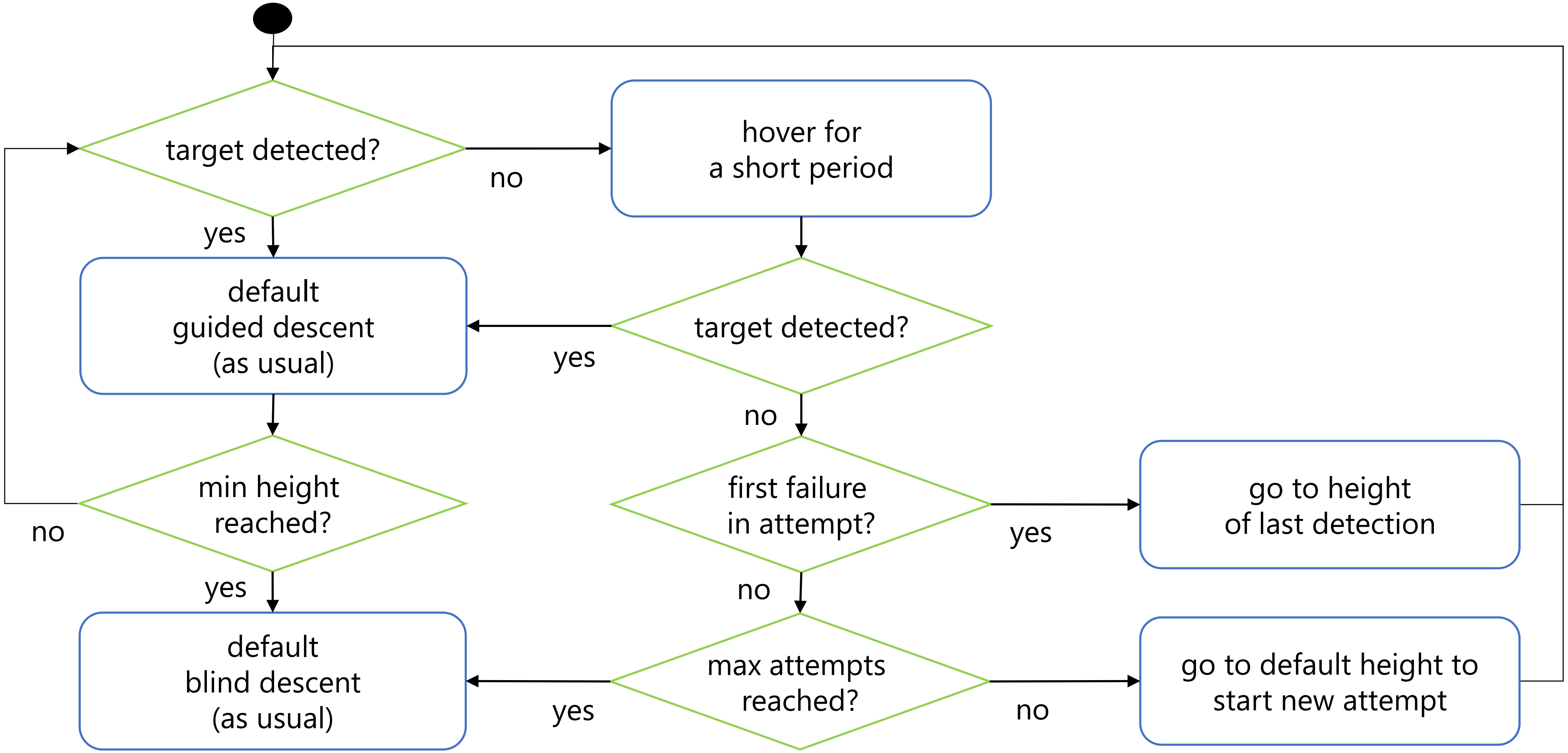
- Front-end hides all back-end details
- Does not have its own back-end
- Connects to the back-ends of the IRLock and Marker sensor subsystems



Fusion logic



Cautious land mode



Controlled (artificial) failures

- Drop modes
  - random
  - periodic

- Modified the front-ends of the respective precision landing sensor subsystems
- Drop sensor values produced by the respective back-ends

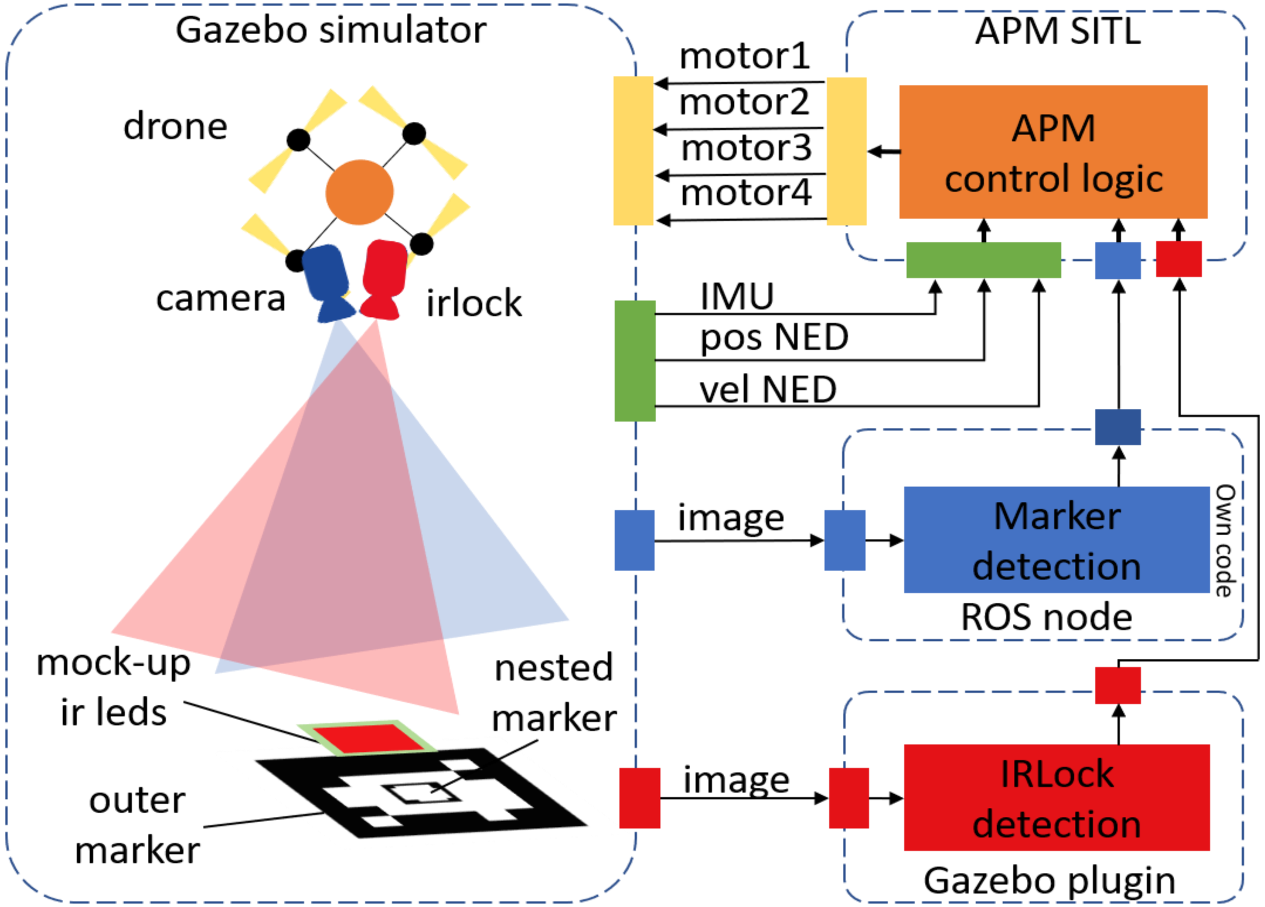
- Configuration at runtime via a MavLink command

Field	Type	Description
target	uint8_t	Target sensor subsystem (IRLock: 1, Marker: 2, both: 3)
r_drop	float	Probability for dropping a new sensor value
p_keep	uint16_t	Number of consecutive new sensor values to keep
p_drop	uint16_t	Number of consecutive new sensor values to drop

MavLink message

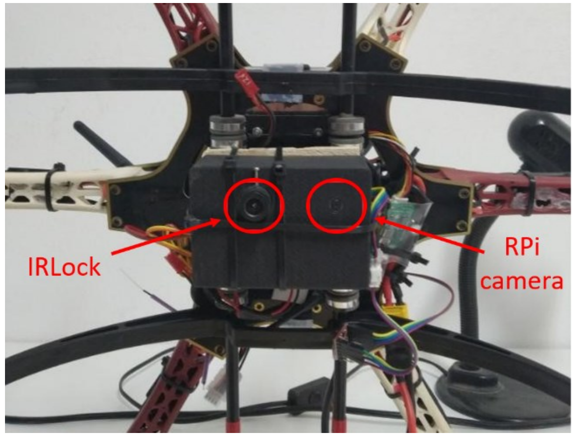
Simulated setup

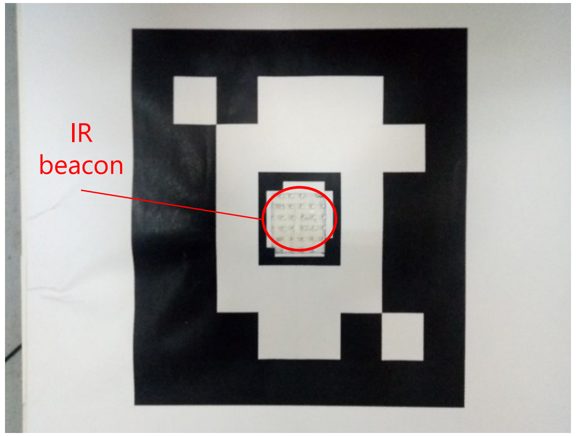
- Gazebo acts as a flight-dynamics simulation engine for the APM autopilot
- SITL platform configuration for the APM autopilot, derived from the same code base that targets real controllers
- Gazebo and APM communicate via UDP/IP



Drone setup

- Autopilot CUAV Nano v5
  - ICM20689 accel/gyro
  - ICM20602 accel/gyro
  - BMI055 accel/gyro
  - IST8310 magnetometer
  - MS5611 barometer
- Neo v2 GPS/Compass
- Raspberry Pi & camera
- IRLock target tracking system



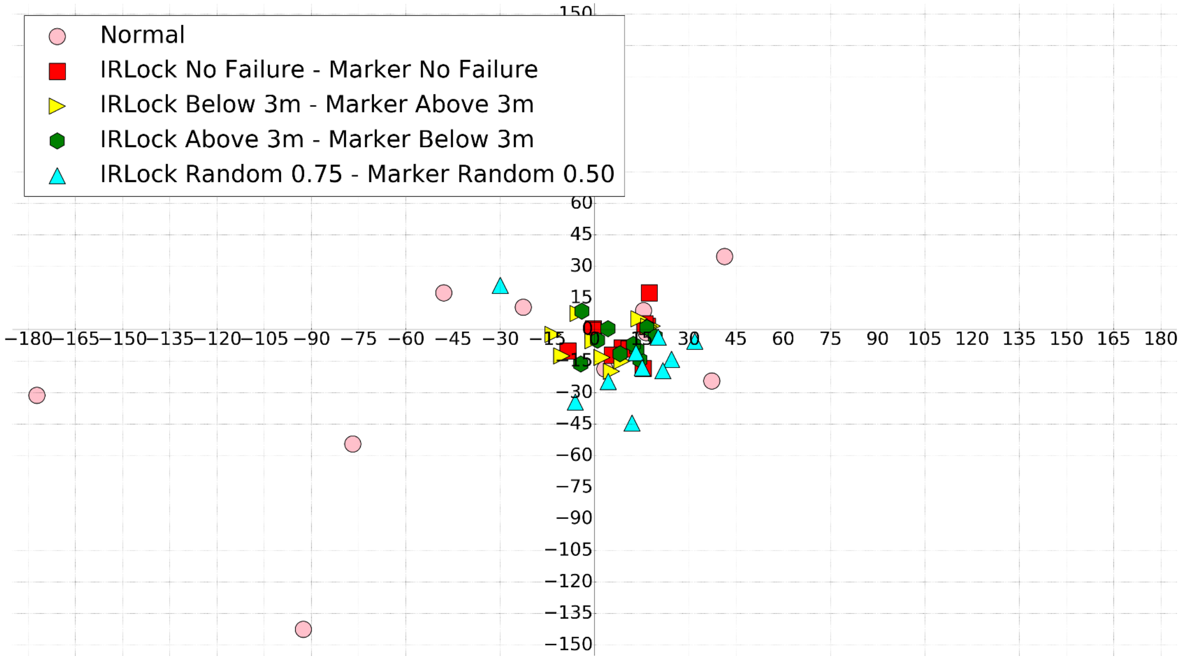


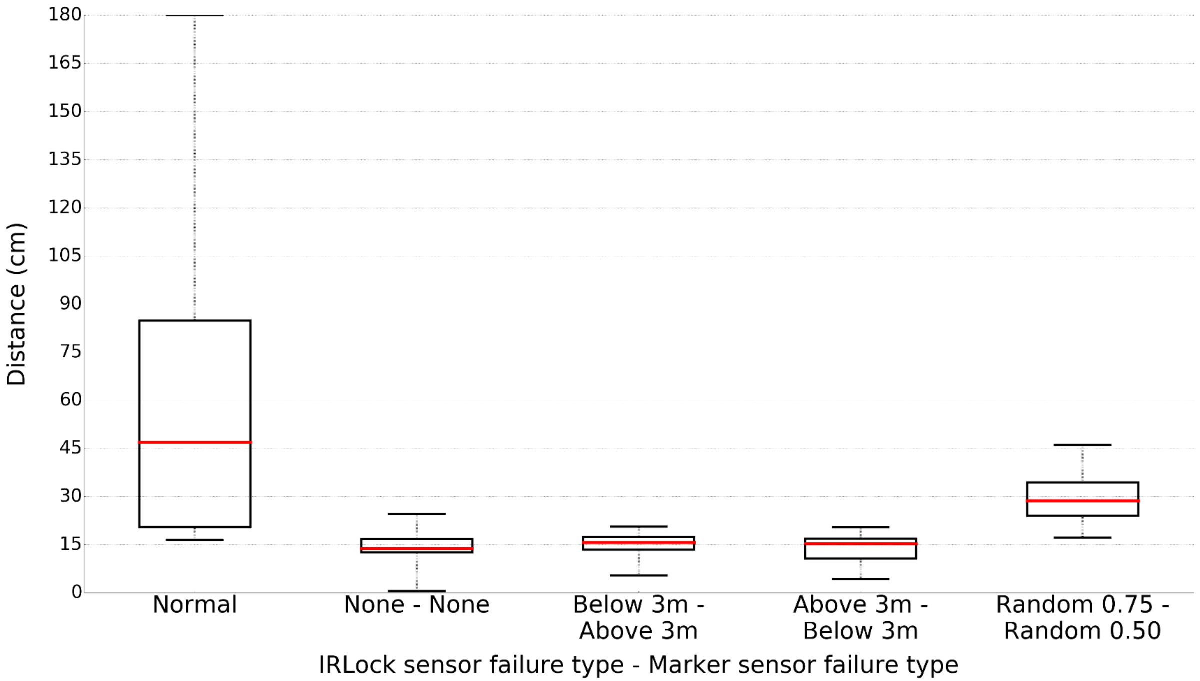
Field tests

- The drone takes-off from a position outside the landing area
- Makes a small tour
- Returns to the home waypoint
- Once this is reached, precision landing is activated
  - guiding the drone on top of the visual marker and IR beacon

