



Creation of a network and web-based video surveillance system for robot remote control.

Presented by: Loup RUSAK

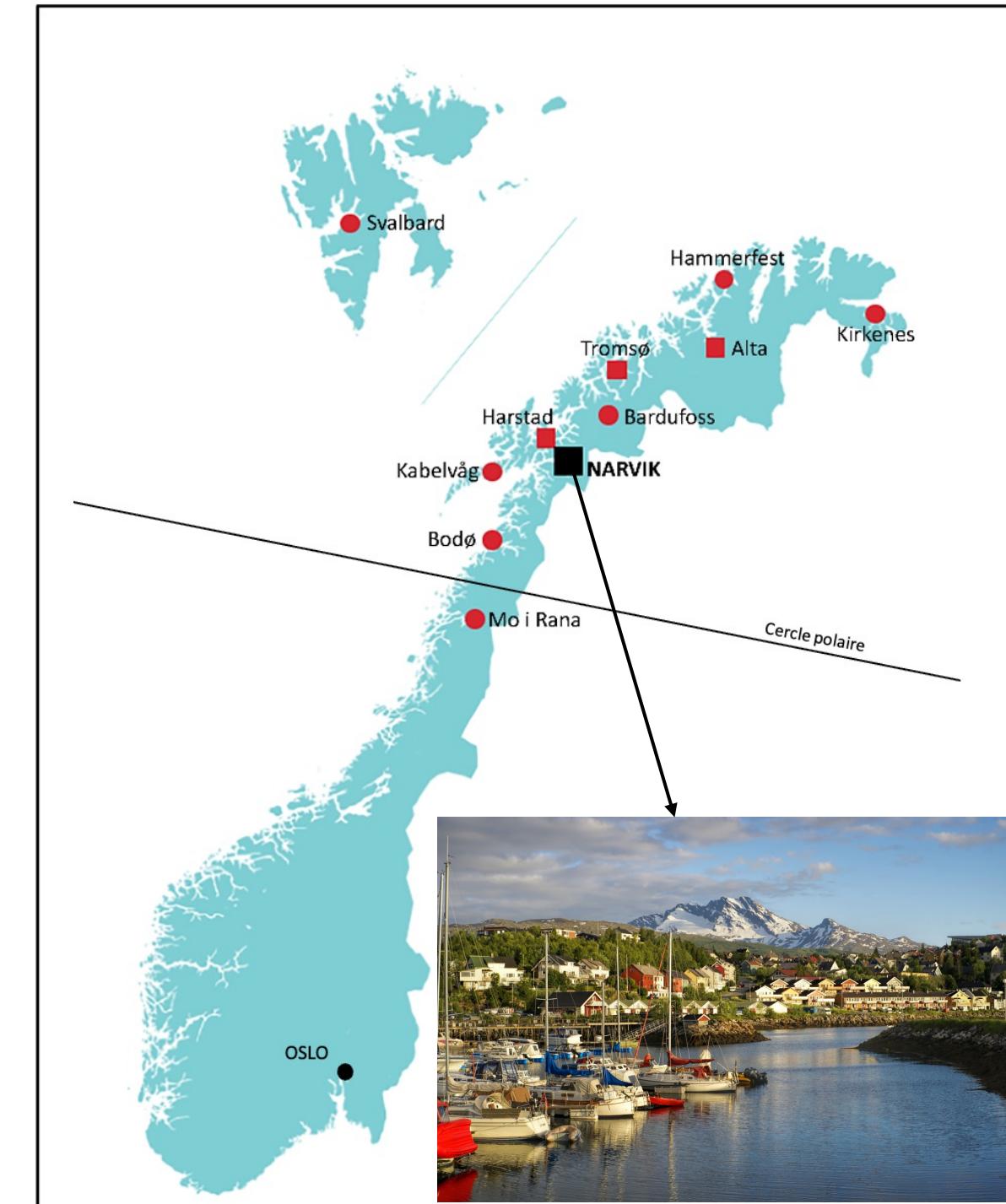
ISIMA Tutor: Romuald AUFRERE
UiT Tutors: Hao YU & Beibei SHU

Introduction



UiT The Arctic
University of Norway

- **Narvik** Campus
- Founded in : **1968**
- Formerly known as « **Tromsø University** »
- **Third largest** university of Norway
- **Northernmost** university in the world

