**Problem Description**

**Name:** Prabhdeep Singh Sethi

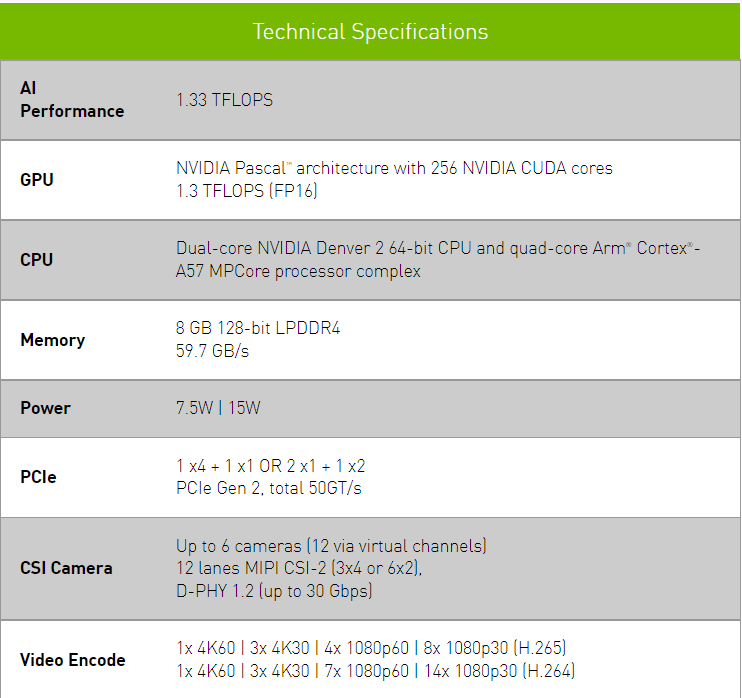
**Email:** [prabhdeepsethi0@gmail.com](mailto:prabhdeepsethi0@gmail.com)

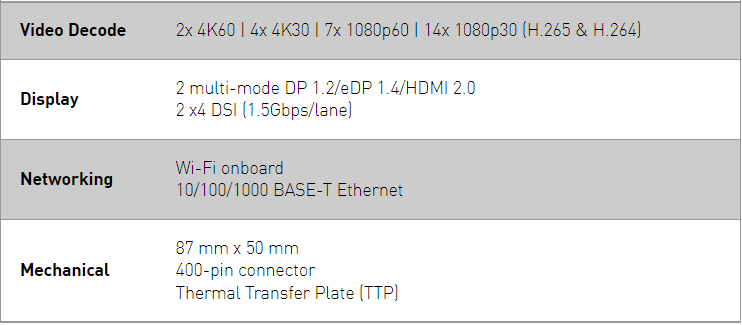
**Mob.**: +919112360030

**Place:** Nagpur, Maharashtra, India

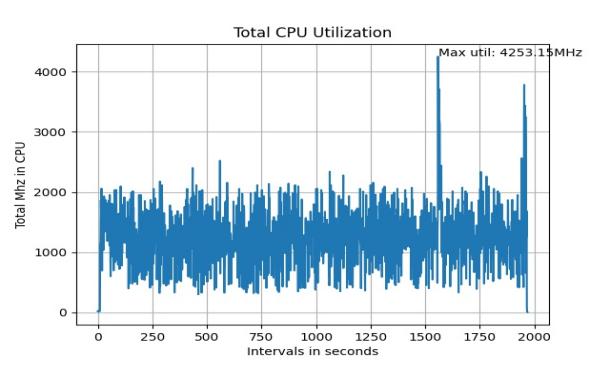
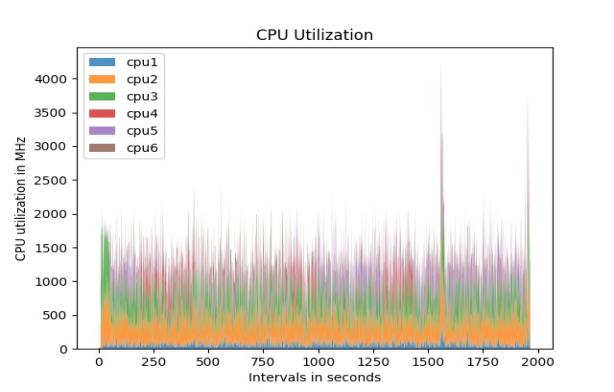
**Date:** 15/12/21 (15th December 2021)