- Experiments are performed for different failure scenarios
  - values are dropped from one or even both sensors in a controlled way

Results


Normal
IRLock No Failure - Marker No Failure
IRLock Below 3m - Marker Above 3m
IRLock Above 3m - Marker Below 3m
IRLock Random 0.75 - Marker Random 0.50


Distance (cm)
Normal
None - None
Below 3m - Above 3m
Above 3m - Below 3m
Random 0.75 - Random 0.50
IRLock sensor failure type - Marker sensor failure type

Drone-based remote sensing architecture

Drones as a flexible remote sensing resource

- Target area to be monitored periodically or on demand
- Use one or more drones to perform the required sensing
- Process the results to detect events/phenomena of interest

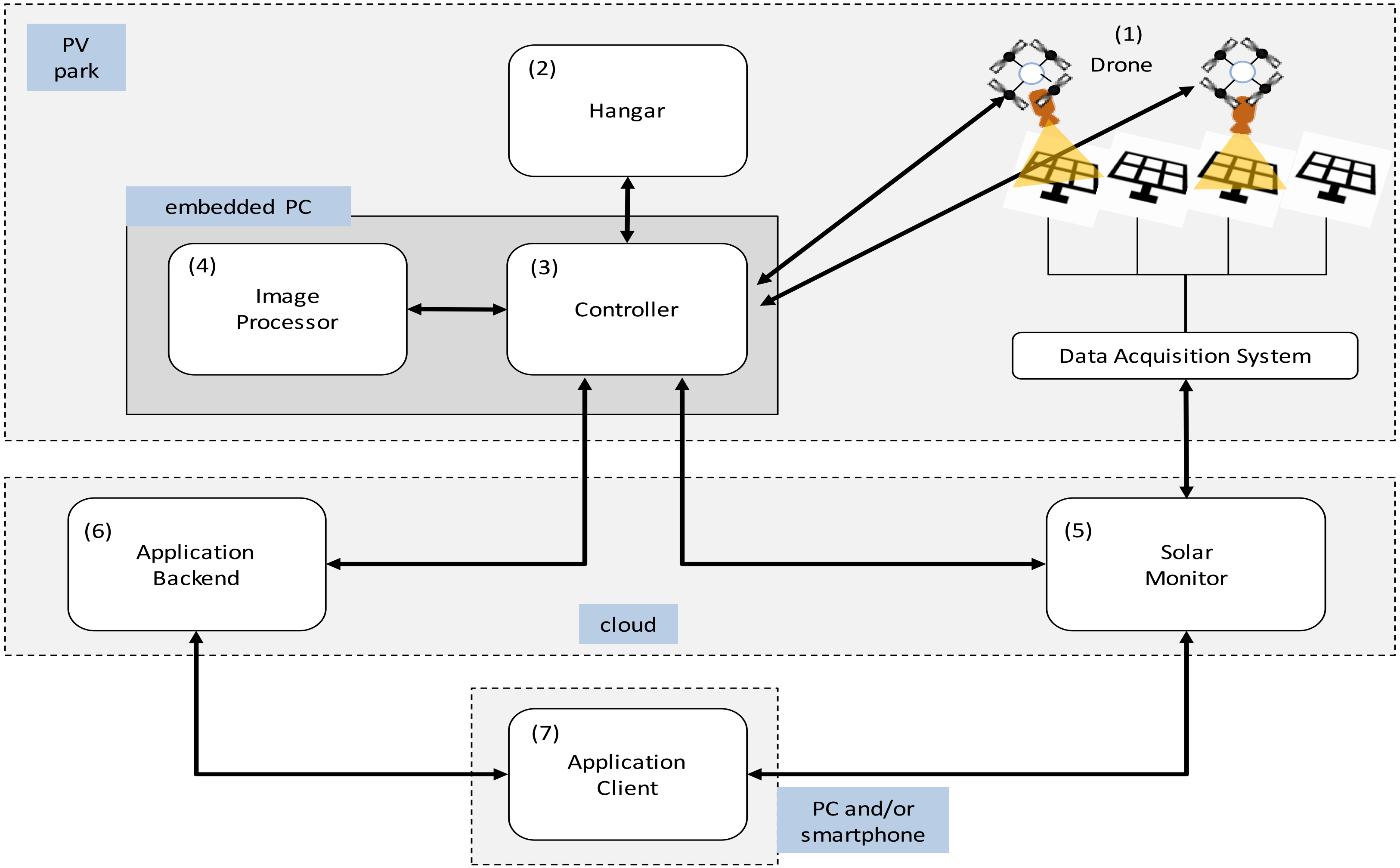
<u>Application use case</u>

- Panel inspection in a solar park
  - periodically
  - when receiving a user request
  - when receiving a signal from the park's electrical monitoring system
- Process IR images to detect issues on panels (e.g., cracks, hotspots)

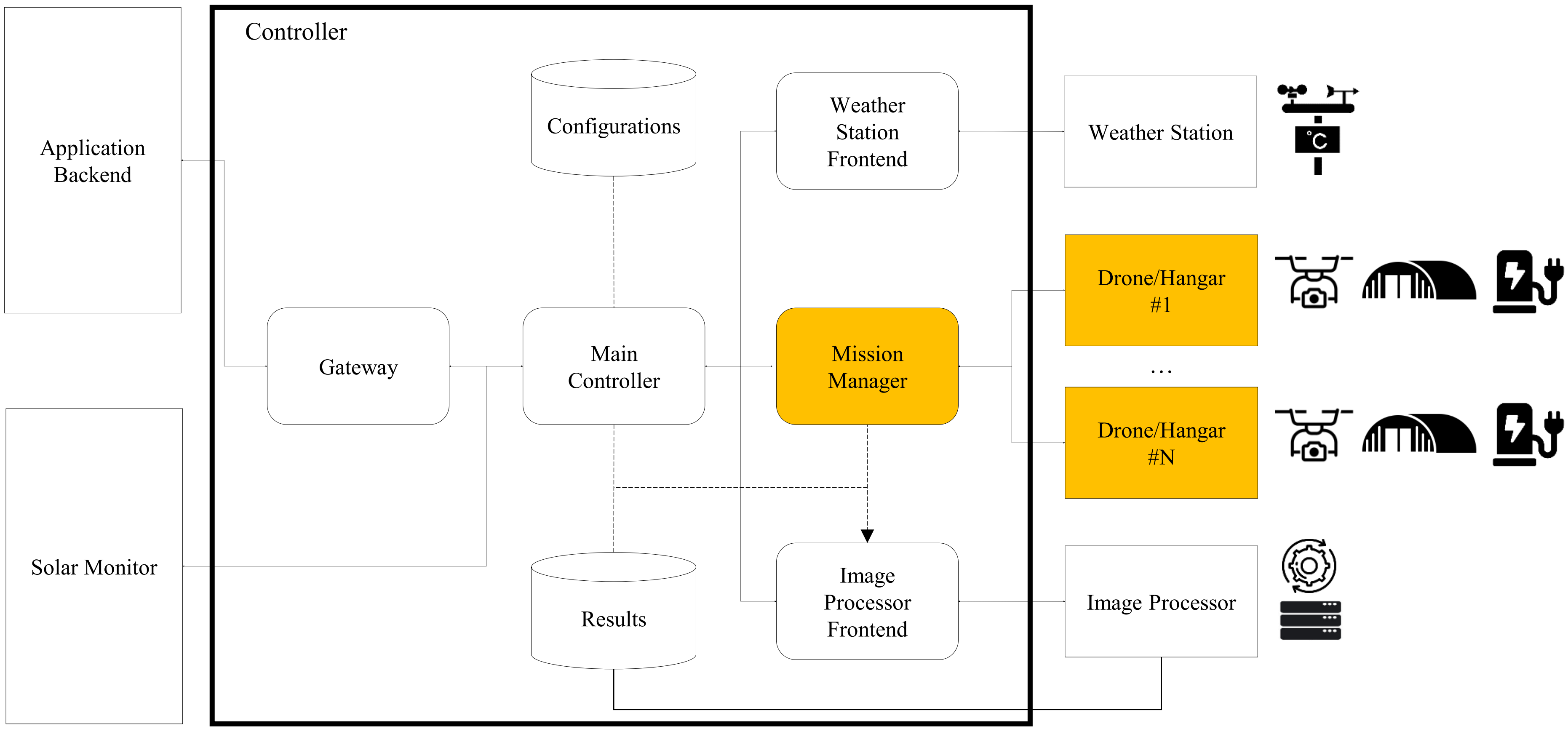
Edge-oriented approach

- Drone housed in a hangar, also used to charge the drone's batteries
- The drone & hangar are part of the park infrastructure
- The entire management logic runs in an embedded computer
  - can be placed inside the hangar
- Even image processing runs locally