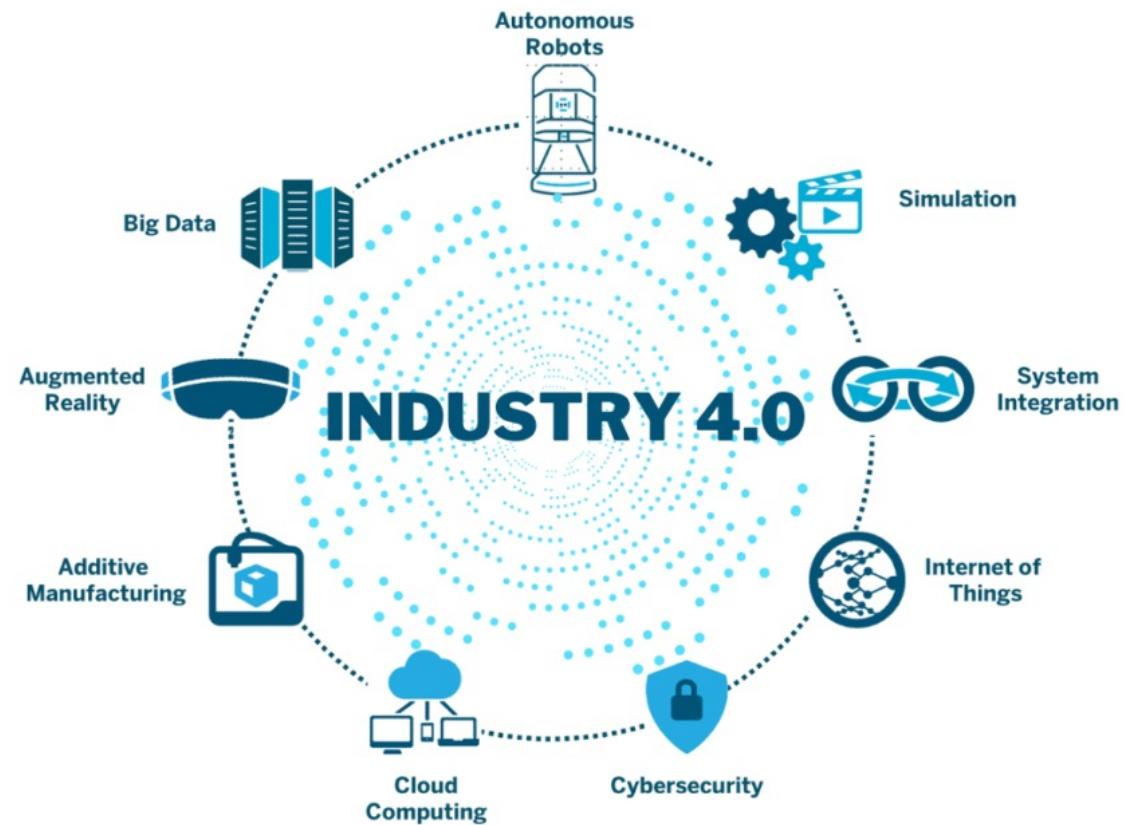


Introduction

- Industrial Engineering Department
- Own laboratory
- Robot **UR10E**
- **Pickit Camera**



Develop a smart logistic system



Introduction

Web based camera view for robot remote control

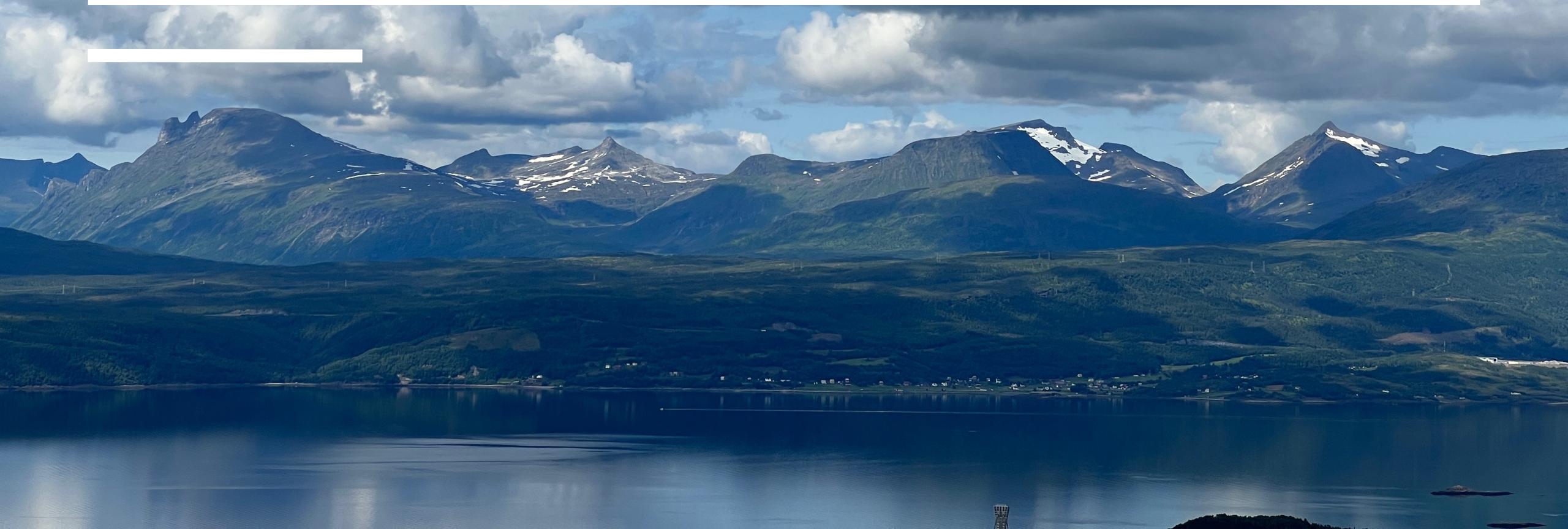
- Multiples cameras
- Point the robot cell
- Different angles



Ensure safety of Human and Robot



How to create from scratch a video surveillance system using Python technologies ?



