I- The Main Objective of the Jetson:

As part of this project, object detection was carried out using the Jetson Nano and the YOLOv8 algorithm. The Jetson Nano, developed by NVIDIA, is a small but powerful computing board designed specifically for artificial intelligence (AI) applications and real-time image processing.

YOLOv8, short for "You Only Look Once," is a real-time object detection algorithm known for its speed and efficiency. It enables the detection and classification of various objects in a scene in a single pass, thus offering optimal performance for applications such as video surveillance, autonomous driving, and many others.

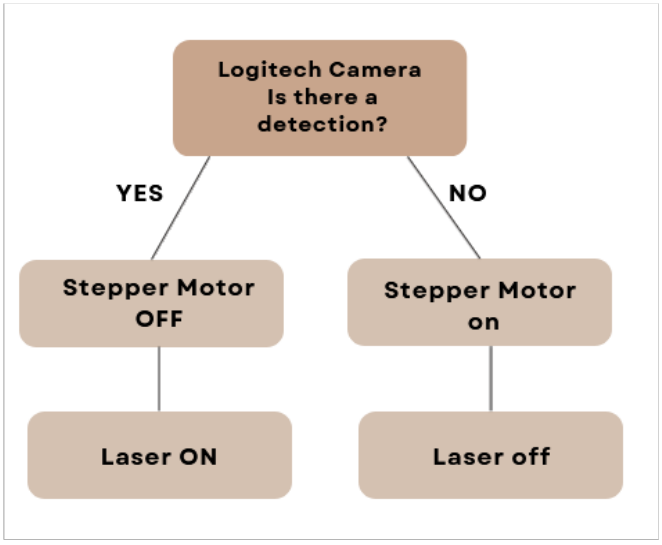
The main objective of using the Jetson Nano combined with YOLOv8 was to perform real-time detection of a specific target in each environment. Once the target was detected, the Jetson Nano was expected to be able to communicate with other devices to trigger specific actions based on the detection results.

In this project, the Jetson Nano was configured to communicate with an Arduino Uno board via a serial communication protocol. The Arduino Uno, a popular electronic prototyping platform, was used to control various peripherals and perform actions in response to signals from the Jetson Nano.

For example, once the target was detected by the Jetson Nano, the coordinates or other relevant information could be transmitted to the Arduino Uno via the serial link. The Arduino Uno could then use this data to activate actuators such as motors, lights, or other devices, depending on the specific application scenario.

This integration between the Jetson Nano and the Arduino Uno enables intelligent and responsive interaction between object detection and device control, paving the way for a multitude of practical applications.

II- Machine State:



Explanation of the Algorithm:

The system begins with the Logitech camera, which attempts to detect a target. If no target is detected, the stepper motor continues to rotate in search of the target.

When a target is detected, the stepper motor stops, and the laser and the two LEDs turn on.

After detection, the system resets: the LEDs and laser turn off, ready for a new detection and launch cycle.

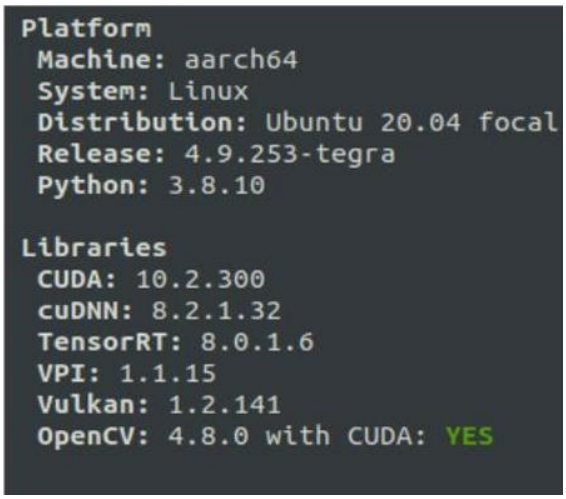
In summary, continuous communication exists between the Jetson, the camera, and the Arduino to ensure accurate target detection and activation of the laser.

III- Progress of Part 2:

1. Preparation of the Working Environment

1.1 Download and Installation of the System Image

To begin, I downloaded a system image optimized for the Jetson from Q-Engineering.
This image uses Ubuntu 20.04 and supports several libraries such as PyTorch and OpenCV, compiled with CUDA to utilize the Jetson's GPU, which allows for good performance.



