Eagle Eye AI

Smart Following Camera with Face Recognition

Matjaz Zibert
https://hackster.io/matjaz4
https://github.com/s59mz

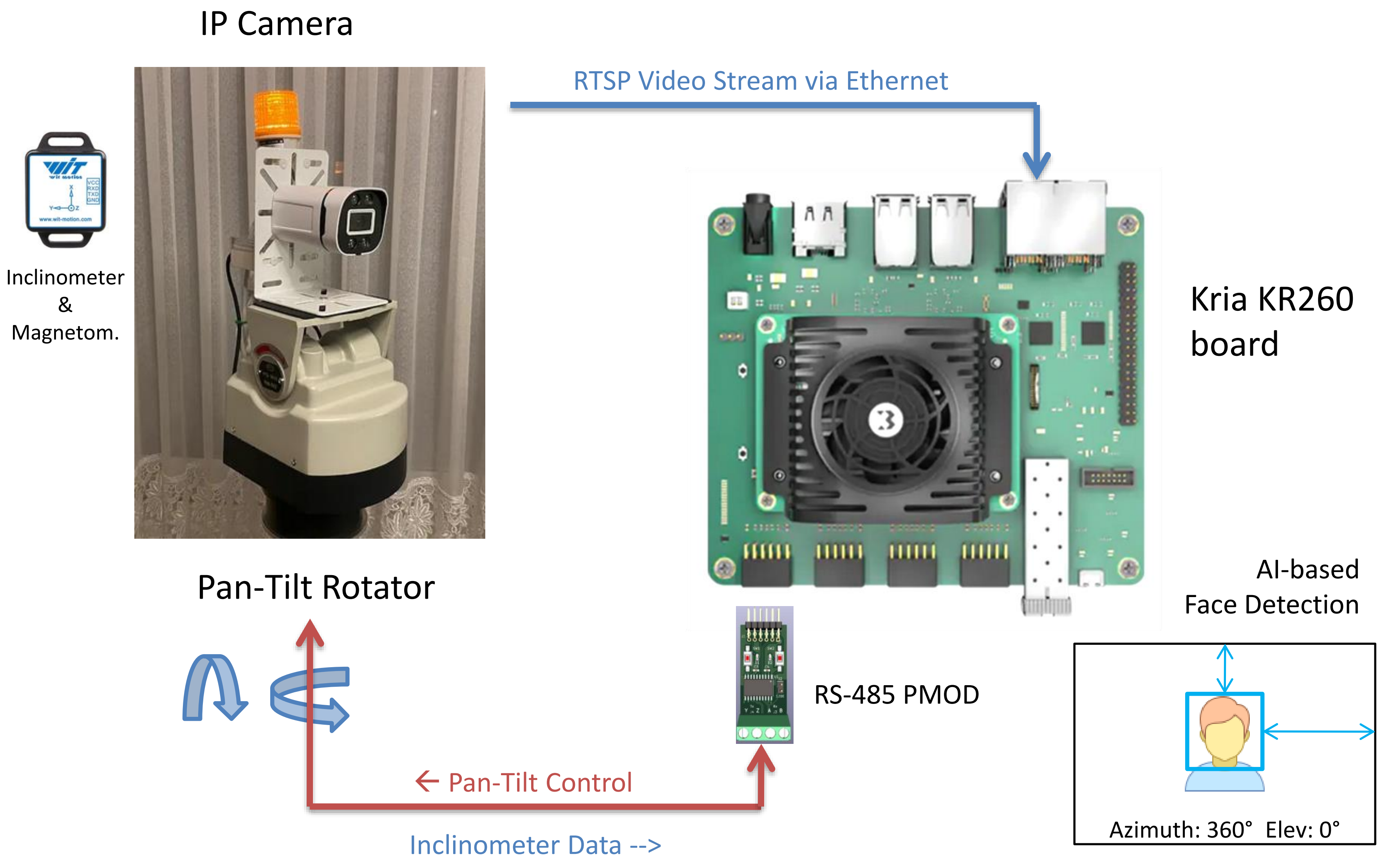
The Goal of Eagle-Eye-AI

- The main objective of this project is to integrate AI-driven **face detection** with **real-time pan-tilt camera tracking**.
- The aim was to create a smart and autonomous system capable of following and focusing on human subjects, with potential future enhancements to track various other objects.
- This system has broad applications, including security monitoring, interactive robotics, and video production.