Summary

1) Organization

2) Conceptualization

3) Realization



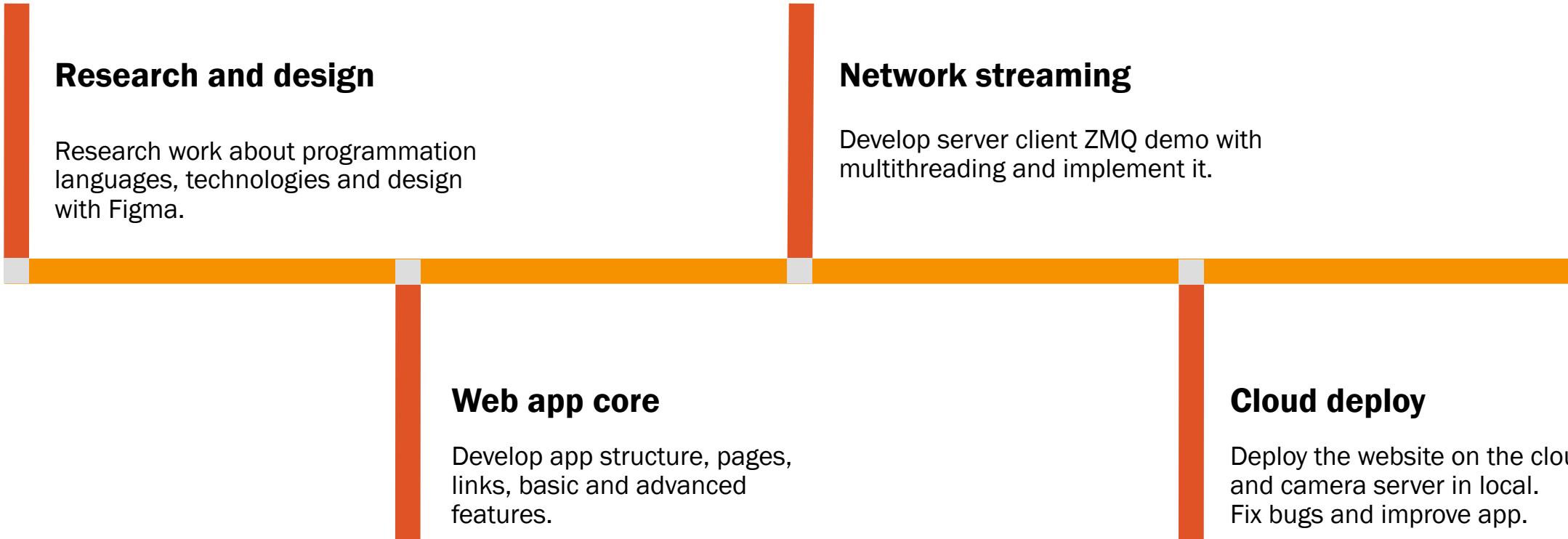
1) Organization



1) Organization



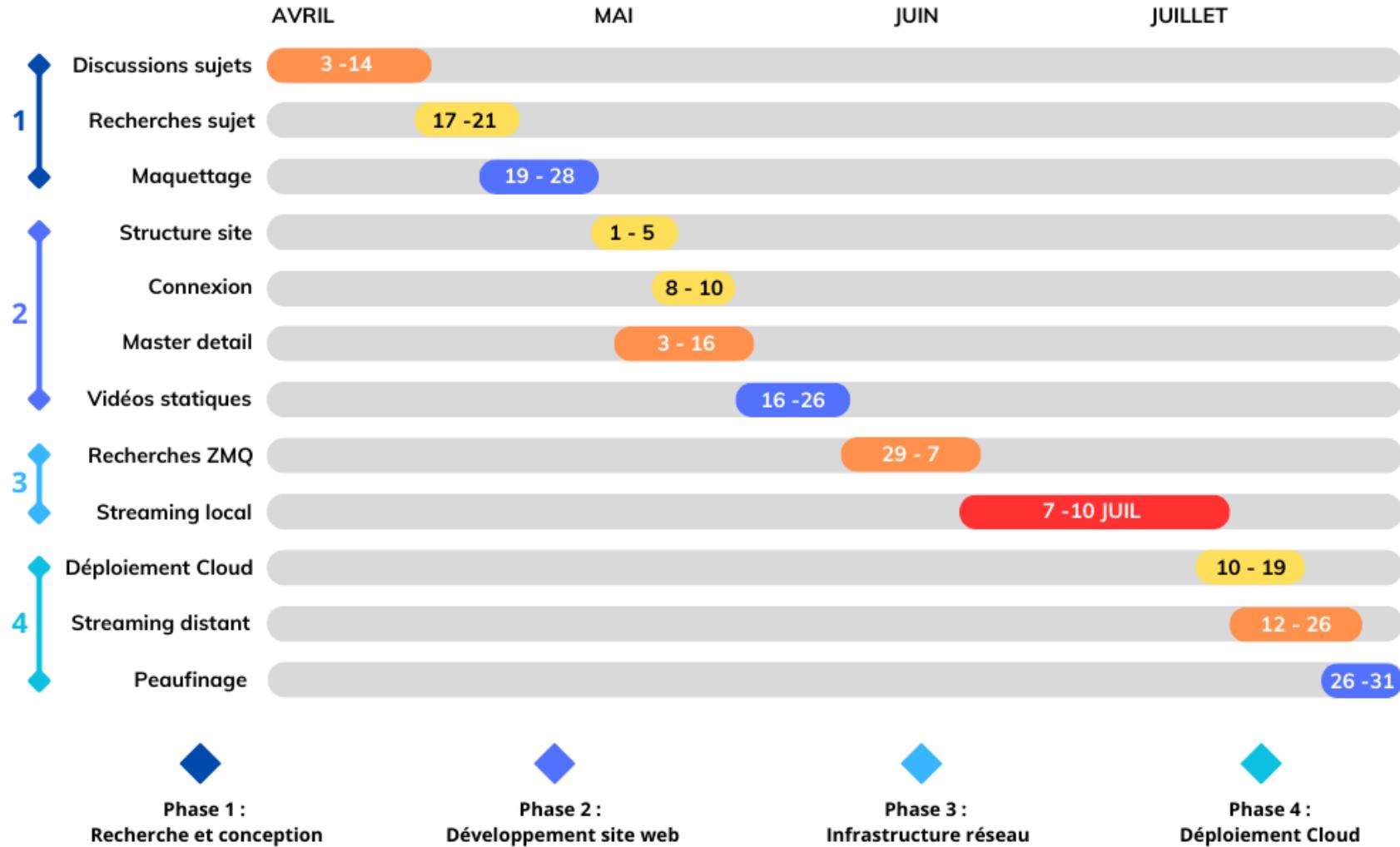
Production steps



1) Organization



Gantt Diagram



1) Organization



Hardware tools - cameras



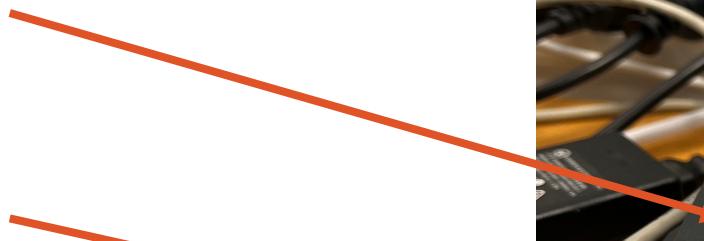
- 3 **IP wired** Cameras
- 3 view angles (right, left, top)
- **IR CCTV** Cameras
- Indoor/outdoor
- Day/night
- Embedded Linux system
- **3 streams** per camera
- **ONVIF** standard

1) Organization



Hardware tools - network

Switch for PC local network connection



Router for University local subnetwork connection



2) Conceptualization



2) Conceptualization



Software tools

- **Python** : main programmation language.
- **Flask** : Python framework for developping web apps.
- **Bootstrap** : Website creation tools library.
- **OpenCV** : Image processing library
- **ZeroMQ (ZMQ)** : Python network library for asynchronous communication.
- **Waitress** : Python server for Flask



2) Conceptualization



Design

- Design with **Figma**
- Inspired by **real examples**
- Common **navigation bar**
- **3 tabs**
- 2 types of **views**
- **Master Detail**



The image displays four screenshots of a user interface designed in Figma, illustrating the concepts of navigation, tabs, views, and master/detail design.