Top-level architecture

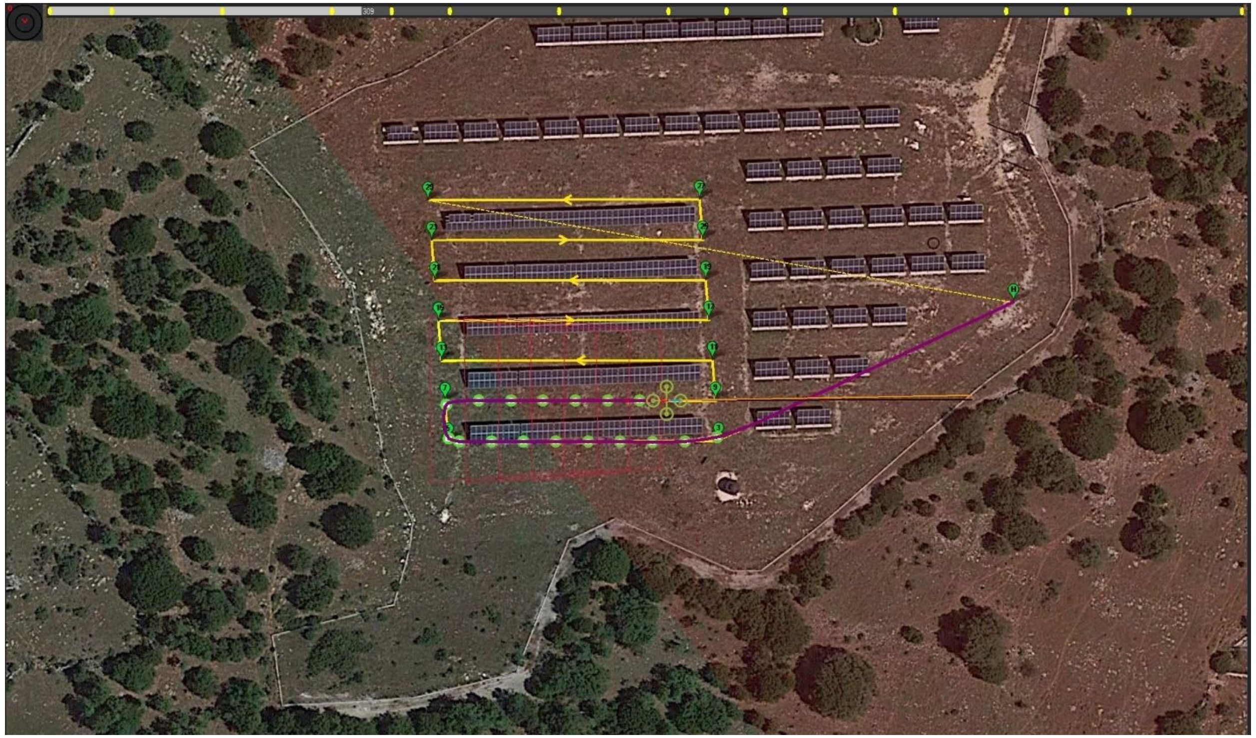


Internal Controller architecture

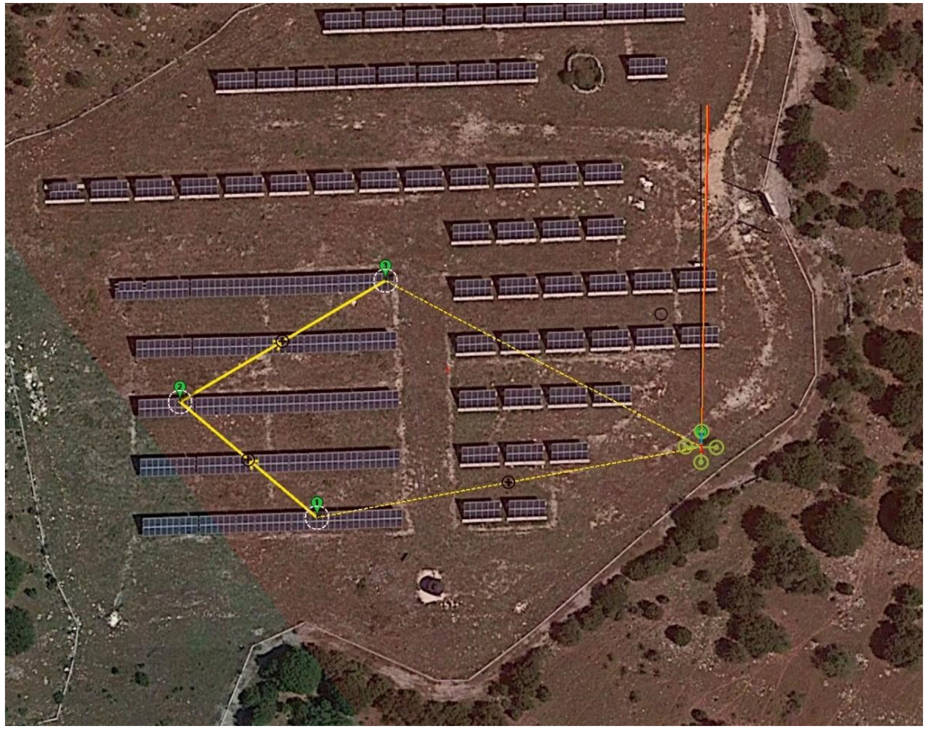


Mission plans

Full inspection



Partial/focused inspection



Multi-drone mission execution

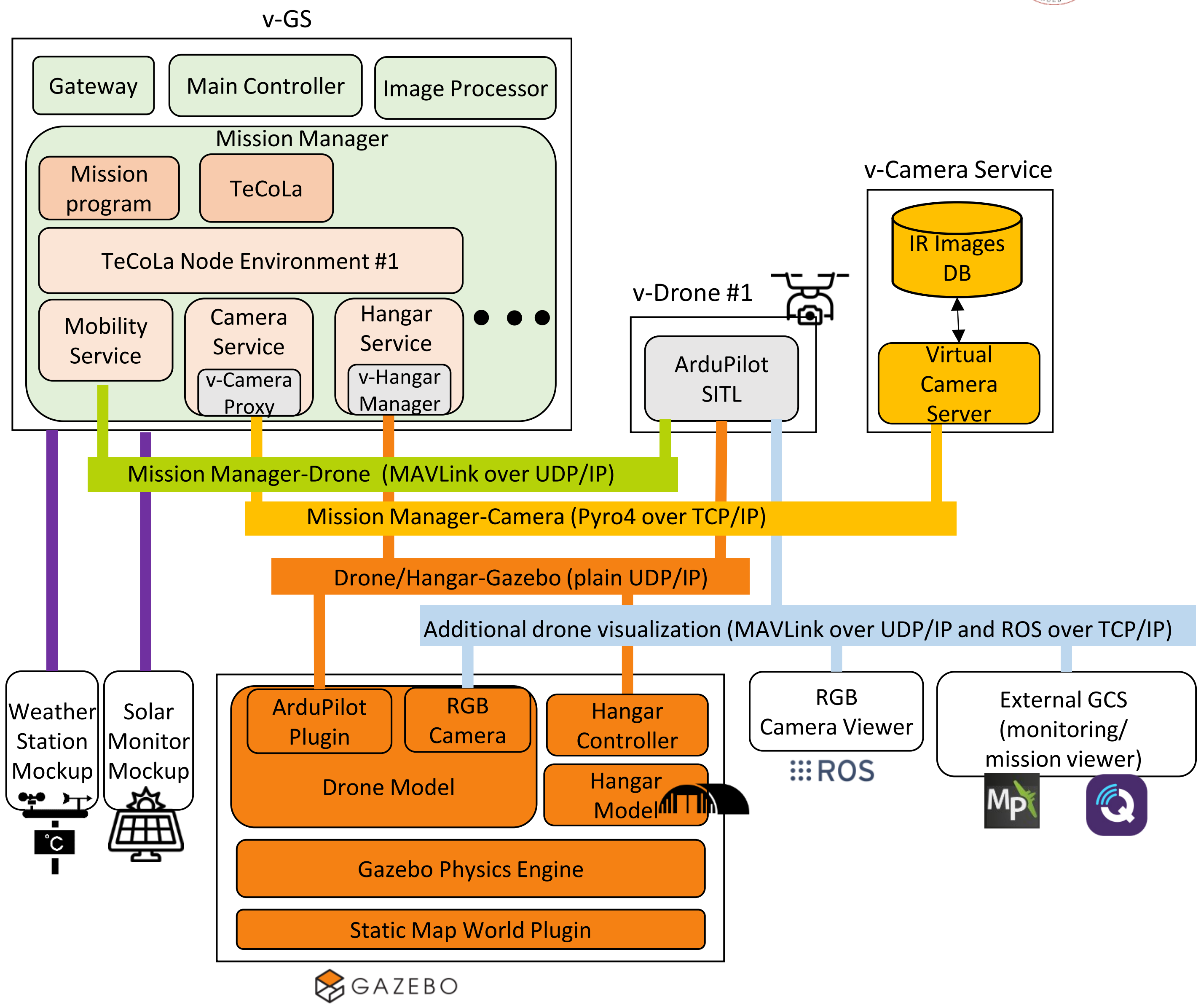
- A mission can be parallelized using multiple drones concurrently
- The mission plan is partitioned into disjoint plans, one for each drone
  - number determined based on availability and/or via a configuration parameter

- At runtime, drone may experience flight problems
- It may also run out of batteries earlier than excepted

- It will be instructed to return to land
- The rest of its plan is distributed to the remaining drones

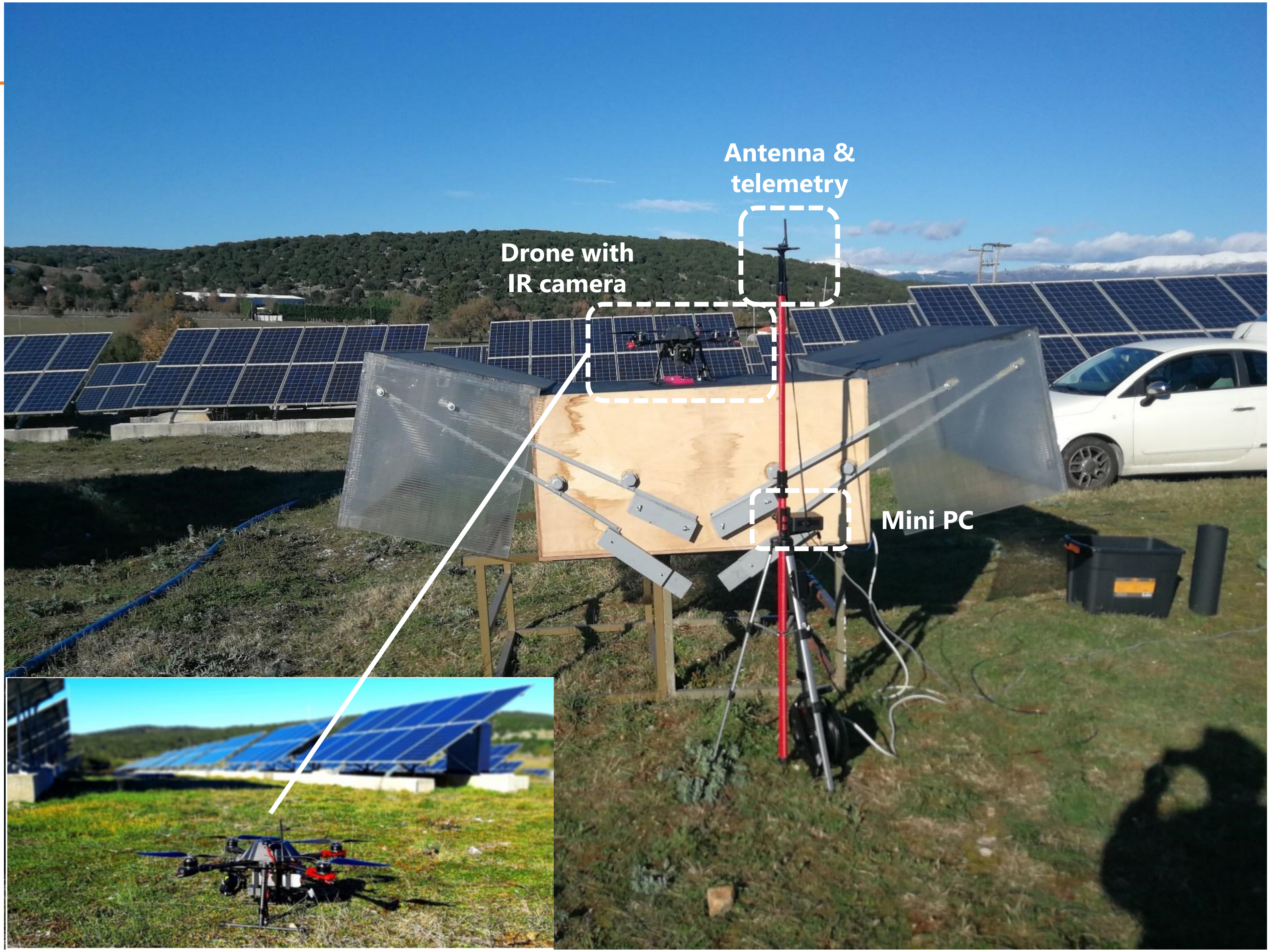
Simulation setup

- Field tests are time consuming
  - preparations, travel time
  - weather, safe perimeter
- Even more so for tests that involve multiple drones
  - stand-by pilot for each one!
- Use a SITL setup to test the SW
  - Ardupilot/SITL
  - Mission Planner
  - Gazebo
  - ROS



Field setup

- Custom hangar with minimal functionality
  - open/close hatch
  - charging plate
  - IRLock beacon
- Portable mast with
  - drone telemetry link
  - WiFi
  - embedded PC (running the entire control & image processing SW)



Full cycle of operation in the field

- Controller SW runs the full cycle
  - open hangar & arm the drone
  - instruct the drone to take off
  - instruct the drone to goto specific waypoints and take pictures
  - when done, instruct the drone to return to the home position
  - instruct the drone to land (using the precision land mode)
  - when the drone lands, start downloading and processing the images take during flight
  - close the hangar

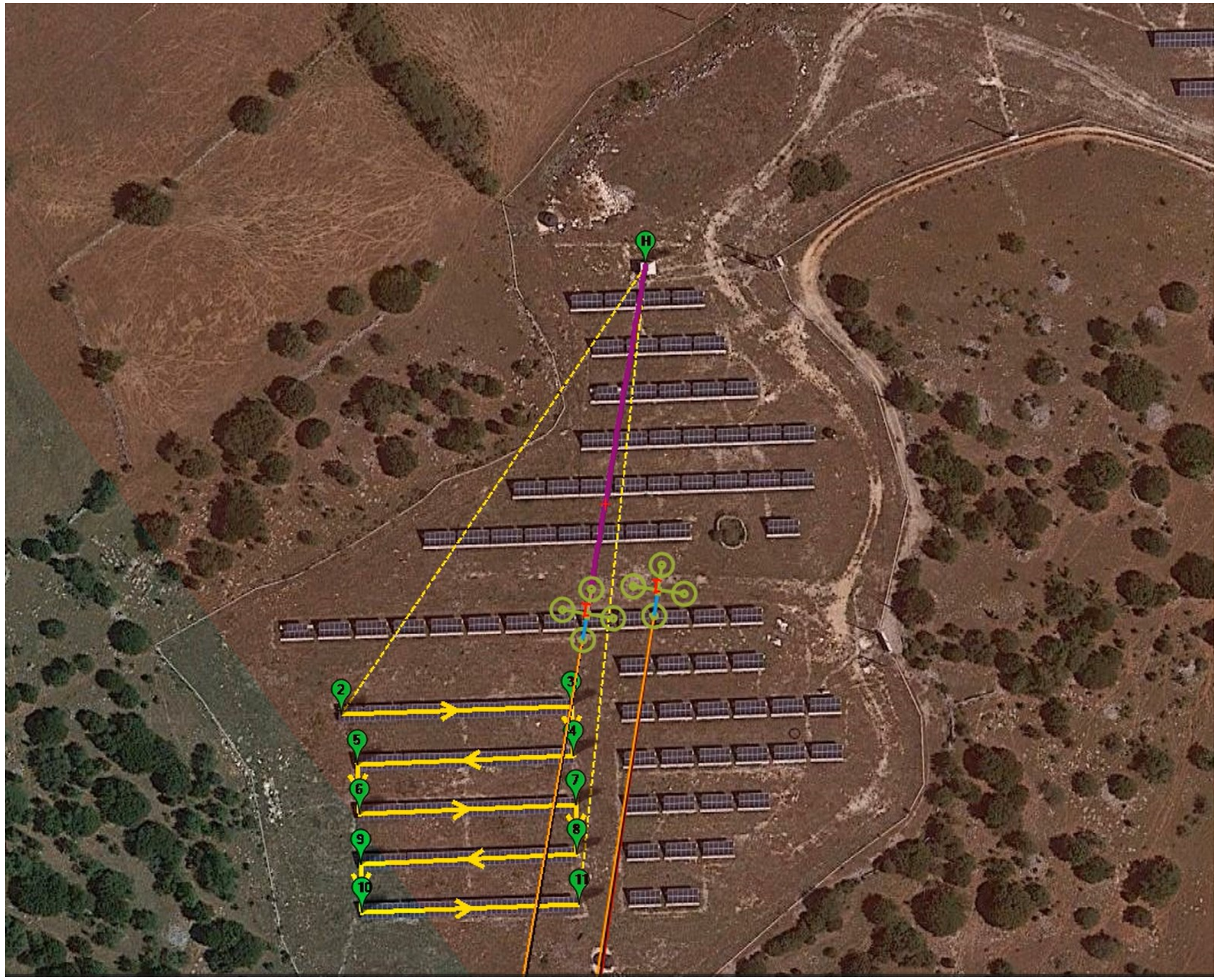
- Everything runs automatically
  - no human in the loop
  - the pilot seen in the video is merely stand-by, to takeover just in case something goes wrong – fortunately, this was not needed

SITL tests with multiple drones

- Similar scenarios as for the single drone
- Using multiple (two) drones

- Controller SW automatically splits the mission into parts, one for each drone
- Concurrently runs and monitors the full cycle of operation for each drone

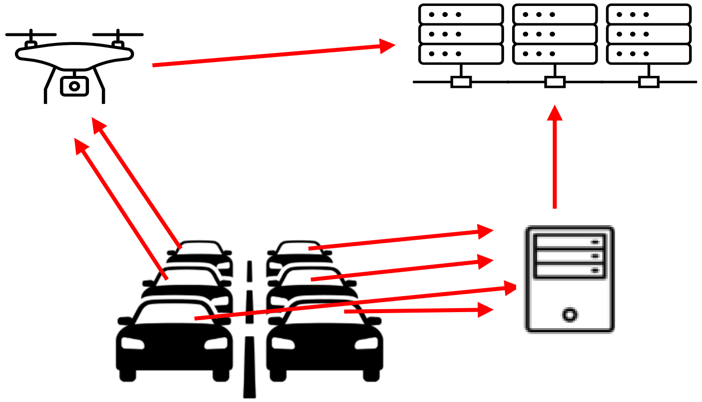
- Such scenarios have also been tested in the field – not captured in video because we did not have a third drone …



Flexible application deployment & orchestration

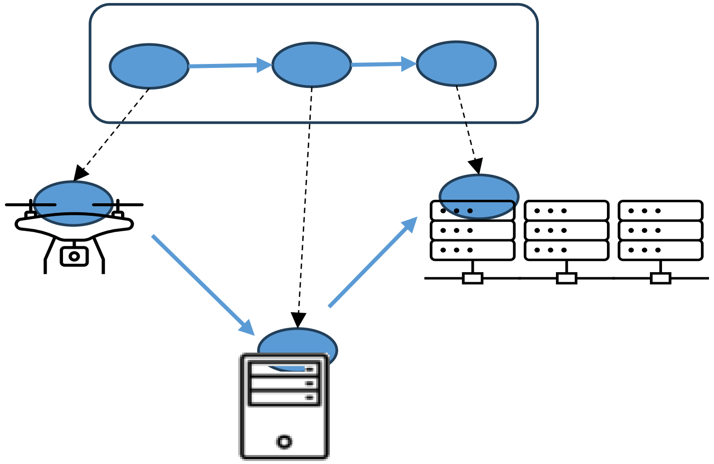
Drones as hosts for application code

- Many works focus on using drones as access points & service providers for ground vehicles
  - complementary to base-stations & edge servers