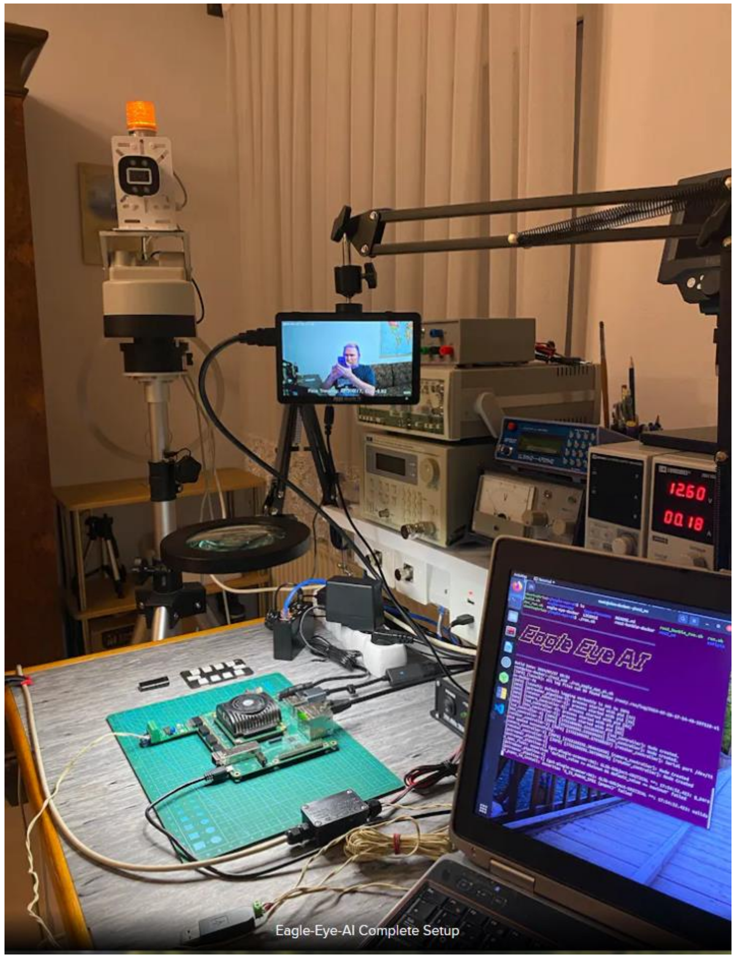
Content

- Live Demo
- Code Structure
- How to Use the Code
  - Building Docker Images
  - Starting the Application
- Project Complexity
- Project Value & Real-World Applications
- Q & A Session