1.2 Initial Performance Test

After the installation, I tested the performance using YOLOv8n, the fastest version of YOLO, and achieved a rate of 13 FPS (frames per second). To improve this performance, I decided to export the model in FP32 and FP16 formats, and I found that the FP16 format provided better performance, reaching 17 FPS.

1.3 Installation of Required Libraries

To do this, I downloaded and installed Ultralytics using the command pip3 install ultralytics. Then, I removed the default version of OpenCV because the new version was not compiled with CUDA, which is crucial for performance on the Jetson.

2. Data Preparation

2.1 Data Collection and Preparation

To train my own model, I found a dataset containing several targets, already annotated, which greatly simplifies the training process. The presence of annotations is essential as it clearly defines the objects to be detected in each image, thereby optimizing the training process.

Roboflow Universe Link: https://universe.roboflow.com/

3. Model Training

3.1 Using Google Collab

To train the model quickly, I used Google Collab, which provides access to powerful GPUs such as the Tesla A100. This GPU, which costs around 8000 euros, significantly accelerated the training process.

Here is the link to the Collab used:

https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolov8-object-detection-on-custom-dataset.ipynb

3.2 Training Parameters

I trained the model on 800 images, with the number of epochs set to 100. This means the model went through the 800 images 100 times. It's important not to increase the number of epochs excessively to avoid overfitting.

Overfitting happens when the model learns the details and noise of the training set too well, which harms its performance on new data.

3.3 Training Result

At the end of the training, a file named best.pt was generated. This file contains the model's optimized weights and was exported in FP16 engine format for optimal performance on the Jetson.

4. Use of the Trained Model

4.1 Implementation and Testing on Jetson

With the model training completed, I used the best.pt file exported in FP16 format for final tests on the Jetson. The model demonstrated satisfactory performance, confirming its effectiveness in object detection tasks.

5. Integration with Arduino

5.1 Using PyFirmata for Communication

To facilitate communication with the Arduino, I chose to use Python as the programming language. This is made possible through the PyFirmata library. PyFirmata allows an Arduino board to be controlled directly from Python, greatly simplifying the integration process. PyFirmata is easy to install and use, with examples available in the library's installation files.

5.2 Arduino Configuration

I configured the Arduino using PyFirmata to enable efficient serial communication with the Jetson. On the Arduino, I uploaded the Firmata firmware, which allows the Arduino to receive and execute commands sent from a Python script.

5.3 Development of Python Scripts

On the Jetson, I created two Python files: main.py and control.py.

- main.py: This file initializes the AI model and starts the target detection process. When objects are detected with sufficient confidence, the information is sent to control.py.
- control.py: This file receives the data from main.py and controls the external components connected to the Arduino, such as LEDs, laser, and stepper motor.

5.4 Establishing Serial Communication

I set up serial communication between the Jetson and the Arduino via a USB port. This ensures fast and reliable data transmission, which is essential for real-time control of external components.

This part has reached a 100% maturity level, with a fully ready and operational solution. All planned activities were successfully completed within the given deadlines, demonstrating a generally satisfactory level of progress.

IV- Year-over-Year Project Improvements

The system has seen notable advancements compared to last year's version. The **object detection model** was upgraded from the standard YOLO to **YOLOv8 trained on a custom dataset**, significantly improving detection accuracy. In terms of **motor control**, the old step-based movement was replaced by **continuous rotation**, offering smoother and more precise tracking of targets.

On the **safety** front, the shift from an electromagnetic cannon to a **laser pointer** marks an improvement in reducing risk and enhancing operational safety. Finally, the **dataset** used for training was greatly enhanced—transitioning from a small, general dataset to a **large and diverse one**, complemented by **data augmentation**, which boosted model performance and reliability.

V- Planned System Improvements for Next Year

To enhance performance and system efficiency, the following upgrades are planned for the upcoming year:

- **Switch to Orin Nano**: Replacing the Jetson Nano with the **Orin Nano**, offering **30x more processing power**, will allow faster AI model inference and better real-time performance.
- **Dual Arduino Setup**: Introducing **two Arduino units**—one for motor control and one for laser control—will reduce system complexity and improve **cable management**.
- **LIDAR Integration with ROS**: Integrating **LIDAR with the Robot Operating System (ROS)** will enable more accurate object detection, mapping, and improved **environmental awareness**.
- **Sensor Fusion**: A **sensor fusion system** will be developed to combine inputs from lidar, camera, and ultrasonic sensors, enhancing perception and decision-making.

These improvements aim to significantly boost system reliability, intelligence, and responsiveness.