- Login page:** Shows a login form with fields for Username (ADMIN) and Password, set against a background image of a cityscape.
- Home page:** Features a navigation bar with tabs for Home, Cams, and All. Below the bar is a 3D rendering of a robotic arm with three cameras mounted on it, emitting signal waves.
- Cams page:** Shows a navigation bar with tabs for Home, Cams, and All. A sidebar on the left lists camera types: Top view (green), Front view (green), Left view (red), Right view (green), and 3D view (red). The main area displays a video feed from a "Right view" camera showing a person working at a workstation.
- All cams page:** Shows a navigation bar with tabs for Home, Cams, and All. It displays a grid of six video feeds. The top-left feed is labeled "Right view" (green dot) and shows a person at a workstation. The bottom-left feed is labeled "Left view" (red dot) and shows a robotic arm. The bottom-right feed is labeled "Top view" (green dot) and shows a wider view of the workspace.

2) Conceptualization



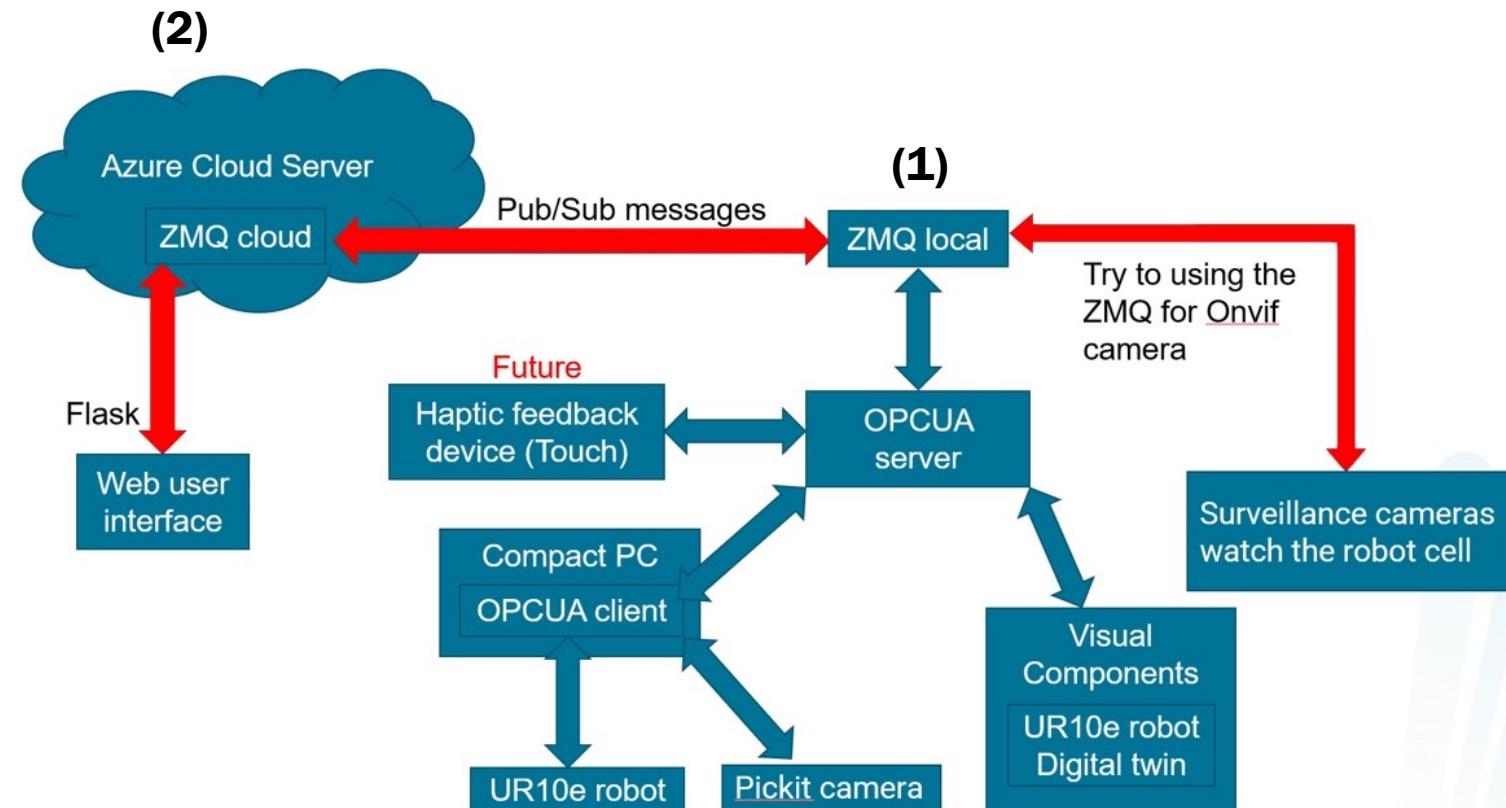
Network architecture

Local Server (1) :

- Get video from cameras
- Image processing
- Send data to Cloud

Cloud Server (2) :

