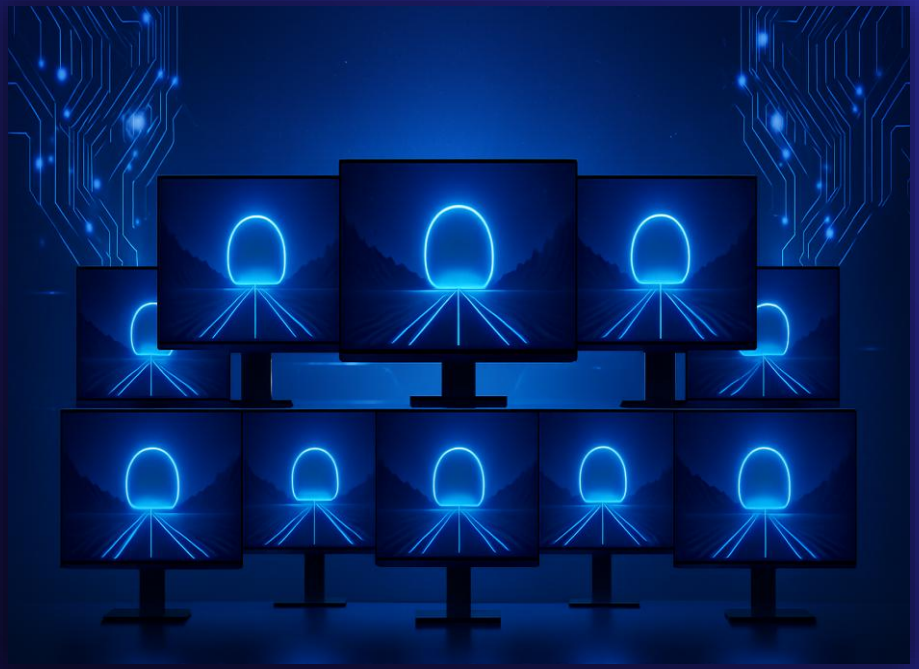


AI-Powered Gaze-Driven Selective Audio System

Team no: **ONL479**
by
Anu Dharshini B
D V R Surya Teja
Malathi A
Prasanna R.N

Problem Statement

The Challenge: In modern offices, control rooms, and classrooms with multiple monitors, users are bombarded with multiple audio sources simultaneously.

Objective: To develop a software-only AI system that intelligently activates audio from the monitor a user is looking at, creating a distraction-free and focus-enhanced multi-screen experience.

Infostrom — Gaze-Driven Selective Audio Demo (Jupyter Dashboard) ¶

Purpose: Interactive notebook demo that simulates gaze-driven audio focus using *simple face-position logic*. This version is tailored for **online Jupyter** environments where webcam access may be restricted. If a webcam is available, the notebook will try to use it; otherwise it falls back to the included demo video `demo_video.mp4`.

Features

- Inline frame display (no external GUI windows)
- 3 virtual monitors (left / center / right)
- Simple face-position gaze estimation (face center -> monitor zone)
- Visual indicator of which monitor is "active" and simulated audio status