- We consider a largely orthogonal scenario

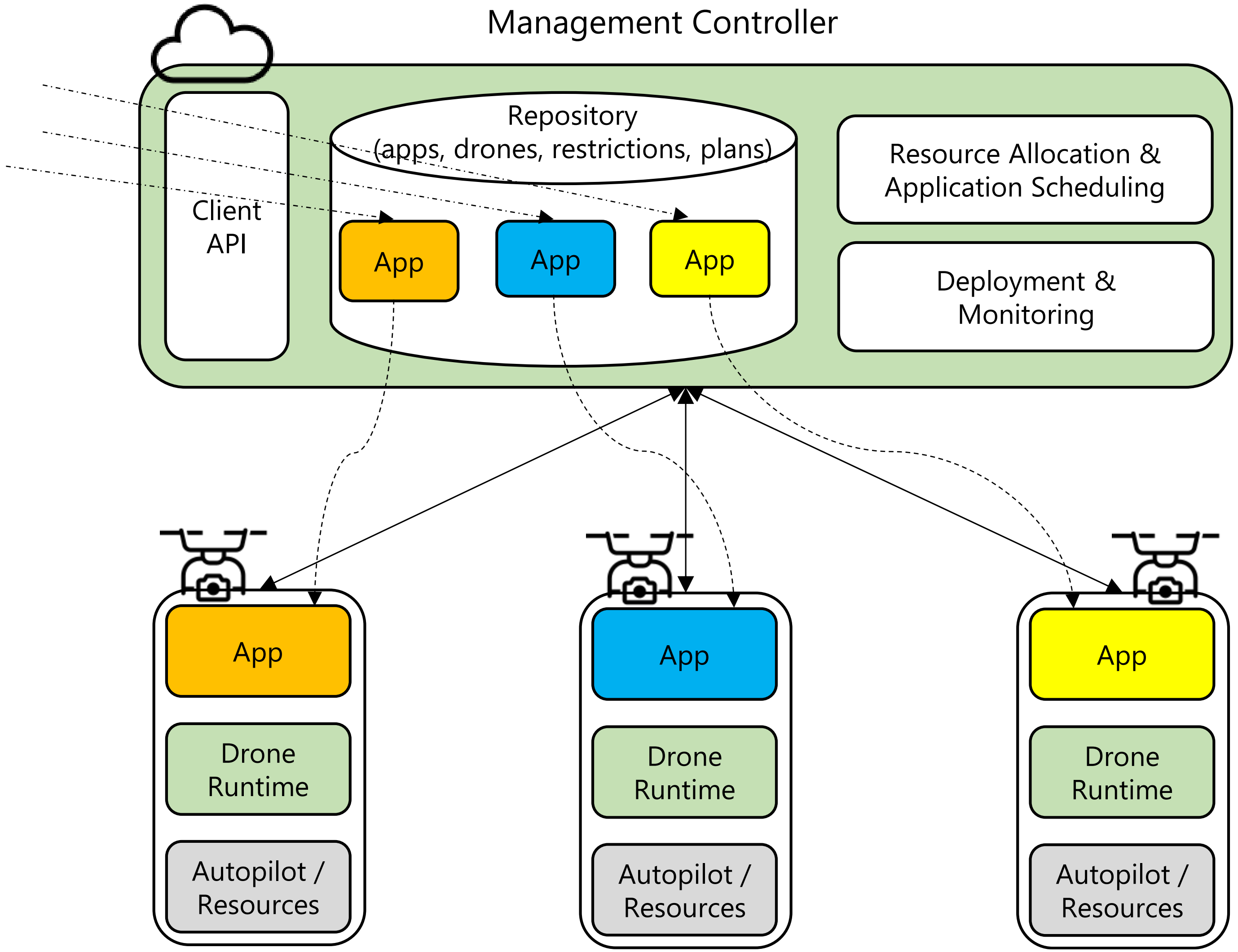
- Use drones as application hosts
  - run application-specific on-board sensing, computation, actuation tasks
- On demand
- Perhaps in conjunction with other infrastructure that can host other parts of the application, at the edge and/or in the cloud



Key aspects

- Support flexible application deployment
- Enforce flight/privacy restrictions
- Support distributed application deployment

Application deployment support



Drone & application descriptions

```
id: 123
model: custom
type: quadcopter
category: small

-- physical features
dimensions: {height=30cm,length=50cm, width=50cm}
weight: 1200g

-- flight features
autopilot: ArduCopterV3.6.11
max-speed: 15m/s
max-alt: 50m
max-time: 15min
capabilities: [hover]

-- computing & communication resources
platform: RaspberryPi3
cpu: ARMv7l
ram: 1GB
storage: 5GB
os: Raspbian
networking: [WiFi, 4G]

-- sensor resources
camera: {type=RGB, res=1920x1080, model=ModelX}
```

```
id: 456
class: surveillance
navigation-type: waypoint-based

-- flight requirements
max-speed: 10m/s
alt: 20m-30m
capabilities: [hover]

-- computing & communication requirements
cpu: ARMv7l
ram: 512MB
storage: 1GB
os: Raspbian
networking: [4G]

-- sensor requirements
camera: {type=RGB, res=1280x720}

-- configuration
waypoints
```

Service-based access control

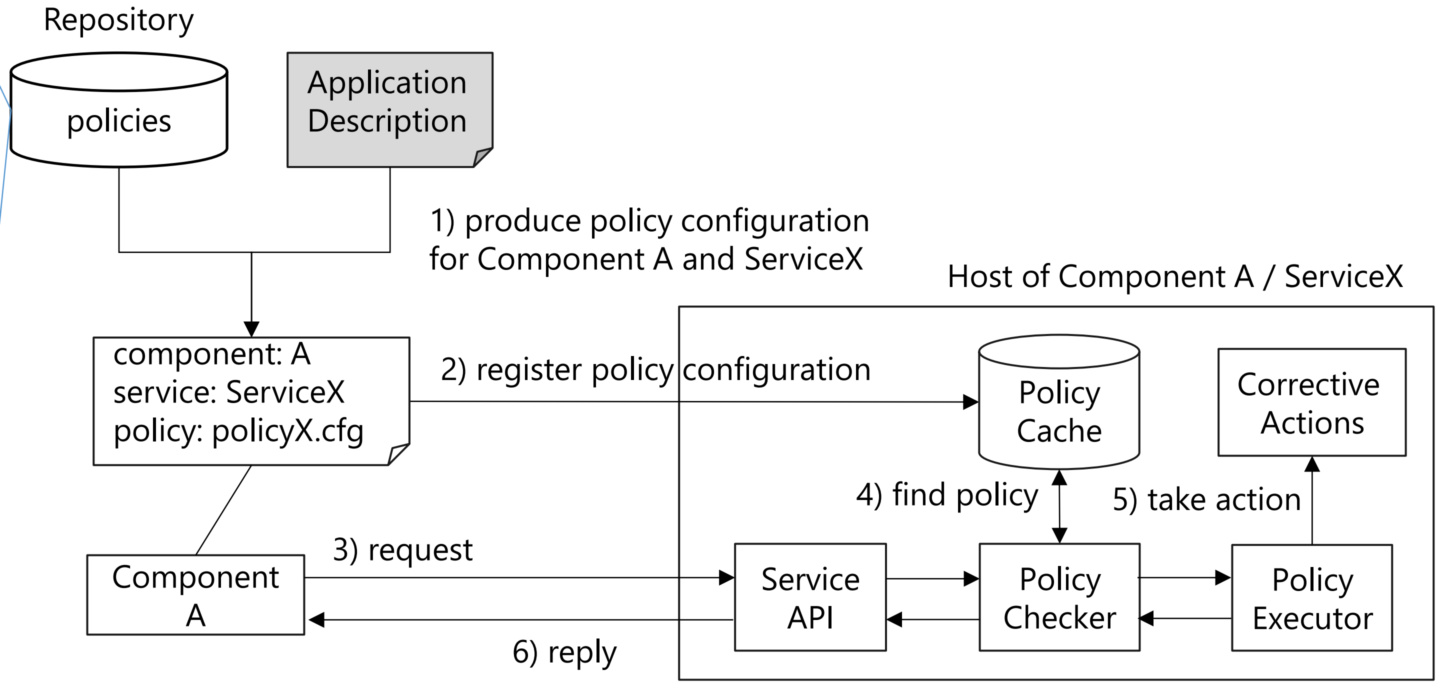
- Not all applications/drones are equal
- Need to specify & enforce the desired behavior/limits

- Capture the capabilities of a drone through services
  - mobility service, camera service, sprayer service, …
- Applications declare the service methods used

- Authorities/operators specify the desired behavior through policies
- These policies are checked & enforced through controlled service access

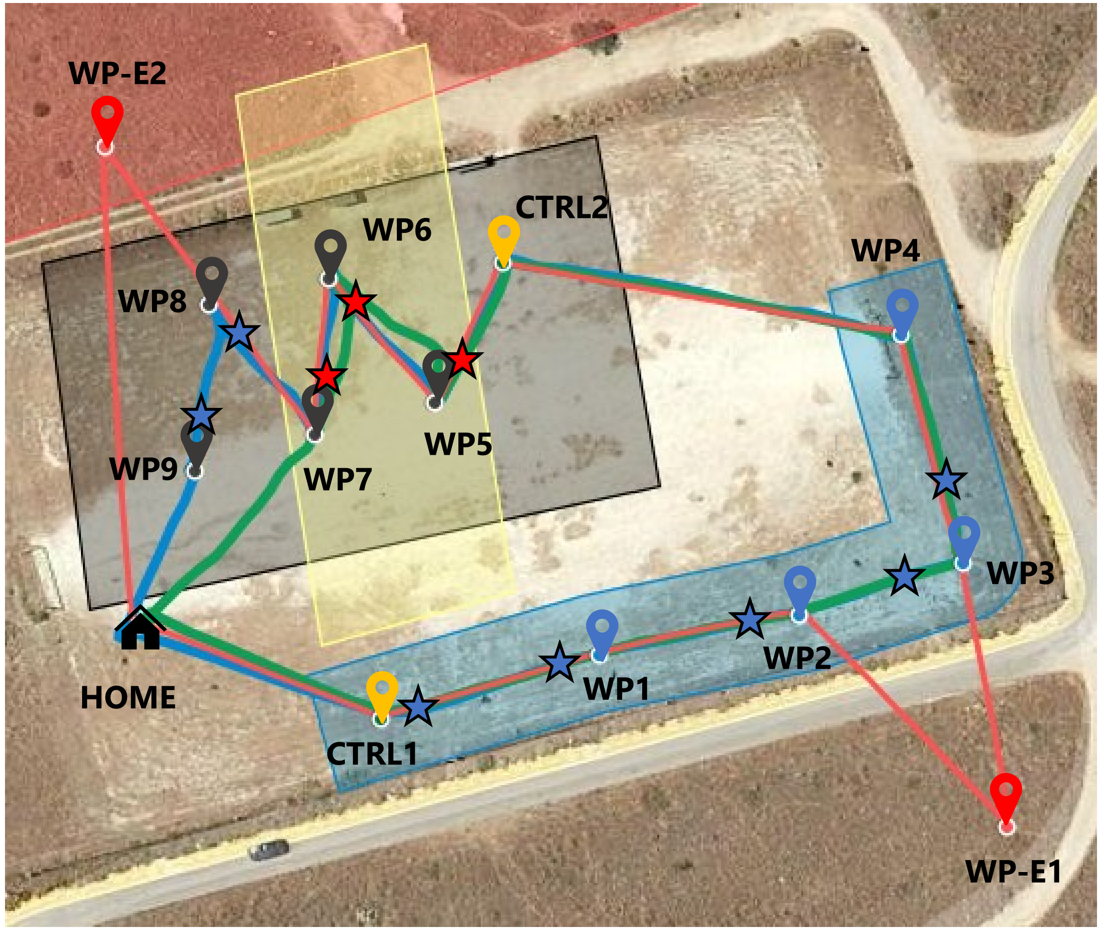
Policy-based service invocation

```
- type: AccessControl # method access
  kind: denial # forbid access
  methods: [CaptureImage, RetrieveImage]
  selector: # for all drones/edge nodes in a given area
    area: region1
- type: LimitControl # speed limit
  kind: speed
  upperLimit: 10.0
  action: enforce # adopt bound
  selector: # for all drones flying above a certain area
    area: region2
- type: LimitControl # altitude limits
  kind: altitude
  upperLimit: 50.0
  lowerLimit: 5.0
  action: enforce # adopt corresponding bound
  selector: # for a given area and certain drone classes
    area: region2
    drone: { key: class, operator: In, values: [C2, C3]}
 - type: GeofenceControl # mobility limits
   kind: exclusion # prevent from entering
   action: land # take over and land the drone
   selector: # for drones exceeding a specific weight
     area: region3
     drone: { key: mtom, operator: Gt, value: 2000}
 - type: PrivacyControl   # for sensors
   kind: camera # for the camera
   action: blurFaces # use suitable filter
   selector: # for all drones/edge nodes in a given area
     area: region3
```



Field test

- REGION1
- REGION2
- No-fly zone
- No-photo zone
- HOME
- Control point
- REGION1 wp
- REGION2 wp
- Extra wp
- Normal course
- Geofence violations – with policies
- Geofence violations – no policies
- ImageCapture success
- ImageCapture discarded



Toward distributed applications

- Monolithic applications cannot exploit the full potential (and heterogeneity) of the continuum

- Explore the concept of distributed applications
- Consisting of distinct components that can be deployed on different parts of the system infrastructure
  - in the spirit of application-specific microservices / service pipelines

- Deploy and orchestrate the execution of application components
- Across the continuum: in the cloud, at the edge, on drones