- Receive data from network
- Image processing
- Display video on Flask web app



3) Realization



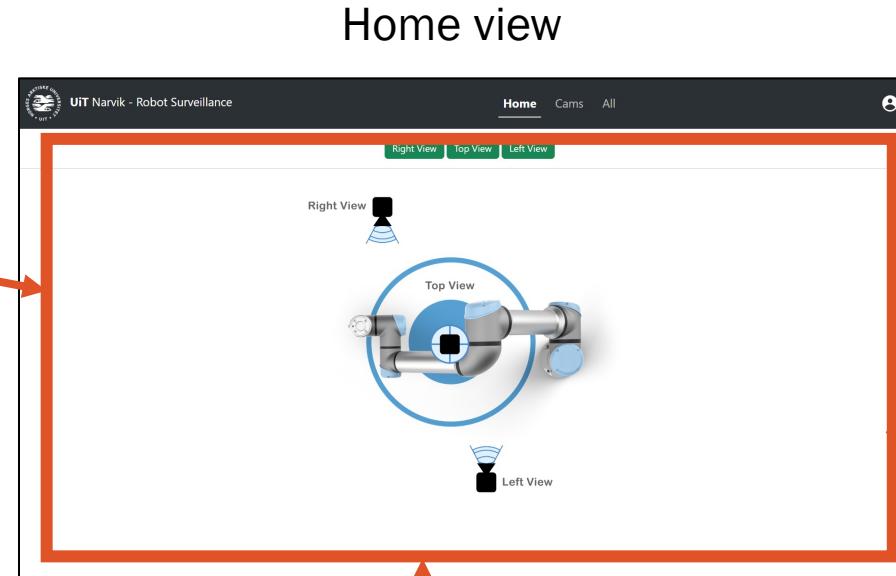
3) Realization



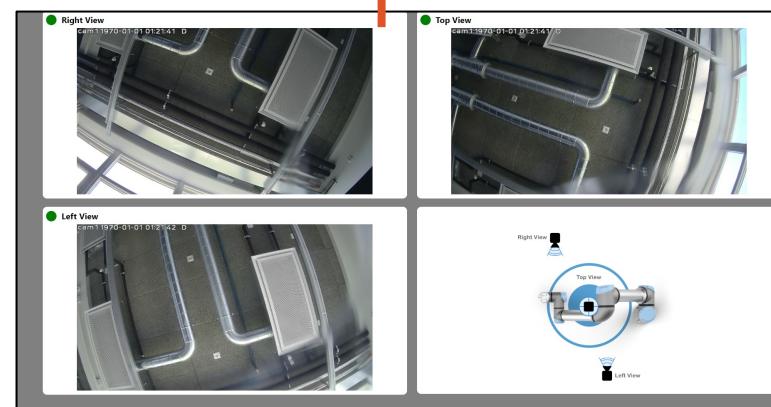
Website – Layout



Detail view



Home view



Global view

```
<!DOCTYPE html>
<html>
<head>
    <!-- Header -->
</head>
<body>
    <NavBar>
        <!-- Navigation Bar -->
    </NavBar>
    {% block content %}
        <!-- Page content -->
    {% endblock %}
</body>
</html>
```

3) Realization



Website – Master detail

```
<div id="sidebarMenu">
  {% for cam in cameras %}
    <div>
      <a
        class="list-group-item list-group-item-action py-2
        aria-current="true"
        data-cam='{{ cam | tojson | replace("\\"", "\'") }}'
      >
        {{ cam.name }}
        {% if cam.status=='active' %}
          <span class="dot cam-active"></span>
        {% else %}
          <span class="dot cam-inactive"></span>
        {% endif %}
      </a>
    </div>
  {% endfor %}
</div>
```

```
# Cameras data
cameras = [
  {'name': 'Right View', 'status': 'active', 'src': 'video_feed_1'},
  {'name': 'Top View', 'status': 'active', 'src': 'video_feed_2'},
  {'name': 'Left View', 'status': 'active', 'src': 'video_feed_3'}
]
```

The screenshot shows a web interface for 'UiT Narvik - Robot Surveillance'. At the top, there's a logo and navigation links for 'Home', 'Cams' (which is currently selected), and 'All'. Below this, a sidebar menu lists three camera views: 'Right View', 'Top View', and 'Left View', each with a green circular status indicator. To the right, a large video feed window displays a view from the 'Left View' camera, showing a dark room with a bright light fixture and a chain hanging down. The video feed has a timestamp 'cam11970-01-09 00:12:50 D'.