[1]:
```
## ============================================================
# Enhanced AI Gaze Visualization Dashboard (Keyboard + Layout Polish)
# ============================================================
import cv2, numpy as np, time, asyncio
import ipywidgets as widgets
from ipycanvas import Canvas
from IPython.display import display
from math import sin

# ============================================================
```

```
# ============================================================
log_history = []

log_output = widgets.Output()
log_container = widgets.Box(
    [log_output],
    layout=widgets.Layout(
        border="1px solid #444",
        height="140px",
        overflow_y="scroll",
        overflow_x="hidden",
        width="100%",
        background="#111",
        padding="6px",
        color="#ddd",
        font_size="12px",
        flex_flow="column nowrap"
    )
)

def log_message(msg):
    """Append message safely to the log box."""
    log_history.append(msg)
    # Limit to last 50 entries
    if len(log_history) > 50:
        del log_history[0]
    log_output.clear_output(wait=True)
    with log_output:
```

```
# Canvas + Events
# ============================================================
canvas = Canvas(width=640, height=400)

def handle_mouse_move(x, y):
    if state.gaze_pattern == "mouse":
        state.target_x = x / 640
        state.mouse_y = y / 400
        state.mouse_active = True
canvas.on_mouse_move(handle_mouse_move)

def handle_key_down(event):
    k = event["key"]
    if state.gaze_pattern == "keyboard":
        if k == "ArrowLeft":
            state.keyboard_x = max(0.0, state.keyboard_x - 0.05)
        elif k == "ArrowRight":
            state.keyboard_x = min(1.0, state.keyboard_x + 0.05)
        elif k.isdigit() and 1 <= int(k) <= state.num_monitors:
            idx = int(k) - 1
            state.keyboard_x = (idx + 0.5) / state.num_monitors
            log_message(f"🎯 Jumped to Monitor {int(k)}")

canvas.on_key_down(handle_key_down)

def show_frame():
```

```
start_btn = widgets.Button(description="▶ Start", button_style="success")
pause_btn = widgets.Button(description="⏸ Pause", button_style="warning")
stop_btn  = widgets.Button(description="⏹ Stop",  button_style="danger")
reset_btn = widgets.Button(description="🔁 Reset", button_style="info")

speed_slider   = widgets.IntSlider(value=int(state.speed), min=10, max=200, step=5, description="Speed (ms)")
monitor_slider = widgets.IntSlider(value=state.num_monitors, min=2, max=8, step=1, description="Monitors")
pattern_dd     = widgets.Dropdown(
    options=["mouse", "sine", "linear", "keyboard", "zigzag"],
    value="mouse",
    description="Mode"
)

status_text = widgets.HTML("<b>Status:</b> 💤 Idle")
status_light = widgets.HTML("<div style='width:12px;height:12px;border-radius:50%;background:#ccc;display:inline-block;margin-
```

```
def update_light():
    color = "#28a745" if state.running else "#dc3545"
    status_light.value = f"<div style='width:12px;height:12px;border-radius:50%;background:{color};display:inline-block;margin

def update_status():
    update_light()
    icon = {
        "mouse": "🖱️",
        "keyboard": "⌨️",
        "sine": "🌊",
        "linear": "➡️",
        "zigzag": "⚡"
    }.get(state.gaze_pattern, "👁️")
    badge = f"🟩 Monitor {state.monitor_idx}" if state.monitor_idx is not None else "🟦 Idle"
    status_text.value = f"<b>Status:</b> {'🟢 Running' if state.running else '🔴 Stopped'} | {badge} | Mode: {icon} {state.ga
```

```
async def simulate_async():
    log_message("🟢 Simulation started.")
    state.running = True
    last_time = time.time()
    state.trail.clear()

    while state.running:
        mode = state.gaze_pattern
        if mode == "mouse":
            gx = state.target_x
        elif mode == "keyboard":
            gx = state.keyboard_x
        elif mode == "sine":
            gx = (sin(time.time() * 0.7) + 1) / 2
        elif mode == "linear":
            gx = (time.time() * 0.25) % 1.0
        elif mode == "zigzag":
            t = (time.time() * 0.5) % 2
            gx = t if t <= 1 else 2 - t
        else:
            gx = 0.5
```

```
def start_clicked(b):
    if not state.running:
        asyncio.create_task(simulate_async())
    else:
        log_message("⚠️ Already running.")

def pause_clicked(b): state.running=False; update_status()
def stop_clicked(b):  state.running=False; state.monitor_idx=None; update_status()
def reset_clicked(b):
    state.running=False
    state.trail.clear()
    state.frame = np.zeros((400, 640, 3), np.uint8)
    state.mouse_x = state.mouse_y = state.target_x = state.keyboard_x = 0.5
    show_frame(); update_status(); log_message("🔁 Reset complete.")

def on_speed_change(ch): state.speed = ch["new"]
def on_pattern_change(ch):
    state.gaze_pattern = ch["new"]
    log_message(f"🔄 Mode changed to {ch['new'].title()}")
    update_status()
def on_monitor_change(ch):
    state.num_monitors = ch["new"]
    log_message(f"🖥️ Monitors set to {ch['new']}")
```

```
controls = widgets.HBox([start_btn, pause_btn, stop_btn, reset_btn, status_light])
config   = widgets.HBox([speed_slider, monitor_slider, pattern_dd])

dashboard = widgets.VBox([
    widgets.HTML("<h3 style='text-align:center;color:#ddd;'>🧿 AI Gaze Visualization Dashboard</h3>"),
    controls, status_text,
    widgets.Box([config], layout=widgets.Layout(justify_content="center")),
    widgets.Box([canvas], layout=widgets.Layout(justify_content="flex-start", margin="0 0 0 40px")),
    widgets.HTML("<h4 style='color:#ddd;'>📜 Event Log</h4>"),
    log_output
])

display(dashboard)
show_frame()
update_status()
log_message("🎮 Ready! Choose mode then press ▶ Start.")
```

```
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Oct  8 01:40:02 PM UTC 2025

  System load:  0.0               Processes:               96
  Usage of /:   4.1% of 49.11GB   Users logged in:         0
  Memory usage: 14%               IPv4 address for enc1000: 148.100.79.241
  Swap usage:   0%

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


 ___ ____  __ __      _    _                    ___  _   _ _____
|_ _| __ )|  \/  |    | |   (_)_ __  _   ___  __/ _ \| \ | | ____|
 | ||  _ \| |\/| |    | |   | | '_ \| | | \ \/ / | | |  \| |  _|
 | || |_) | |  | |    | |___| | | | | |_| |>  <| |_| | |\  | |___
|___|____/|_|  |_|    |_____|_|_| |_|\__,_/_/\_\\___/|_| \_|_____|

================================================================================
=
Welcome to the IBM LinuxONE Community Cloud!

This server is for authorized users only. All activity is logged and monitored.
Individuals using this server must abide to the Terms and Conditions listed here
:
https://www.ibm.com/community/z/ibm-linuxone-community-cloud-terms-and-condition
s/
Your access will be revoked for any non-compliance.
================================================================================
==
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

linux1@infostorm:~$
```