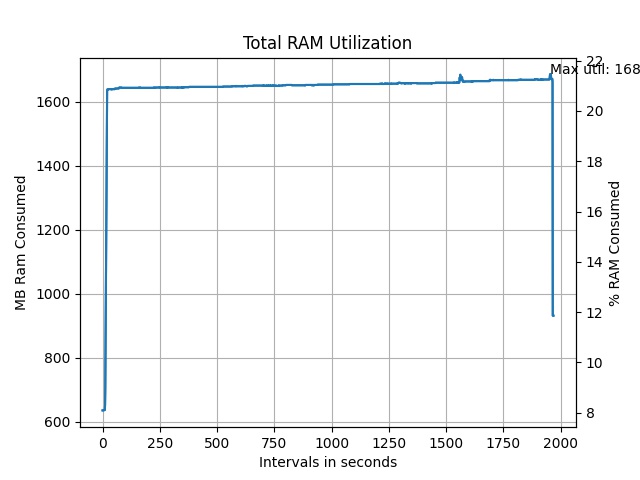
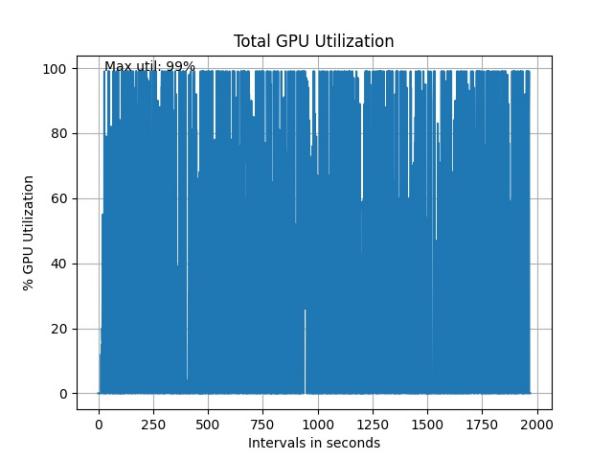
1. **Objective:**
   1. The objective of the project is to make a real-time embedded drone detection system from the infrared data. The data will be collected with different UAVs, altitude, distance and lighting conditions after that team will develop a model which will be able to detect the UAVs on new data. The proposed system will process in the following ways:
      1. Pre-processing of the data received from the IR camera.
      2. Defining the ROI to start the detection of UAV.
      3. Detection of the UAV with confidence index above defined threshold.
      4. Continuing the detection and/or tracking with guided search.
      5. The output of the system would be the a co-ordinate of the centroid of bounding box of UAV as well as the confidence index.
      6. System will be deployed on a custom PCB which needs to fit in a 70 mm diameter space.
      7. System should work in a dynamic environment with high body rates, velocity and non-co-operative situations.
      8. System should be able to process 50 frames per seconds (FPS).
   2. A gist of it is: Detecting UAV in IR video at 50 FPS and the system would be deployed on a custom PCB of size 70mm diameter.
2. **Key Specifications:**
   1. Hardware size: 70mm diameter / 49 mm square
   2. Inference time: 20 ms
   3. Start time / Boot time Required: 2s
   4. Other environmental condition:
      1. Shock proof
      2. Operational temperature: Military grade (-55°C to +125°C)
      3. Electromagnetic interference
   5. Power Availability: (Data not available)
3. **Current Approach to solve the problem:**
   1. Problem: To find out the hardware required for running our desired software
   2. Solution:
      1. First we started with a hardware readily available with us so that we can benchmark it and find out the requirement accordingly
      2. We had with us a [Nvidia Jetson TX2](https://www.nvidia.com/en-in/autonomous-machines/embedded-systems/jetson-tx2/). Attaching the device specifications overview