3) Realization



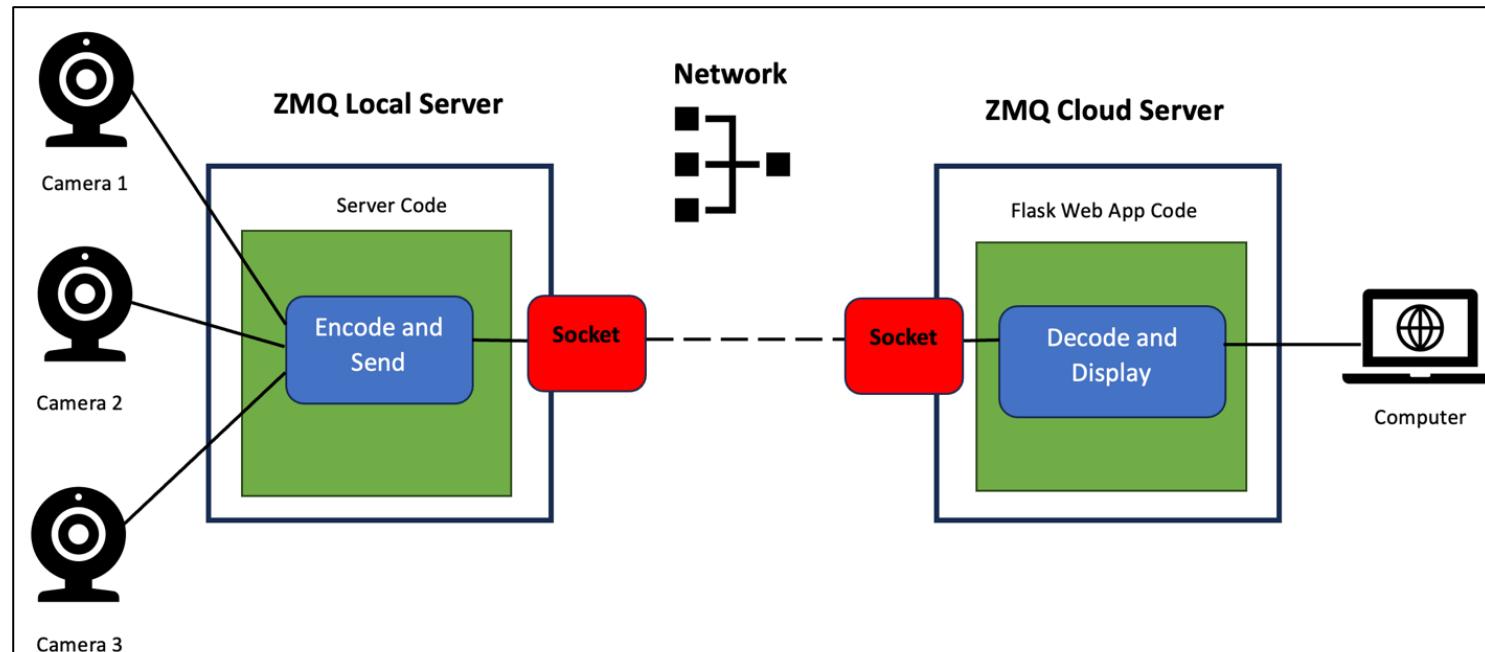
Streaming – Demo

ZMQ Local Server :

- PUB Socket
- Diffuse :
`bind('tcp://*:5555')`
- Video capture :
`VideoCapture / Read`
- Processing :
`imencode('.jpg', image)`
- Send :
`send_multipart(data)`

ZMQ Cloud Server :

- SUB Socket
- Listen :
`connect('tcp://ipaddr:5555')`
- Receive :
`recv-multipart()`
- Processing :
`imdecode(data)`
- Display



3) Realization



Streaming – Issues

1) Exponentially high video latency

- 1 or 2 seconds at the beginning
- Up to 45 seconds 1 minute later

Bottleneck somewhere ?

2) Time delay when switching views

- Leaving and come back
- Timestamp remains the same
- Non direct visualization

Frame queue ?

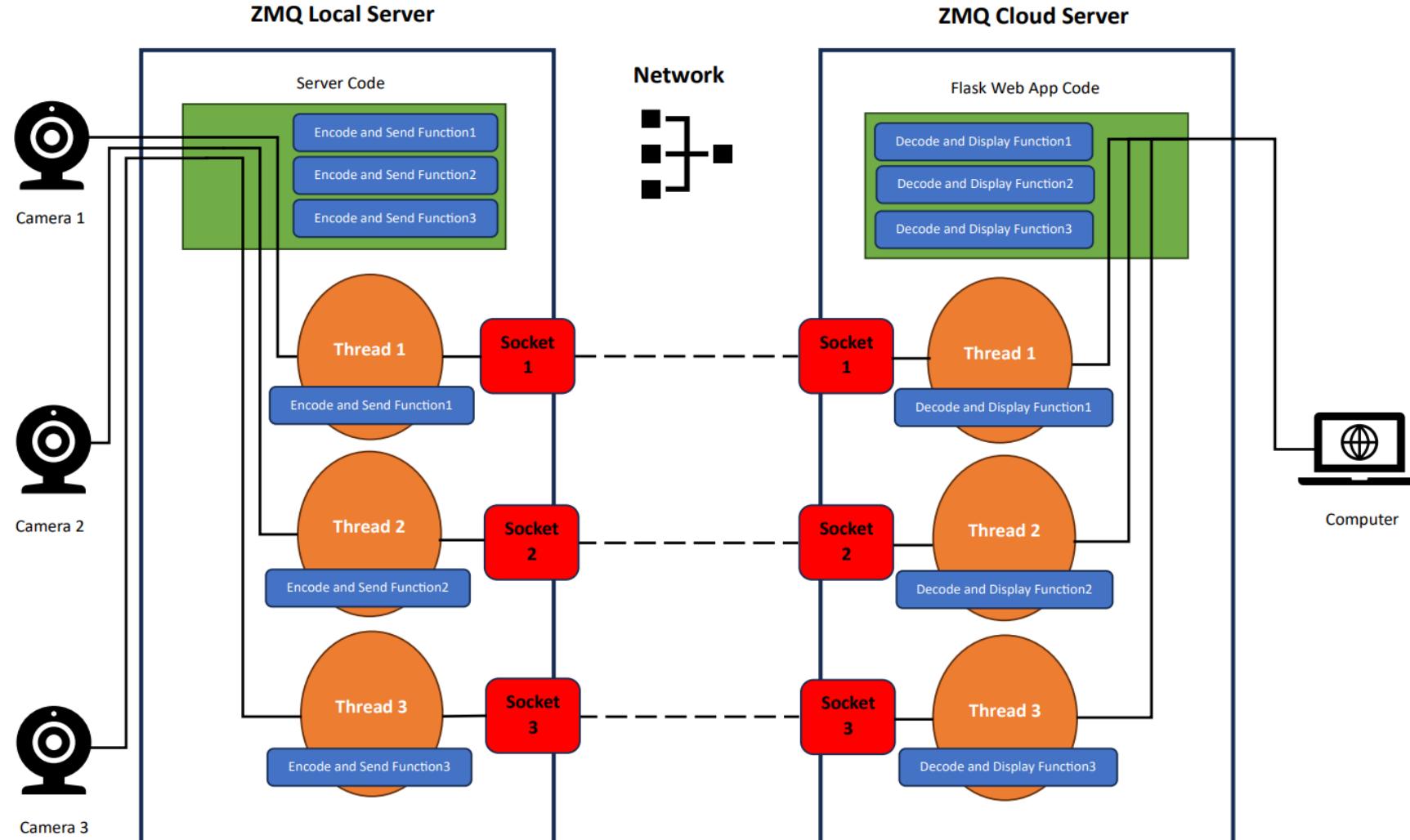
3) Realization



Solution 1 - Multithreading

On each side :

- **1 socket per camera**
- **1 thread per camera**
- **1 threaded function**
(encode/decode)



3) Realization



Solution 2 – Skipping frames

« Flush » socket waiting queue ?