Approach

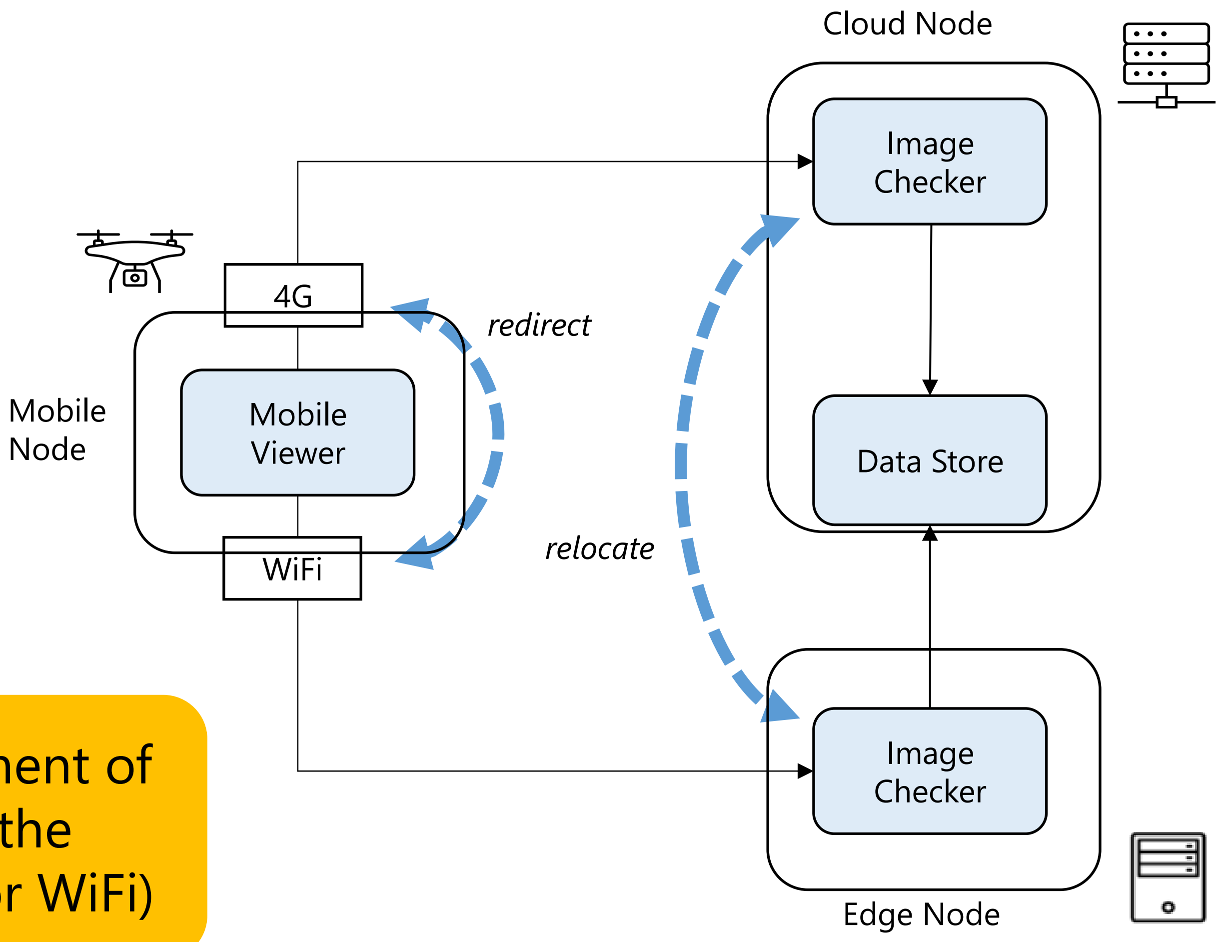
- Allow the user to express the deployment of such distributed applications in a declarative way (e.g., similar to cloud-based deployments)
- Support the deployment & orchestrated execution of such applications in the cloud-edge-drone continuum in a transparent way

- Support the relocation of application components
- Support the redirection of application traffic
- Make it easy to experiment with different deployment & configuration policies, or even change them at runtime

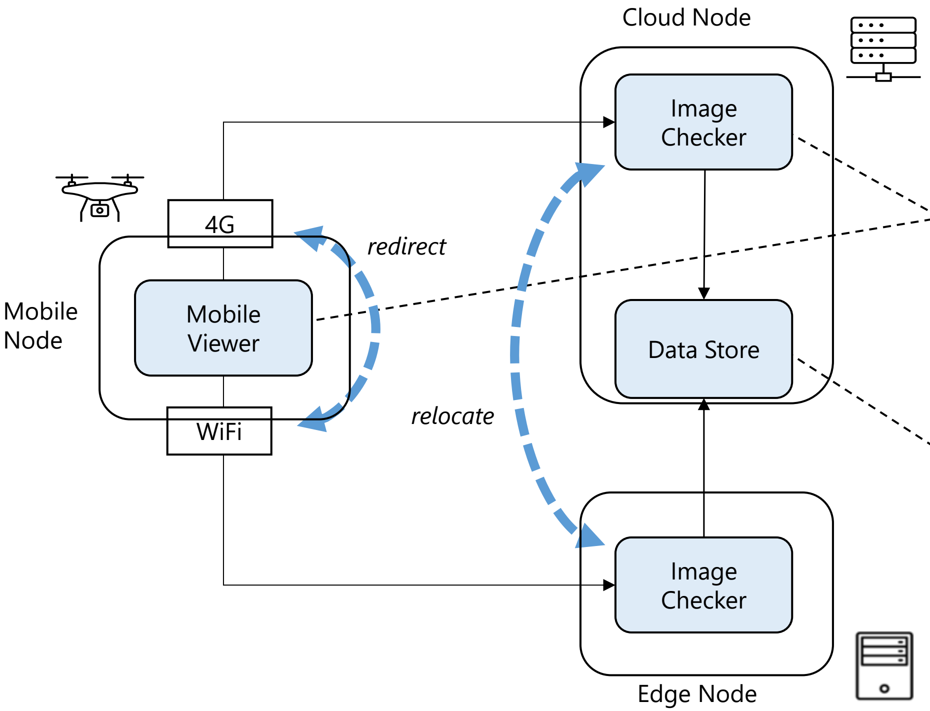
Application example

- MobileViewer
  - takes photos via the drone's camera & possibly preprocesses them
  - may also include mission logic, controlling the drone's autopilot
- ImageChecker
  - performs heavyweight computation on images to detect objects of interest
- DataStore
  - stores images of interest

Objective: Support (i) the flexible deployment of the ImageChecker (cloud or edge) and (ii) the redirection of application traffic (over 4G or WiFi)



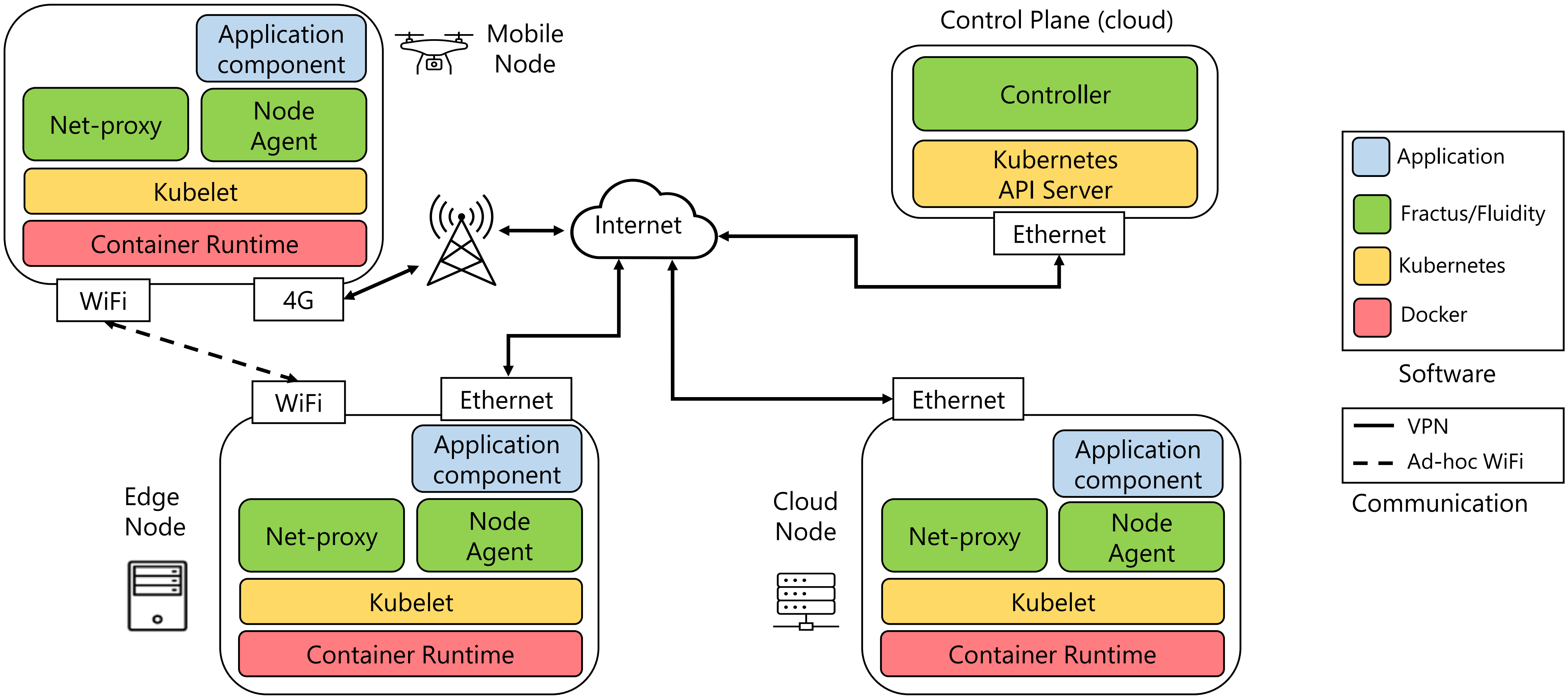
Application description



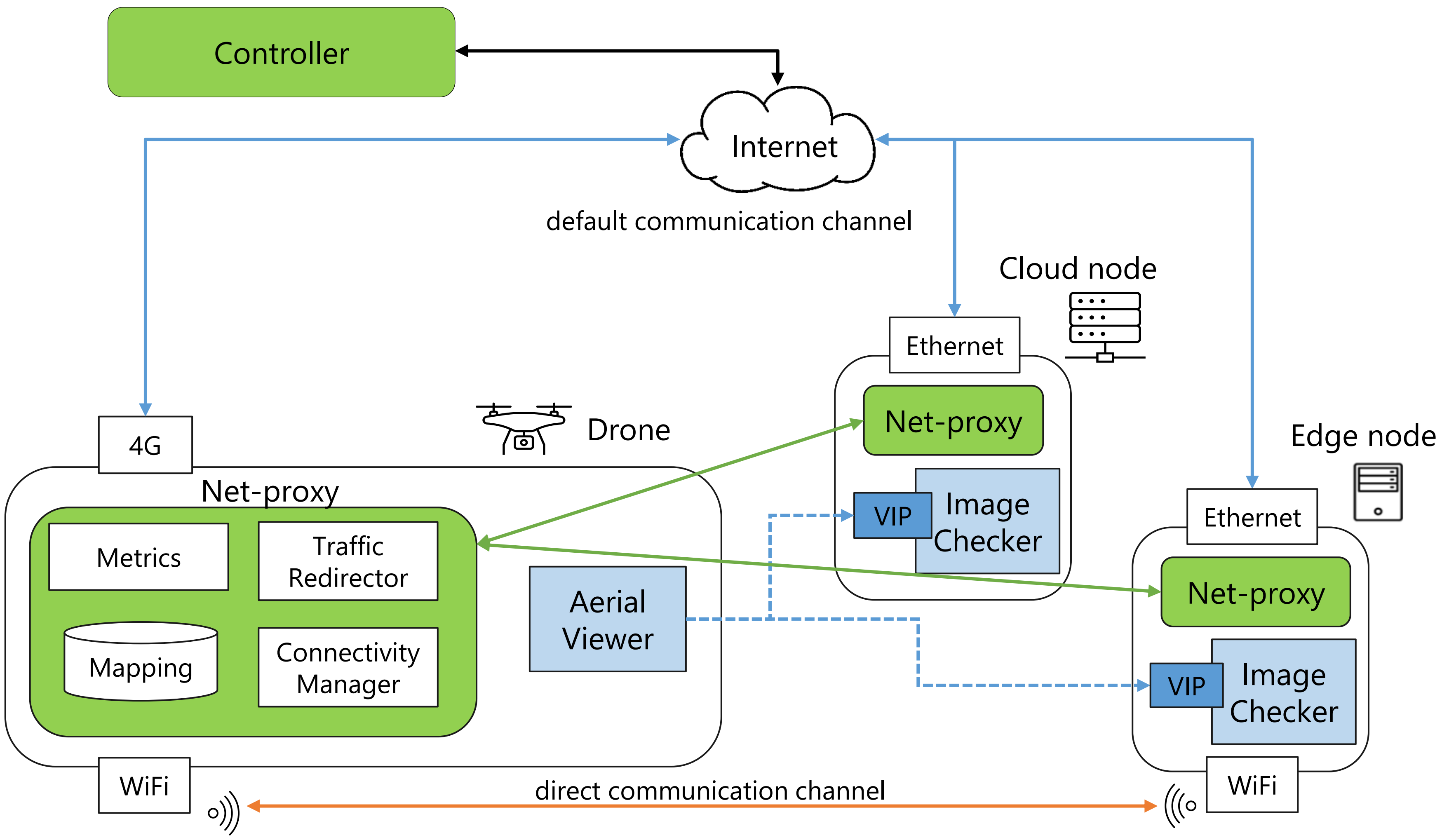

```yaml
kind: Application
name: ObjectDetectionApp
components:
- name: MobileViewer
  podSpec: PodMobileViewer.yaml
  placement: mobile
  systemServices:
    camera:
      methods: [CaptureImage, RetrieveImage]
      sensorType: RGB
  egress: [ImageChecker]
- name: ImageChecker
  podSpec: PodImageChecker.yaml
  placement: hybrid
  ingress: [MobileViewer]
  egress: [DataStore]
- name: DataStore
  podSpec: PodDataStore.yaml
  placement: cloud
  ingress: [ImageChecker]
Policy:
- name: simple_policy.py
```