Working Demo: Key Components

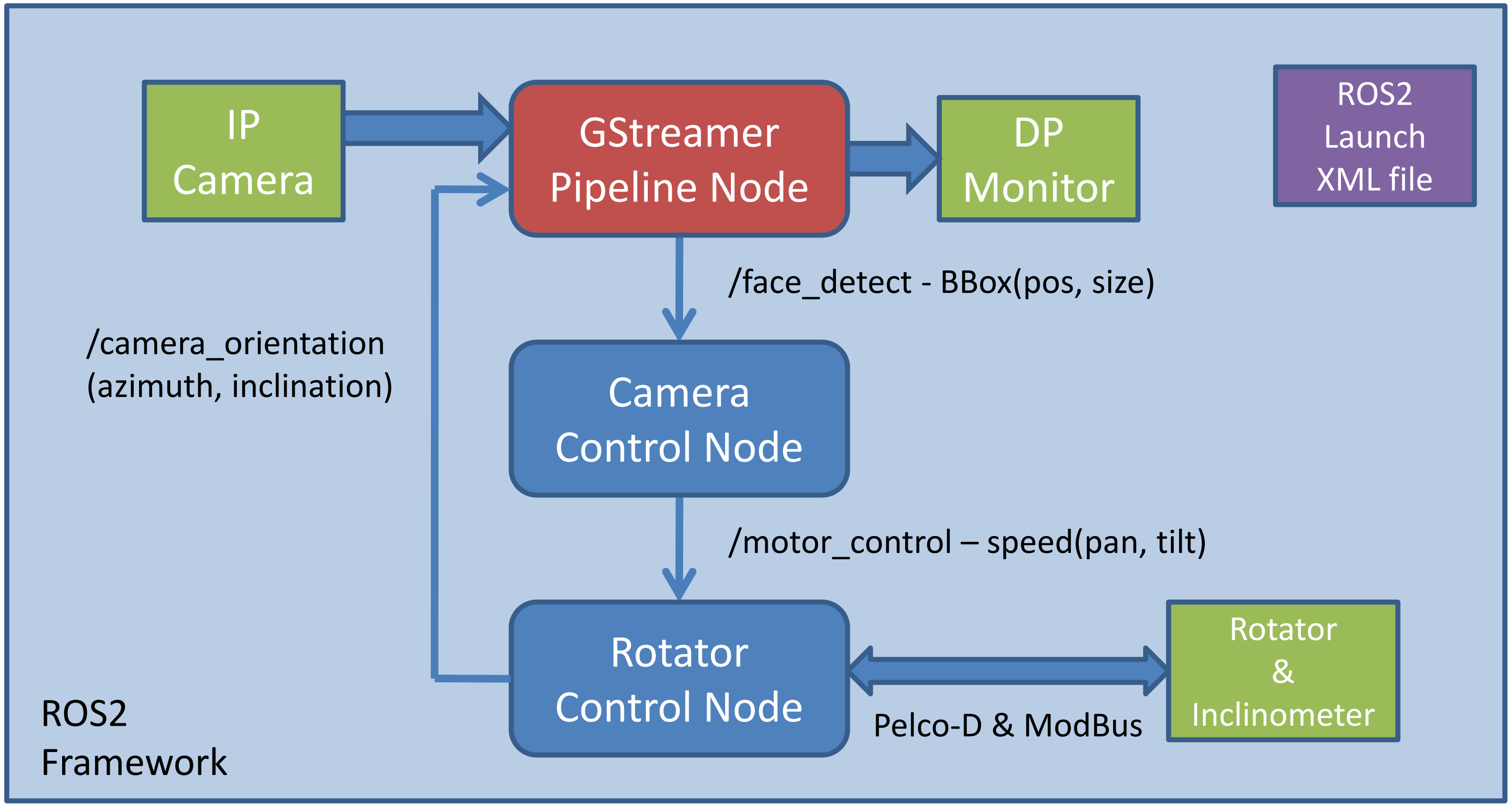


Live Demo



Code Structure: ROS2 Nodes

- The entire system is built on the ROS2 framework
- The GStreamer Pipeline is integrated into a ROS2 Node



Gstreamer Pipeline

- GStreamer Pipeline is embeded into the ROS2 Node
- Based on the official **Smartcam** demo application (taken fron Jupyter Notebook)

```
rtspsrc location=<camera_url> ! rtph265depay ! h265parse ! omxh265dec ! queue ! videoconvert !
videorate ! video/x-raw, format=NV12, framerate=30/1 !

tee name=t ! queue !

vvas_xmultisrc kconfig="/opt/xilinx/kr260-eagle-eye/share/vvas/facedetect/preprocess.json" ! queue !
vvas_xfilter kernels-config="/opt/xilinx/kr260-eagle-eye/share/vvas/facedetect/aiinference.json" !

ima.sink_master vvas_xmetaaffixer name=ima ima.src_master ! fakesink

t. ! queue max-size-buffers=1 leaky=2 ! ima.sink_slave_0 ima.src_slave_0 ! queue !
vvas_xfilter kernels-config="/opt/xilinx/kr260-eagle-eye/share/vvas/facedetect/drawresult.json"
name=draw ! queue !

kmssink driver-name=xlnx plane-id=39 sync=false fullscreen-overlay=true
```