Web app side :

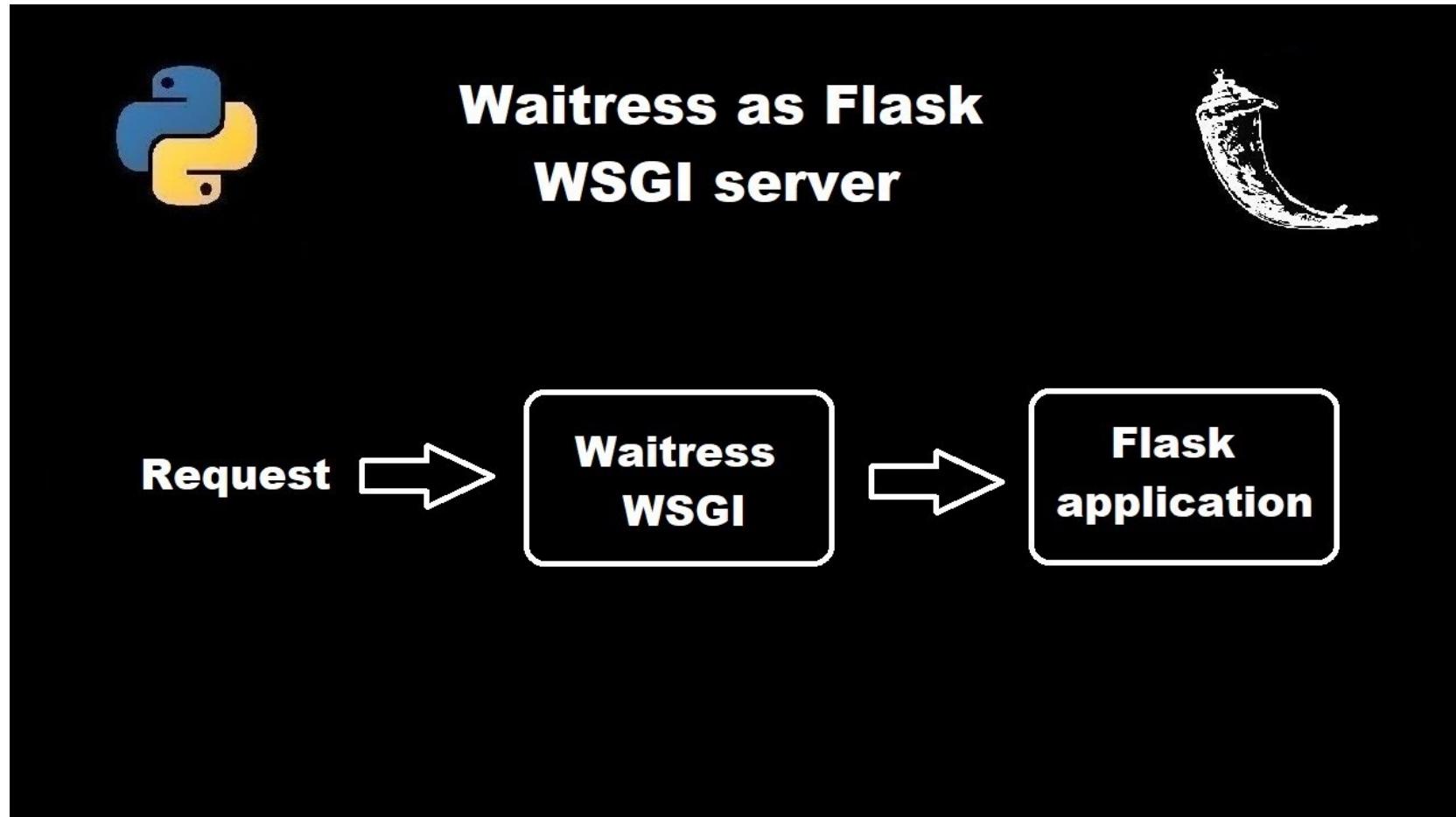
- Catch-up frames
 - Skip frames
-
- **Compare times**
 - **Calculate number of frames**
 - **Receive loop for skipped frames**
 - **No processing**

```
# Skipping lost frames
current_time = time.time()
if last_visualization_time is not None:
    elapsed_time = current_time - last_visualization_time
    skipped_frames = int(elapsed_time / frame_time)
    for _ in range(skipped_frames):
        try:
            socket.recv_multipart(zmq.NOBLOCK)
        except zmq.error.Again:
            break
    last_visualization_time = current_time
```