place in the cloud or at the edge

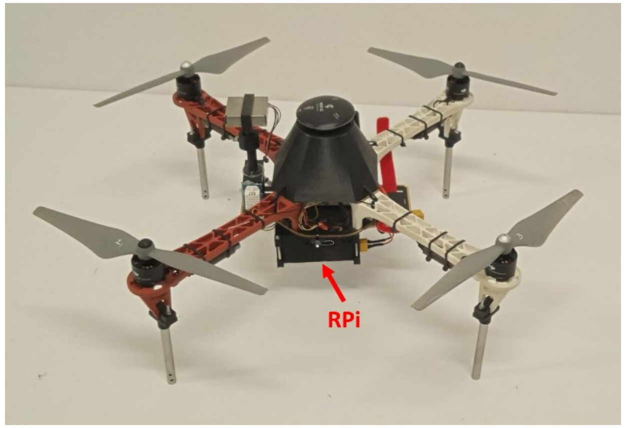
Architecture & test setup

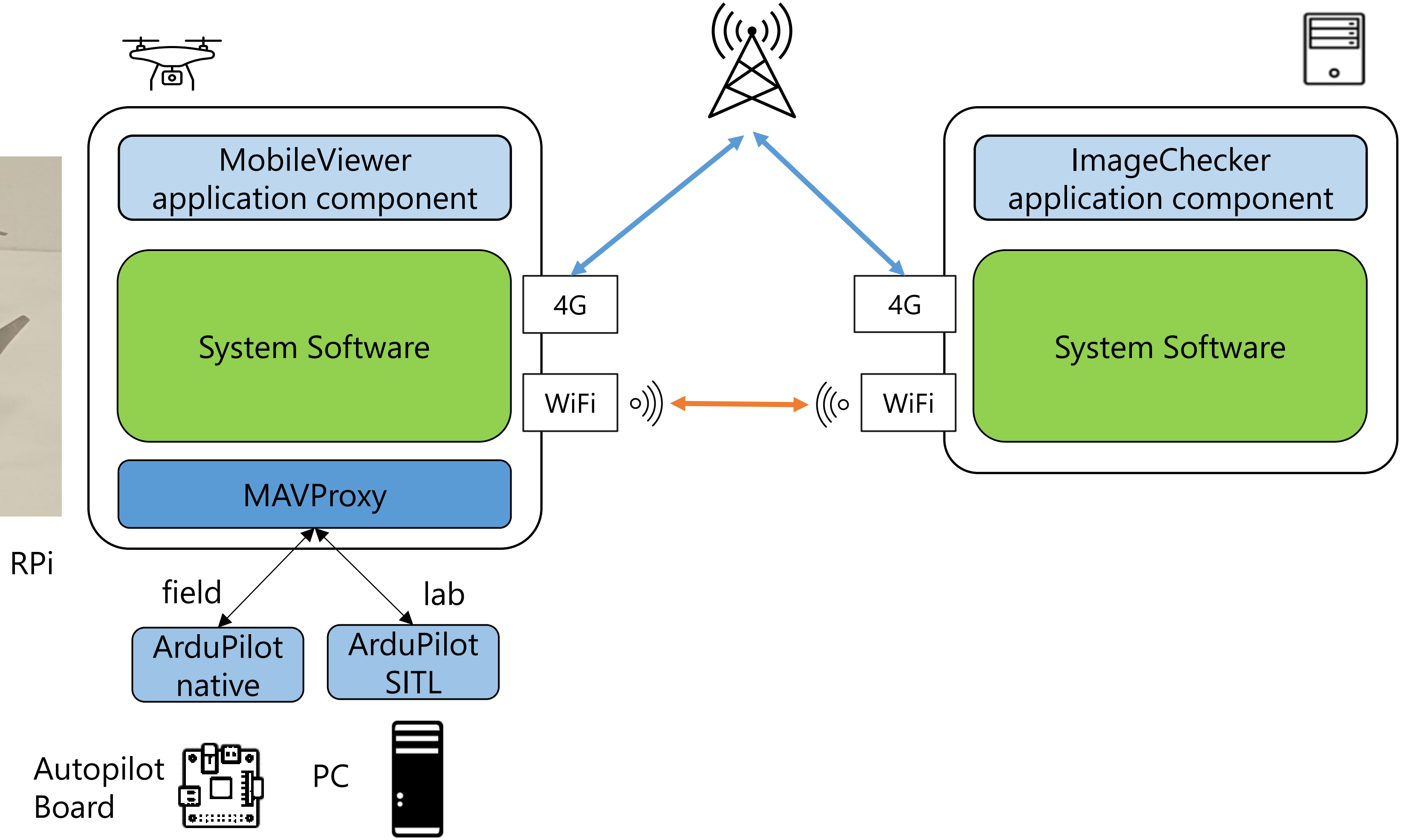


Application traffic redirection

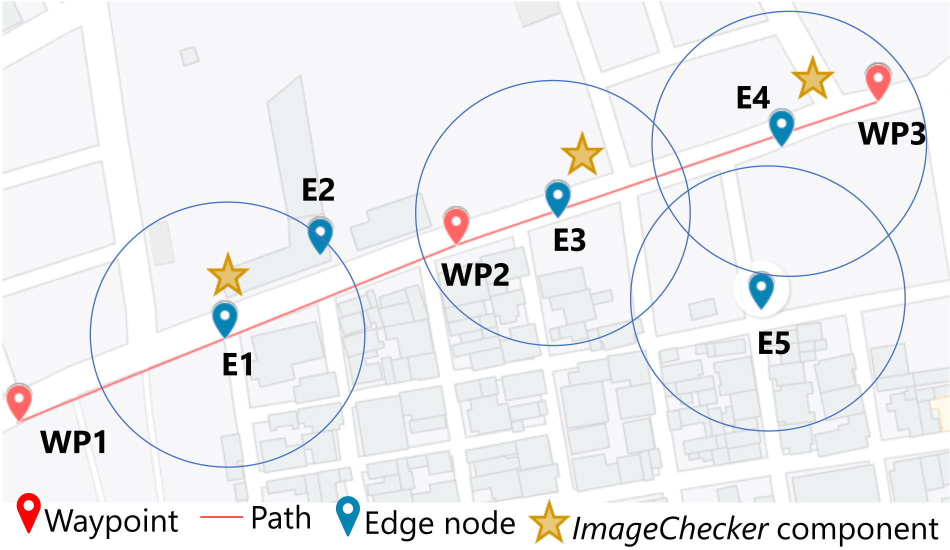


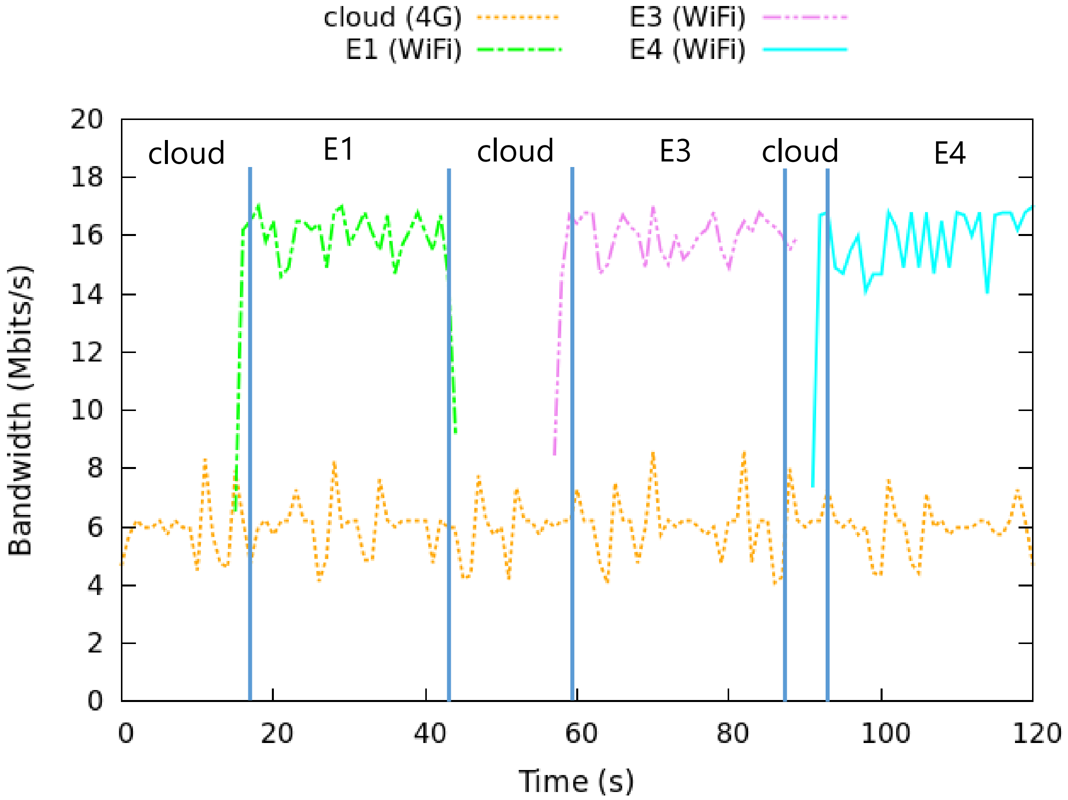
Experimental node setup





Results


E4
WP3
E2
E3
WP2
E1
E5
WP1
Waypoint
Path
Edge node
ImageChecker component


cloud (4G)
E1 (WiFi)
E3 (WiFi)
E4 (WiFi)
cloud
E1
cloud
E3
cloud
E4
Bandwidth (Mbits/s)
Time (s)
0
2
4
6
8
10
12
14
16
18
20
0
20
40
60
80
100
120

Separation between mechanisms and policy

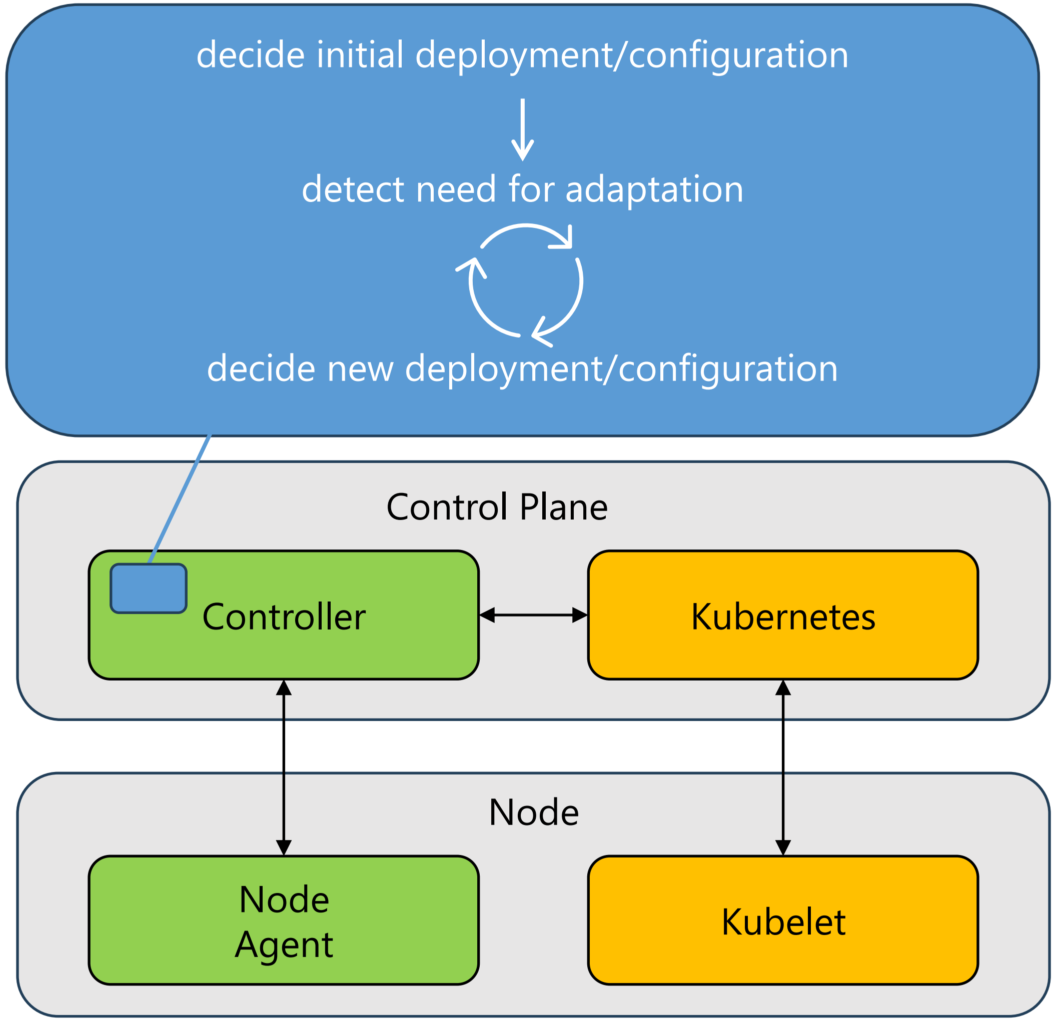
- Various configuration choices
  - Where to deploy (hybrid) application components?
  - Over which network interface / link to redirect application traffic?

- Can adopt different decision approaches
- Can have different objectives / optimization targets

- Make it easy to experiment with different policies

Pluggable policies

- The application deployment/configuration policy is a plug-in
- Interacts with the rest of the framework (Controller) via a well-defined API
- Can access/process arbitrary telemetry and maintain its own state (data)
- Produces deployment plans
  - for execution by the Controller
- The plan can be adapted at runtime
- The policy itself can be changed at runtime, while the application is running



Different policies for component relocation

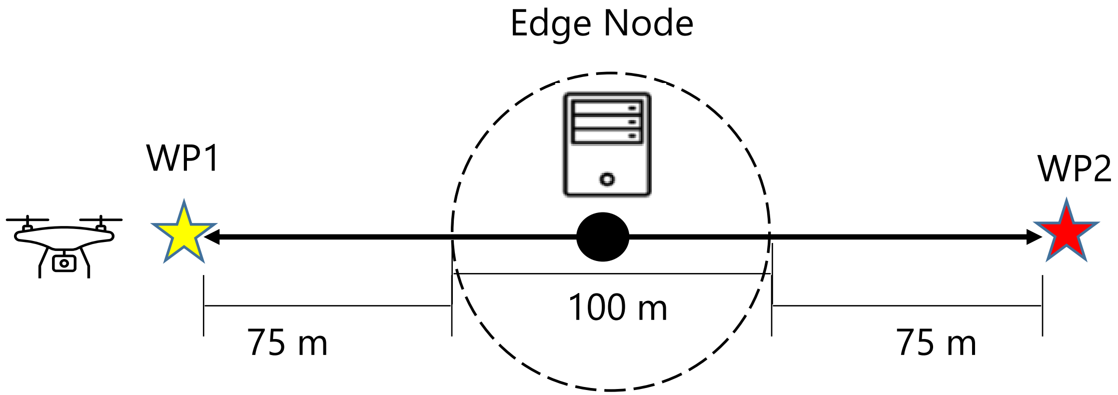
Naïve (blind)

- Place the ImageChecker application component by default in the cloud
- Relocate ImageChecker to the edge when the drone enters the range of the edge node
- Relocate the ImageChecker back to the cloud when the drone exits the range of the edge node

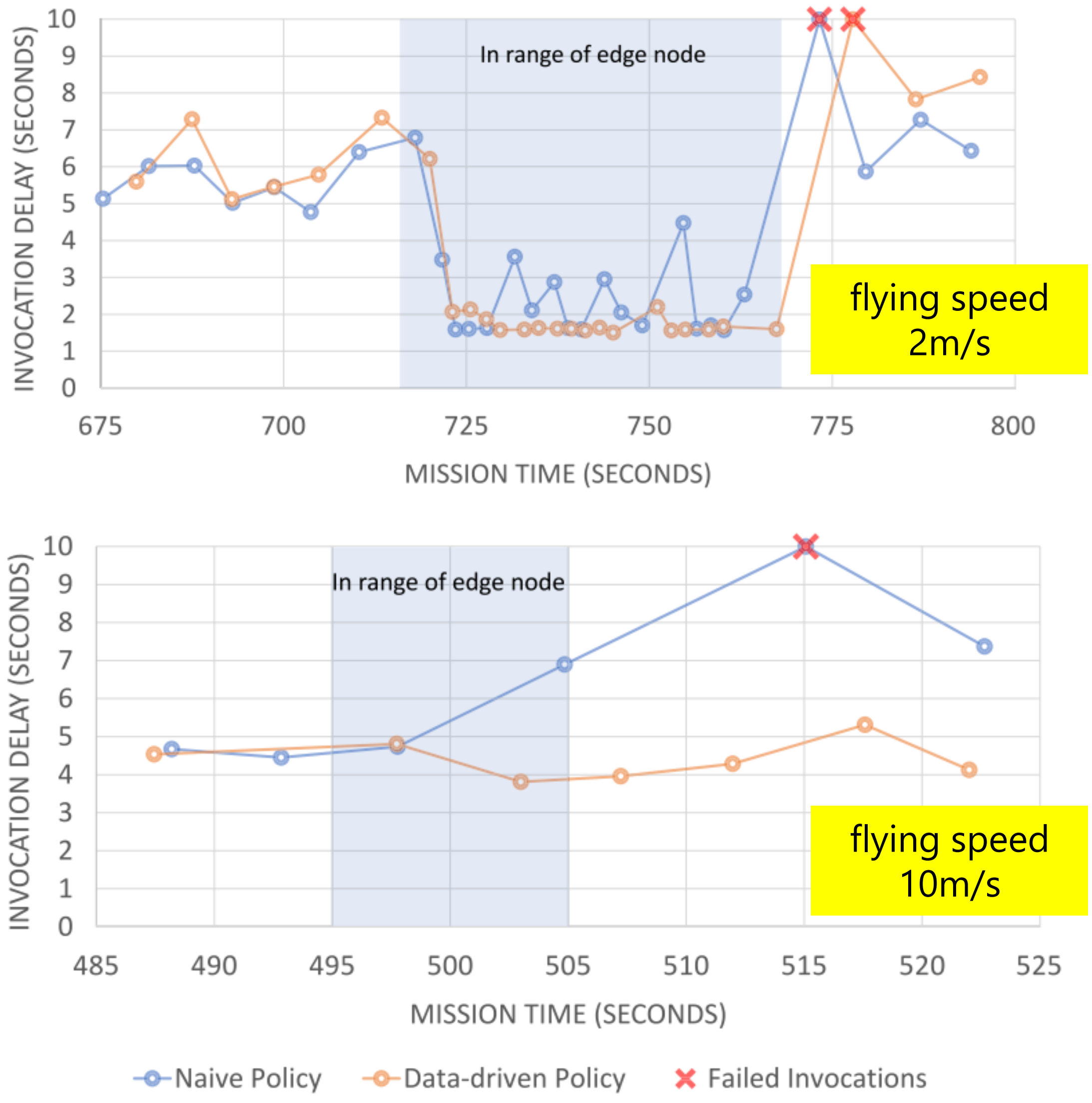
Data-driven (learning)