gstreamer_pipeline Node

Easy of Use

- GitHub Repo: **eagle-eye-ai**
  - http://github.com/s59mz/eagle-eye-ai.git

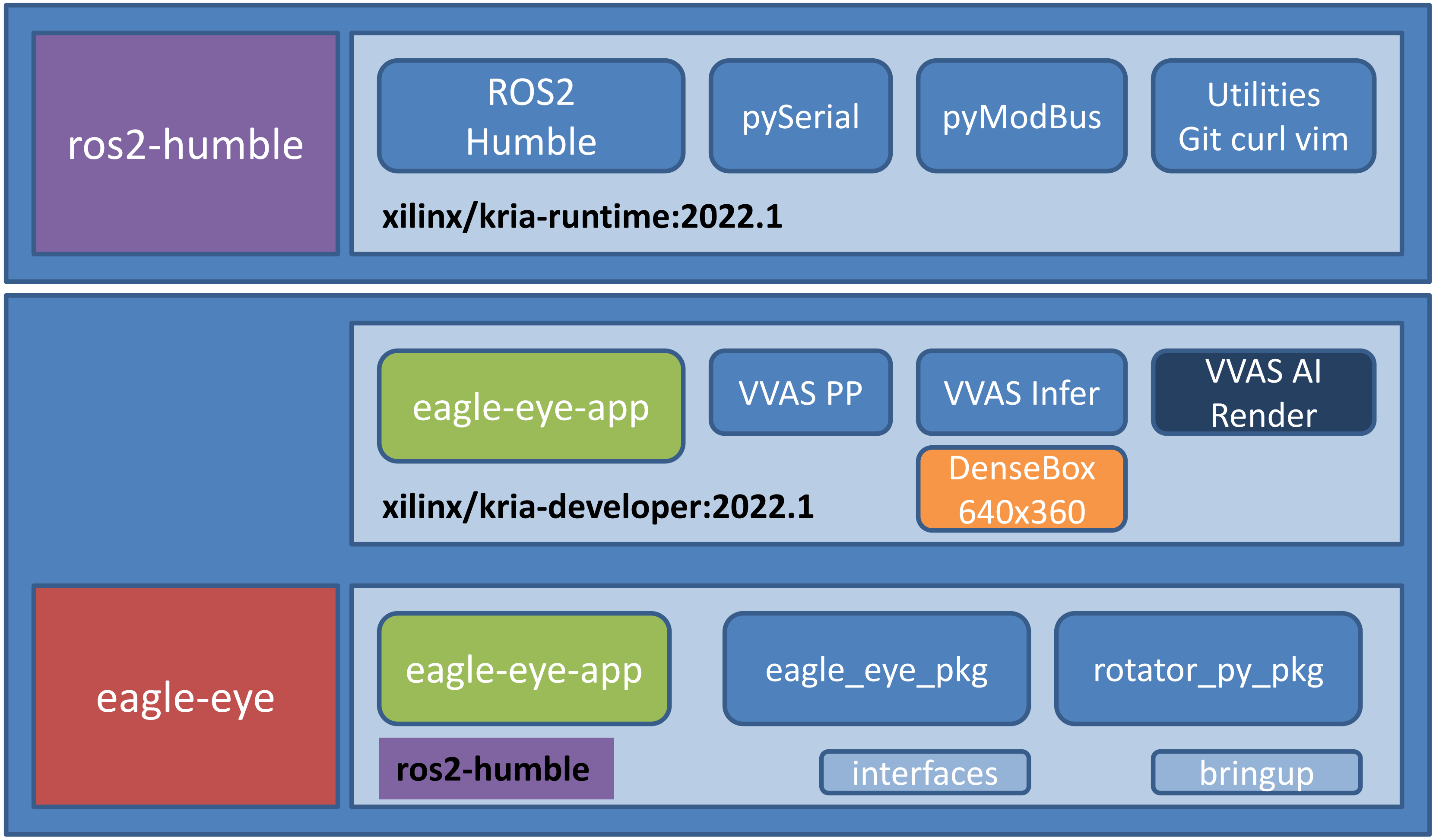
- FPGA Firmware – custom made platform
  - Debian file for Firmware Installation
  - "`sudo apt install ./firmware-kr260-eagle-eye.deb`"
  - "`sudo xmutil loadapp kr260-eagle-eye`"

- Application is in the Docker Image
  - **Build.sh**: builds the Eagle-Eye-AI Docker image
  - **Run.sh**: starts the Docker Image
  - **Run_eagle_eye_ai.sh**: starts the Application in the running Container

- User Interface
  - System is autonomous, the output video is displayed on HD Monitor
  - Two status LEDs for pan and tilt motors activity
  - Two push-buttons for rotating the camera manually.

Code Quality: Docker Images

- **Build.sh**: All Docker images are build by one script only
- **Run.sh**: Starts the final Docker image with all necessary configuration
- **Run_eagle_eye_ai**.sh: Bringup the Application by ROS2 launch xml script