File Edit View Run Kernel Tabs Settings Help

infostrom_demo.ipynb

Code

Notebook Python 3 (ipykernel)

Filter files by name

/ IBM Z Datathon 25 /

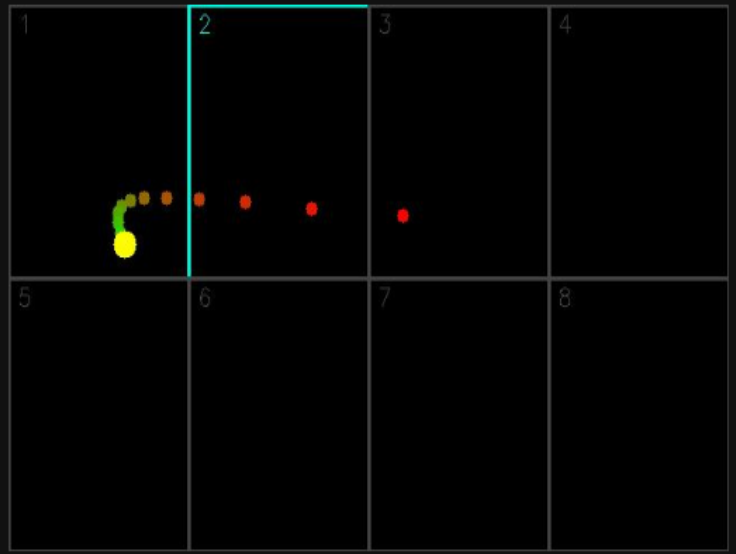
Name	Modified
demo_video.mp4	4 hr. ago
infostrom_demo.ipy...	42 sec. ago