- Place the ImageChecker application component by default in the cloud
- Initially adopt the naïve policy
- Record the relocation delays and invocation time for cloud and edge
- Relocate ImageChecker only when the edge invocation times are **expected** to outweigh the relocation overhead

Test scenario & results



Phase number	Round-trips	Velocity (m/sec)
#1	1	2
#2	5	10
#3	1	2



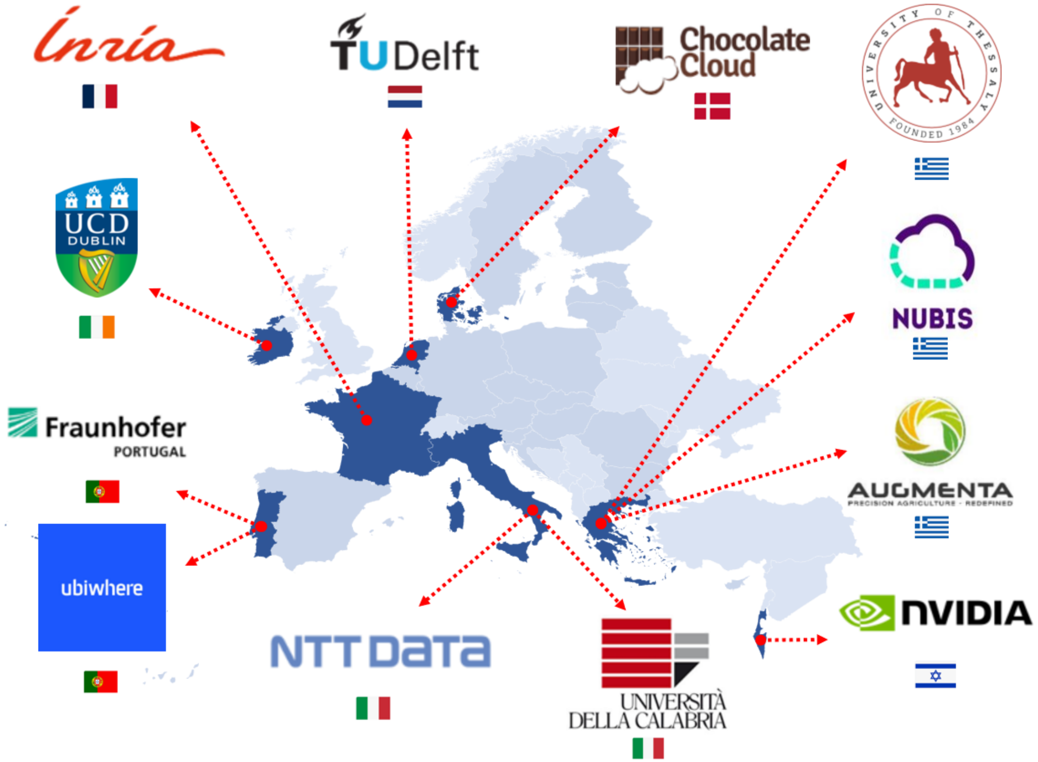
MLSysOps project

Project focus

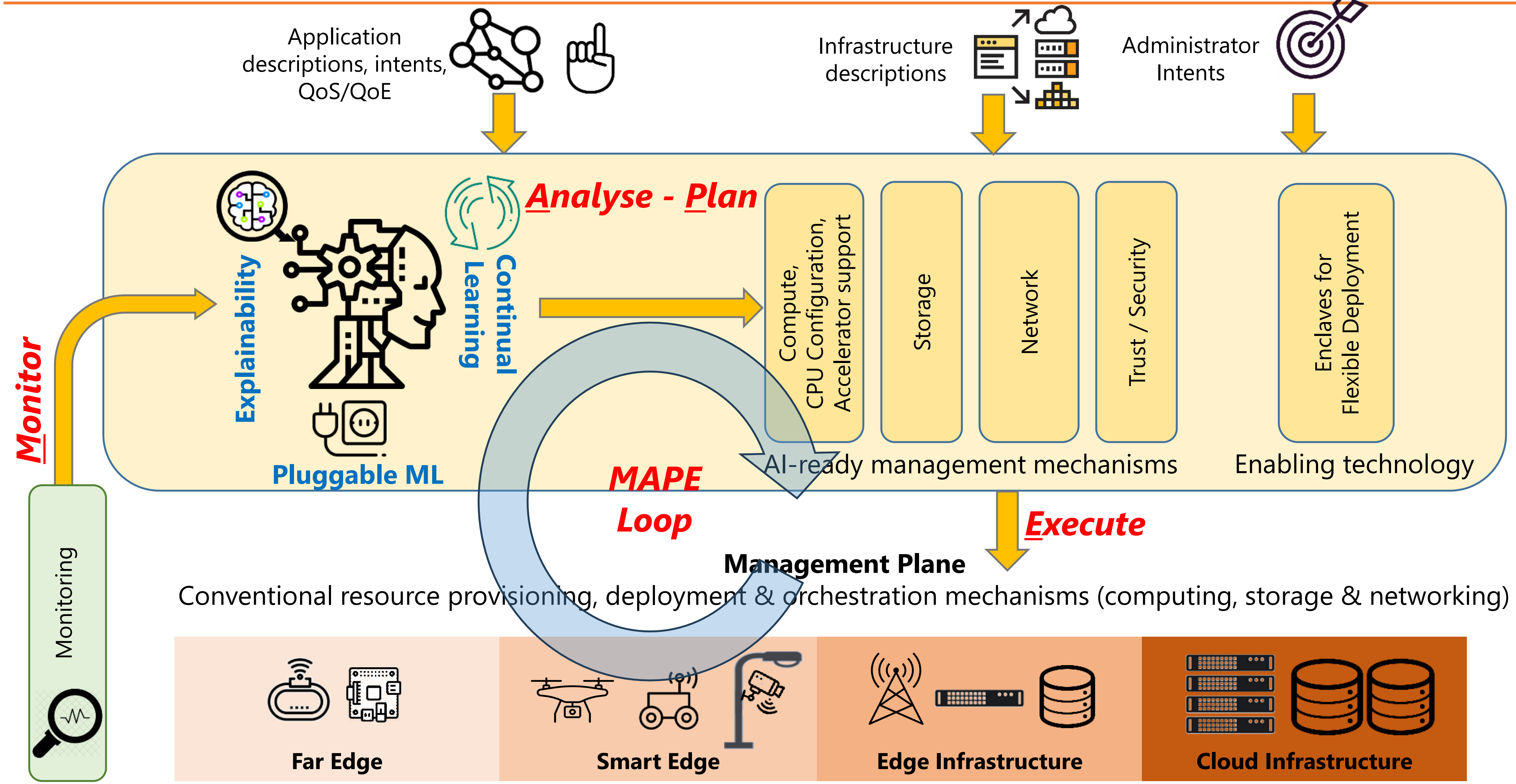
Autonomic system management and configuration
*in the **cloud-edge-IoT continuum** using **AI/ML methods***

MLSysOps

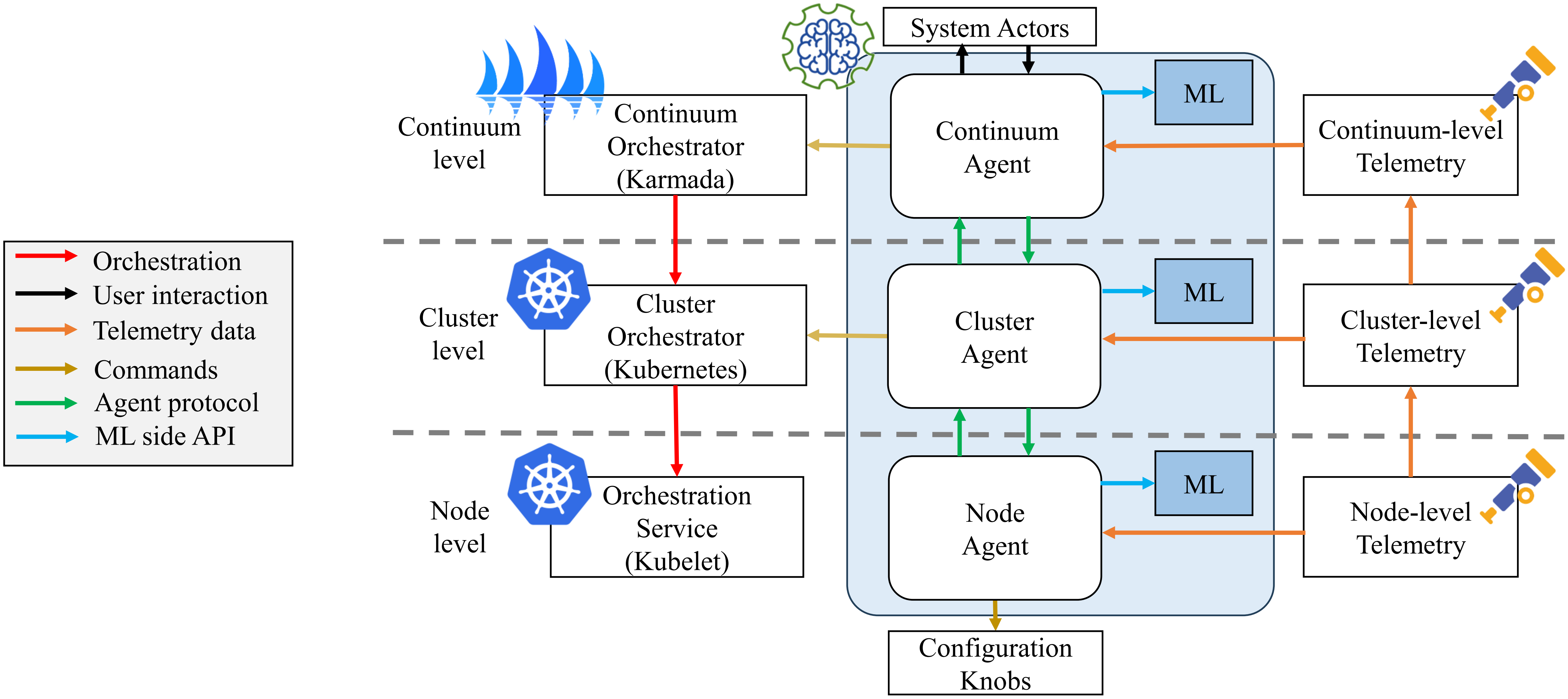
- Modular, distributed applications
  - different (interacting) components
- Explore different management aspects
  - deployment, computing/acceleration, storage, communication/networking, trust
- Disassociate management from control
  - AI/ML-ready (policy-neutral) mechanisms
  - take decisions using suitable ML models
- Key AI/ML features
  - distributed / hierarchical agent-based approach
  - different agent may use different ML models
  - continual learning, efficient retraining, explainable ML



MLSysOps concept

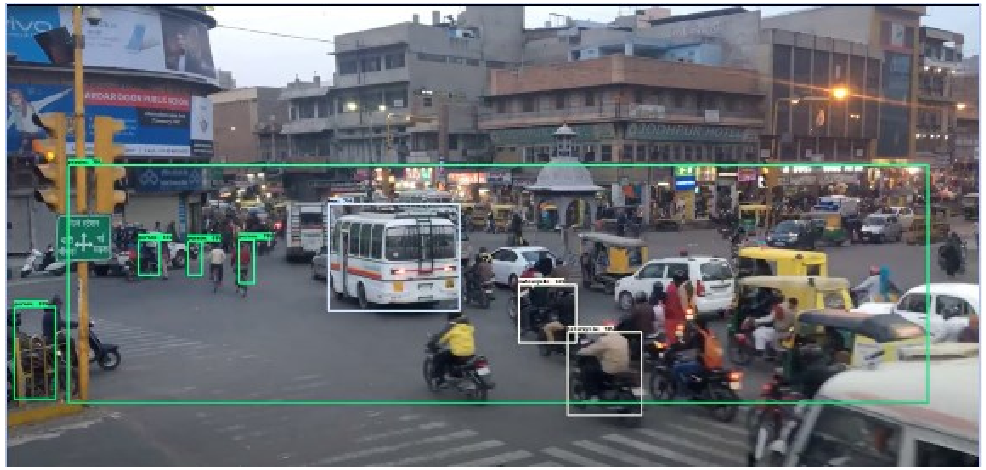


MLSysOps architecture



Edge-oriented application use cases

Smart City

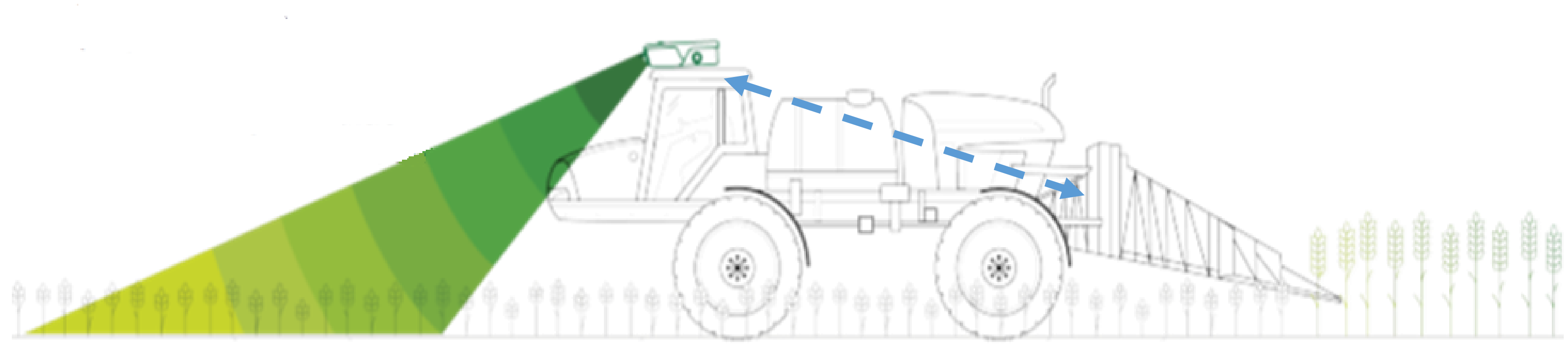


Use AI/ML-driven control to manage/configure application modules that can be deployed on smart lampposts for object/incidence detection

Smart Agriculture

Use AI/ML-driven control to jontly manage/configure the image processing application pipeline in camera-based devices mounted on a tractor and a companion drone.