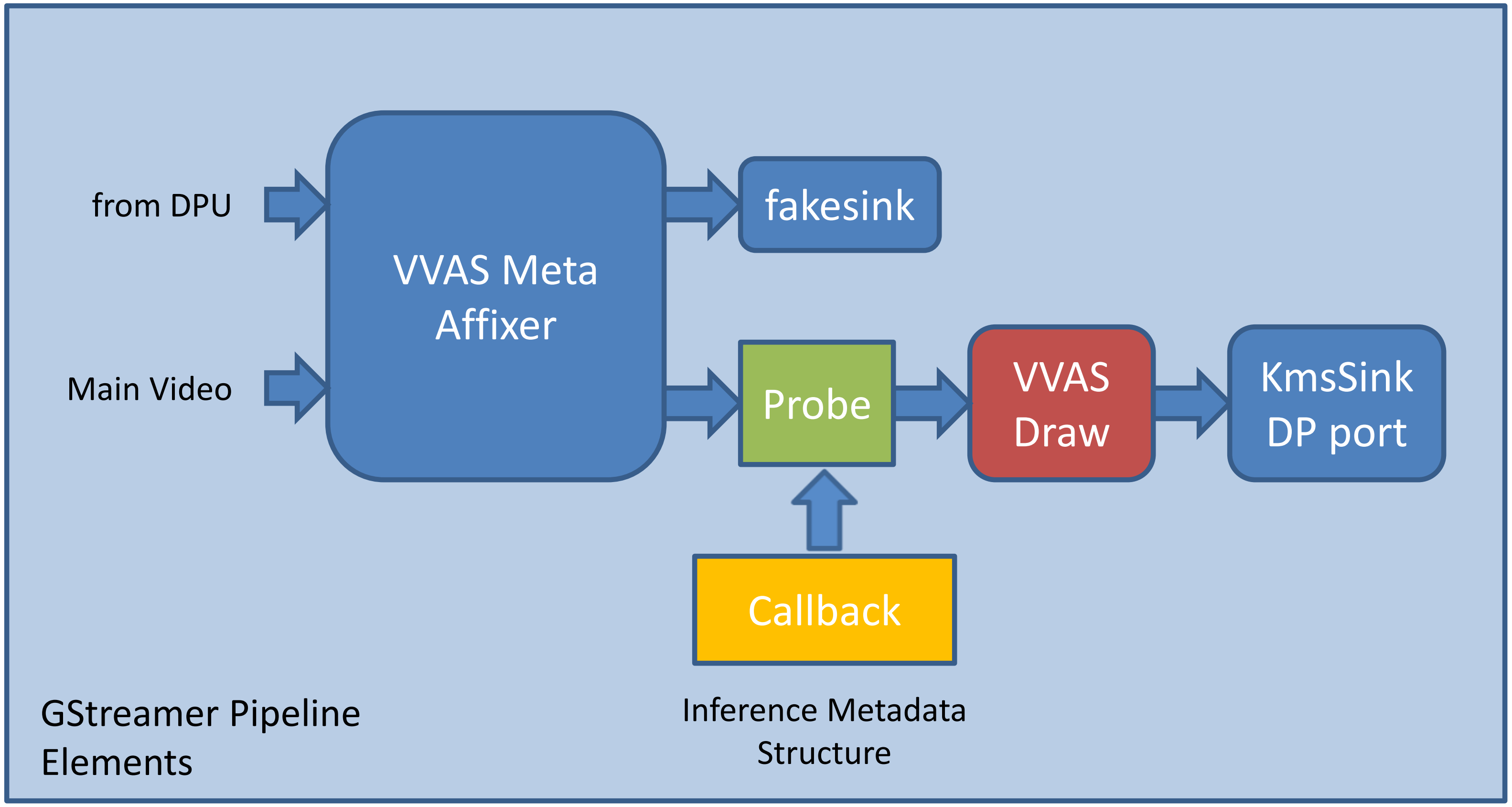


Project Complexity

Technical Challenge	Solution
IP Camera Integration	Adapted the GStreamer pipeline from the SmartCam demo by replacing the input with an RTSP source and utilizing the H.265 hardware decoder for efficient video streaming.
ROS2 and GStreamer Integration	Integrated the GStreamer pipeline within a ROS2 Node to enable real-time video streaming and AI inference. Utilized ROS2's communication framework to ensure scalability and high performance.
Extracting Boundary Boxes from Inference Data	Inserted a probe with a callback function into the Draw element of the GStreamer pipeline to process AI inference metadata and extract face detection boundary boxes. Also, update metadata with inclinometer data to show the camera orientation status on-screen.
Handling Multiple Serial Protocols on One Bus	Integrated Pelco-D and ModBus protocols on a shared RS-485 bus. Implemented both protocols within a single ROS2 Node and thread, carefully coordinating message transmission and reception while flushing buffers to prevent collisions.

The Most Challenging Part

- Extracting the Boundary Boxes from the Inference Metadata structure
- Updating the Inference Metadata structure with Inclinometer Data



Project Value & Real-World Apps

Real-World Applications	Eagle Eye AI has a wide range of real-world applications, from security surveillance to wildlife monitoring and video productions. Its real-time object tracking makes it suitable for use cases where immediate response and accurate tracking are crucial.
Scalability & Future Improvements	The project is designed to be scalable—additional cameras and sensors can be easily integrated using the same ROS2 architecture. It can also be deployed in different environments, such as remote wildlife reserves, urban security networks, or even industrial inspection systems. Can be also integrated with drones.
Project's Adaptability	With its flexibility, the system can be adapted to track different kinds of objects, like vehicles or people, or to monitor large areas with more precision.