AI Gaze Visualization Dashboard

▶ Start | ⏸ Pause | ■ Stop | ↻ Reset

Status: Running | Monitor 1 | Mode: Linear | FPS: 18

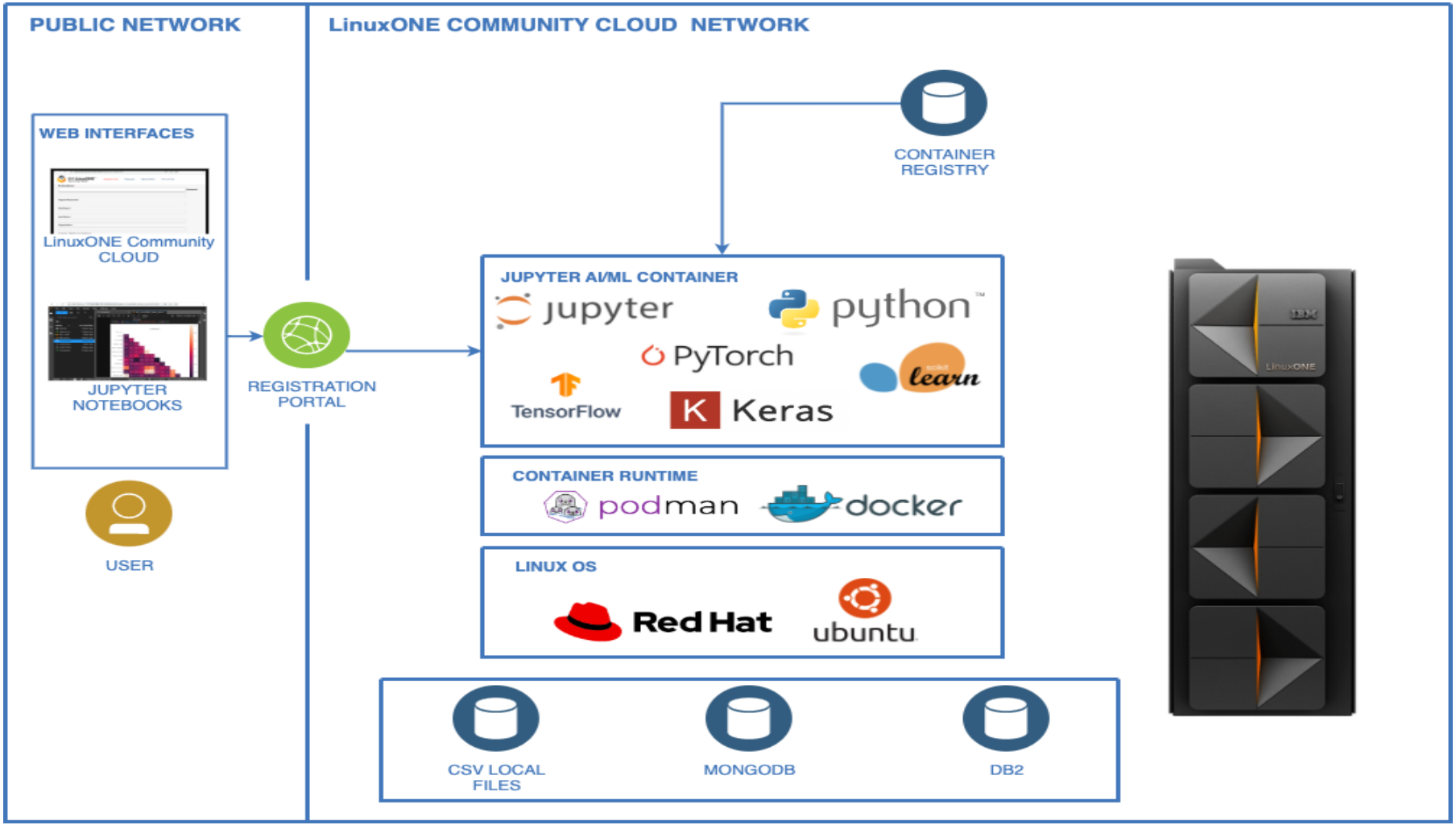
Speed (ms) 50 Monitors 8 Mode linear



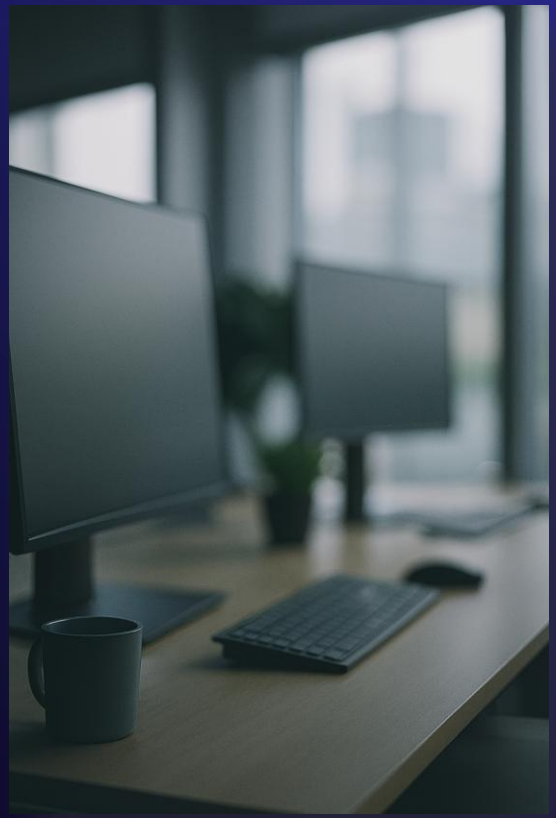
Event Log

```
Ready! Choose mode then press ▶ Start.
Monitors set to 7
Monitors set to 5
```

Simple 5 3 Python 3 (ipykernel) | Idle Mode: Command Ln 57, Col 27 infostrom_demo.ipynb 1


PUBLIC NETWORK
LinuxONE COMMUNITY CLOUD NETWORK
WEB INTERFACES
LinuxONE Community CLOUD
JUPYTER NOTEBOOKS
USER
REGISTRATION PORTAL
CONTAINER REGISTRY
JUPYTER AI/ML CONTAINER
jupyter
python
PyTorch
learn
TensorFlow
Keras
CONTAINER RUNTIME
podman
docker
LINUX OS
Red Hat
ubuntu
CSV LOCAL FILES
MONGODB
DB2

Impact and Value



Value to Community: Enables inclusive workspaces for people with attention deficits and improves focus in multi-display environments.

Usefulness: Simplifies multi-screen management without physical interaction or extra sensors.

Scalability: Designed for enterprise setups with multiple monitors per user — can scale to control rooms and smart classrooms.