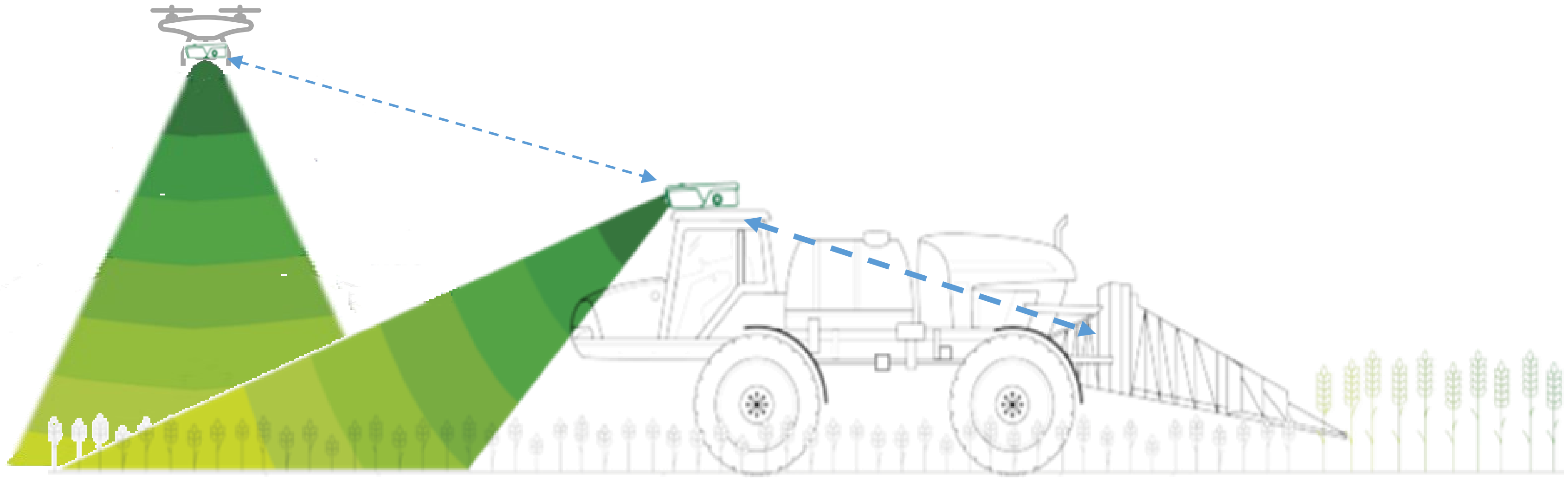
Tractor operation



- On-tractor device is used to detect weeds in the field
- This information is used to control the sprayer at the back of the tractor
- Significant savings in herbicides and reduced soil pollution

- However, image processing may face issues due to shadows, glares, bumps, etc.

Tractor operation with drone

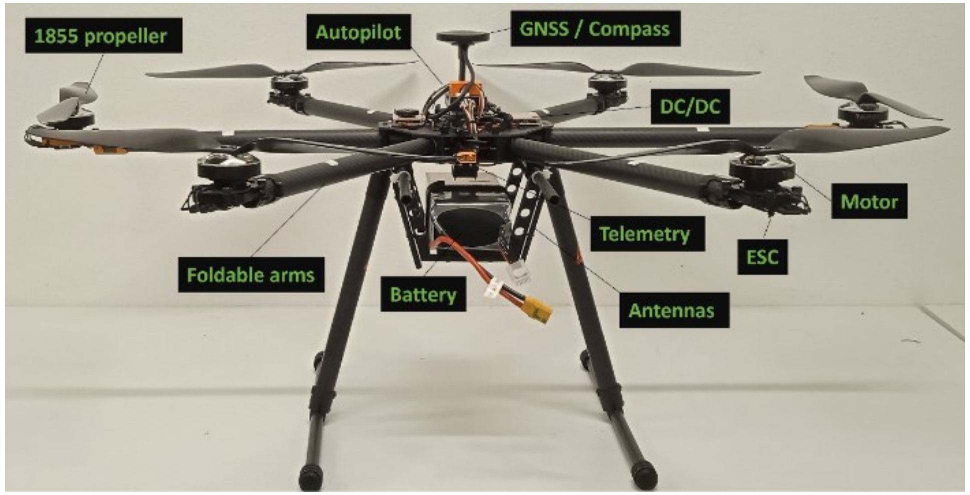


- Use a companion drone to aid the tractor featuring a similar device
- Vertical view may resolve some of the previous issues

- However, the drone must be engaged in a smart way
- Only if the tractor is not doing well and this is expected to continue “sufficiently long”

The nodes

Tractor Node



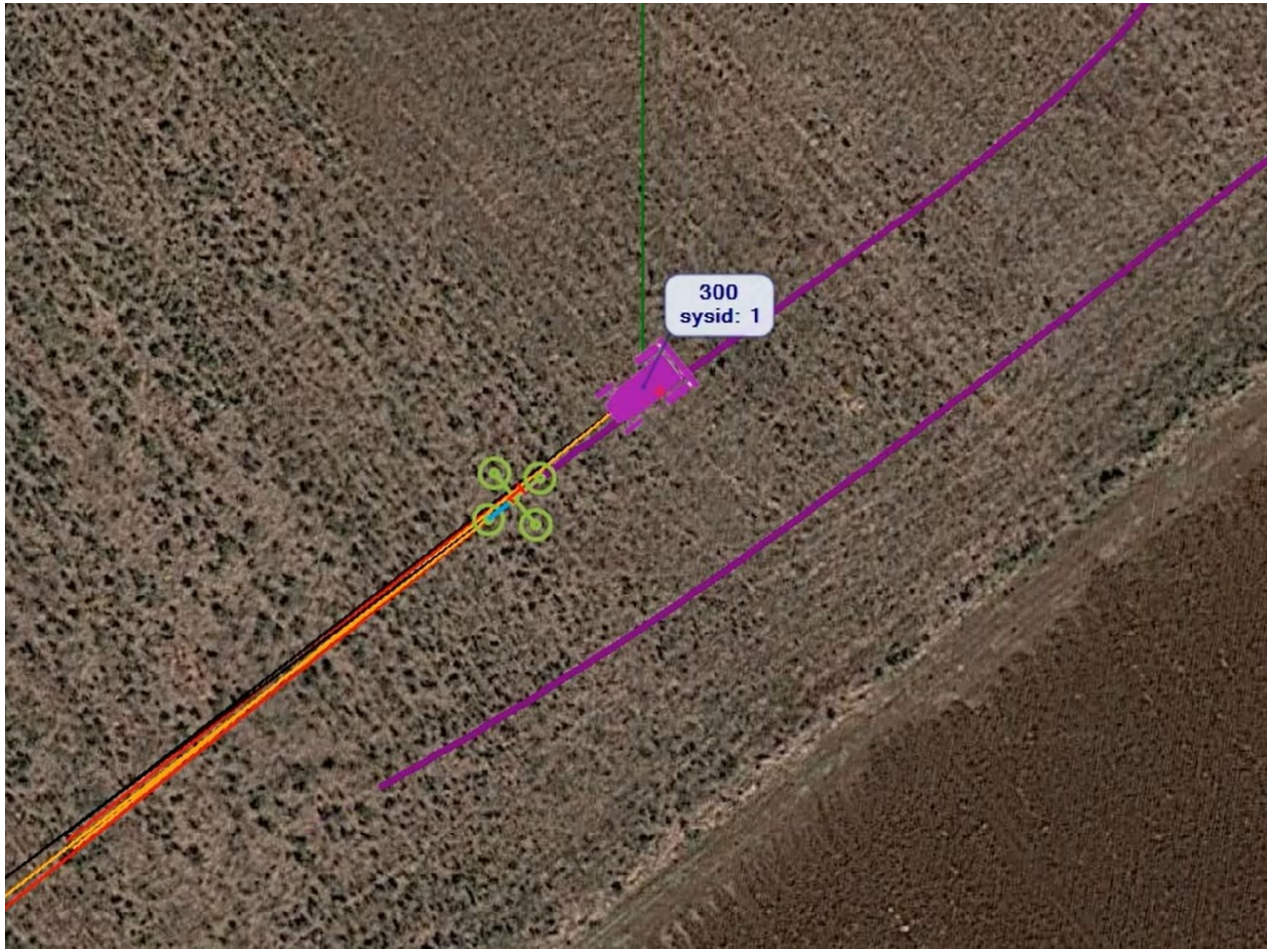
Drone

SITL tests

- Tractor and drone as virtual vehicles
- Each vehicle runs a separate SITL autopilot configuration

- Tractor starts scanning the field
- Drone is engaged/disengaged to start/stop following the tractor

- For testing purposes, drone (dis)engagement is triggered manually via suitable commands
- Ultimately, these decisions will be taken automatically via AI/ML



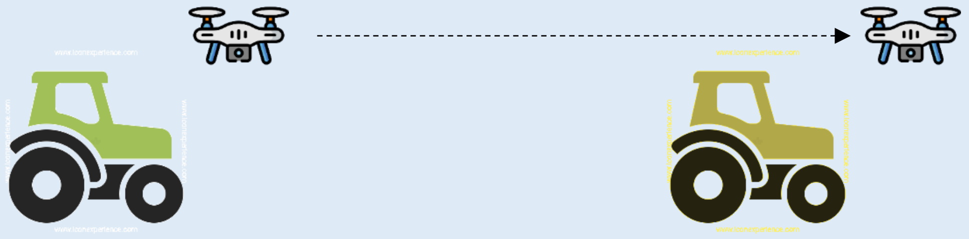
Field tests

- Similar scenarios as in SITL tests
- Using a real tractor and drone

- Confirm basic drone operation without human intervention under real conditions
- Standalone and tandem operation of the image processing pipeline
- Collect real data for system and application performance

Possible extensions/variations

- Same drone supports multiple tractors
  - in an alternating way



- Several drones used in parallel
  - each supporting a different tractor

- Several drones support a single tractor
  - in relay

Wrap-up

Summary

Motivation

- Use drones as first-class resources in next-generation applications

Work presented

- Automation of the full operation cycle, including landing/housing/charging
- Flexible application deployment/orchestration in conjunction with the edge/cloud
- Manage, check & enforce flight and sensing/actuation restrictions

Testing, testing, testing

- Field tests with drones are very time-consuming
- Multi-drone tests are an even bigger challenge
- Using a suitable SITL/HITL environment is of key importance

Further directions

Hangars as a shared resource

- Suitable planning, scheduling, reservations, etc

Drone multi-tenancy

- SW-wise not such a big issue
- Must support sharing also in the flight/mobility dimension
- Need appropriate conventions (e.g., driver-passenger relationship)

Coordinated application deployment/execution on multiple vehicles

- Swarm-based applications (UAV + UAV + … UAV)
- Heterogeneous unmanned systems (UAV + UGV, UAV + USV)

Related publications

- F. Pournaropoulos, A. Patras, C. D. Antonopoulos, N. Bellas, S. Lalis, "***Fluidity: Providing Flexible Deployment and Adaptation Policy Experimentation for Serverless and Distributed Applications Spanning Cloud–Edge–Mobile Environments***", *Future Generation Computer Systems*, vol. 157, 2024.
- S.-F. Pournaropoulos, C. D. Antonopoulos and S. Lalis, "***Supporting the Adaptive Deployment of Modular Applications in Cloud-Edge-Mobile Systems***", *International Conference on Embedded Wireless Systems and Networks (EWSN)*, September 2023.
- N. Grigoropoulos and S. Lalis, "***Fractus: Orchestration of Distributed Applications in the Drone-Edge-Cloud Continuum***", *IEEE Computers, Software, and Applications Conference (COMPSAC)*, June 2022.
- M. Koutsoubelias, N. Grigoropoulos, G. Polychronis, G. Badakis and S. Lalis, "***System Architecture for Autonomous Drone-based Remote Sensing***", *International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous)*, November 2021.
- G. Badakis, M. Koutsoubelias and S. Lalis, "***Robust Precision Landing for Autonomous Drones Combining Vision-based and Infrared Sensors***", *IEEE Sensors Applications Symposium (SAS)*, August 2021.
- N. Grigoropoulos and S. Lalis, "***Simulation and Digital Twin Support for Managed Drone Applications***", *IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, September 2020.
- N. Grigoropoulos and S. Lalis, "***Flexible Deployment and Enforcement of Flight and Privacy Restrictions for Drone Applications***", *International Workshop on Safety and Security of Intelligent Vehicles (SSIV), IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2020.

Publications on planning/scheduling & FT

- G. Polychronis, M. Koutsoubelias, and S. Lalis, "***Should I Stay or Should I Go: A Learning Approach for Drone-based Sensing Applications***", *Workshop on Wireless Sensors & Drones in IoT (Wi-DroIT)*, *International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, April 2024.
- G. Polychronis and S. Lalis, "***Flexible Computation Offloading at the Edge for Autonomous Drones with Uncertain Flight Times***", *International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, June 2023.
- G. Polychronis and S. Lalis, "***Joint Edge Resource Allocation and Path Planning for Drones with Energy Constraints***", *International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous)*, November 2022.
- G. Polychronis and S. Lalis, "***Planning Computation Offloading on Shared Edge Infrastructure for Multiple Drones***", *International Workshop on Wireless Sensor, Robot and UAV Networks (WiSARN)*, *IEEE International Conference on Distributed Computing Systems (ICDCS)*, July 2022.
- G. Polychronis and S. Lalis, "***Safe Optimistic Path Planning for Autonomous Drones under Dynamic Energy Costs***", *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, September 2021.
- T. Kasidakis, G. Polychronis, M. Koutsoubelias and S. Lalis, "***Reducing the Mission Time of Drone Applications through Location-Aware Edge Computing***", *IEEE International Conference on Fog and Edge Computing (ICFEC)*, May 2021.

- N. Grigoropoulos, M. Koutsoubelias and S. Lalis, **"Byzantine Fault Tolerance for Centrally Coordinated Missions with Unmanned Vehicles"**, *ACM International Conference on Computing Frontiers (CF),* May 2020.
- N. Grigoropoulos, M. Koutsoubelias and S. Lalis, **"Active Replication for Centrally Coordinated Teams of Autonomous Vehicles"**, *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2019.

Pointers

https://csl.e-ce.uth.gr/

https://mlsysops.eu

https://exeliigence.tech/

Feel free to ping for more info!

lalis@uth.gr