3) Realization



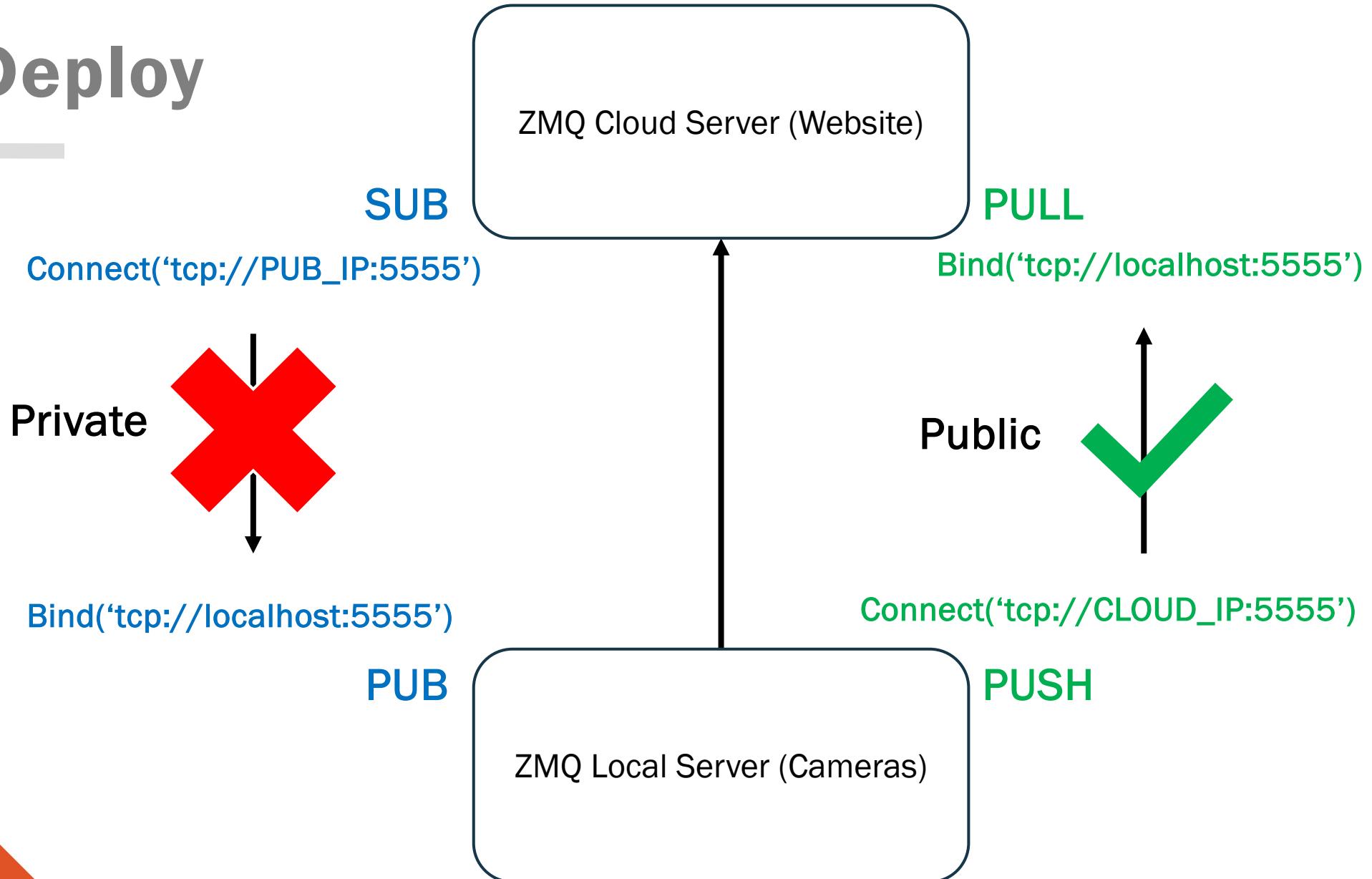
Cloud Deploy



3) Realization



Cloud Deploy



Conclusion

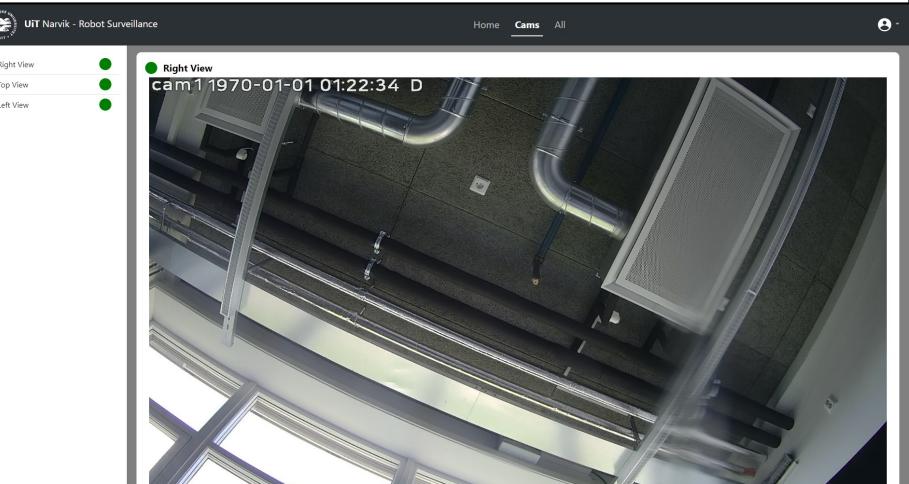
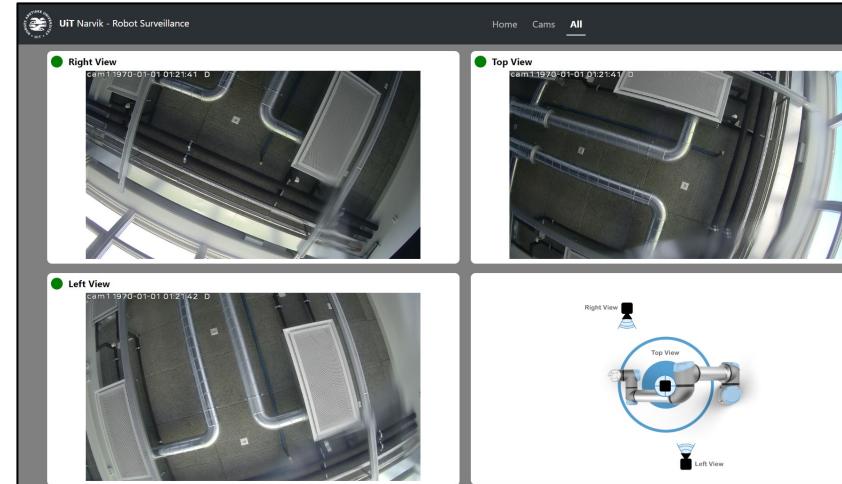
Content :

- Full web app for video surveillance
- Network architecture and servers

Skills and knowledge :

- Research project
- Autonomy
- Web development technologies

The screenshot shows the 'UIT Narvik - Robot Surveillance' web application. At the top, there is a logo for 'HØGSKOLEN I NARVIK' and 'UIT'. Below the logo, the title 'UIT Narvik - Robot Surveillance' is displayed. A 'Login' form is visible, containing fields for 'Username' and 'Password' and a 'Login' button. The main content area features a large blurred background image of a coastal town. In the center, there is a circular diagram of a robotic arm with three camera views labeled 'Right View', 'Top View', and 'Left View'. Below this diagram, there are three small video preview windows labeled 'Right View', 'Top View', and 'Left View' respectively, each showing a different perspective of a ceiling or wall area.



A wide-angle photograph of a sunset over a landscape. In the foreground, there's a paved road with a metal guardrail, a small modern building with a grey roof, and a tall, thin evergreen tree. The sky is filled with dramatic, wispy clouds colored in shades of orange, yellow, and blue. In the background, a large, calm lake stretches towards a range of dark, silhouetted mountains. The sun is low on the horizon, casting a bright glow and creating a lens flare effect.

Thank you for your
attention !