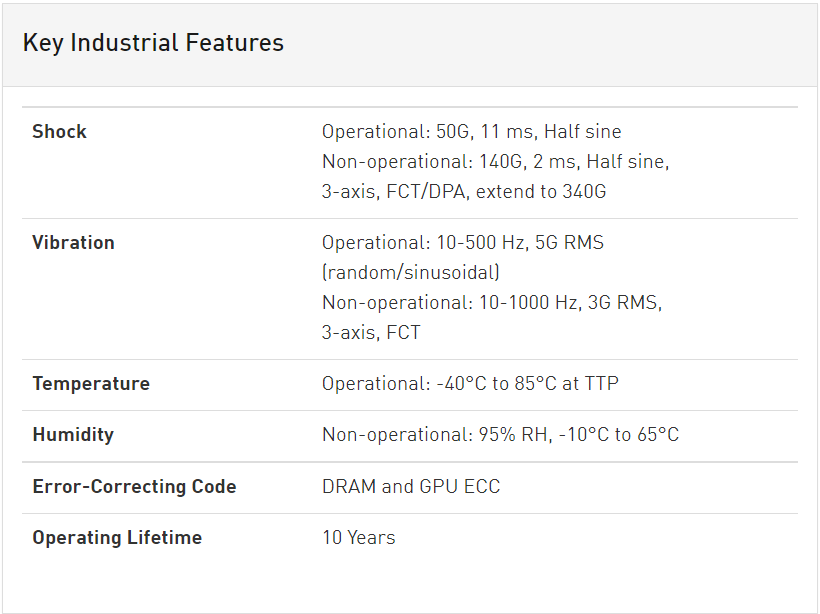
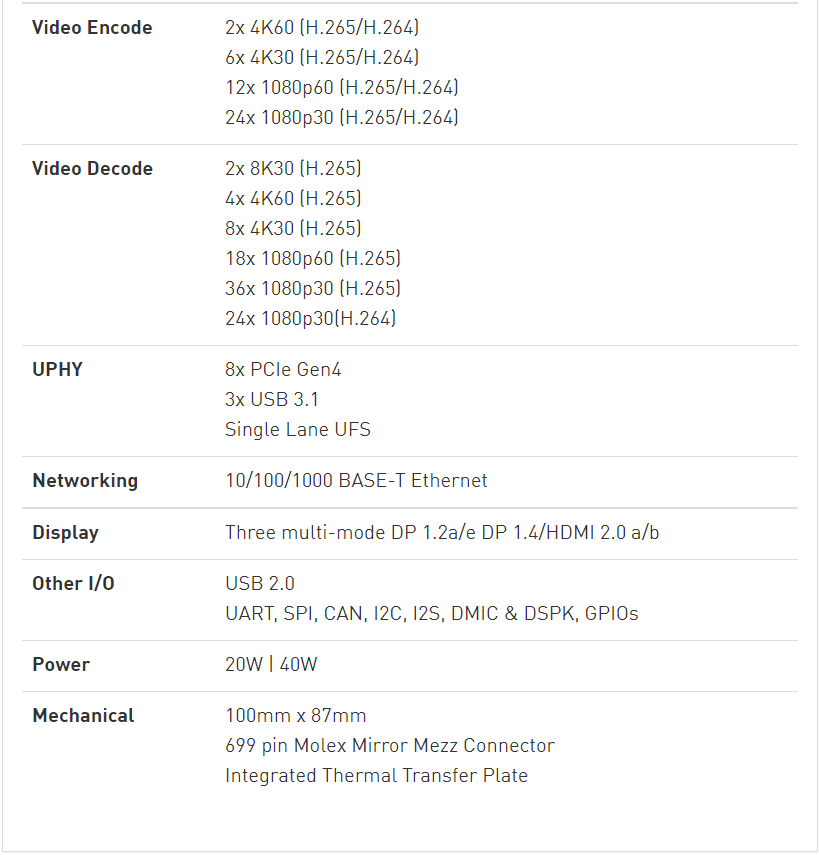
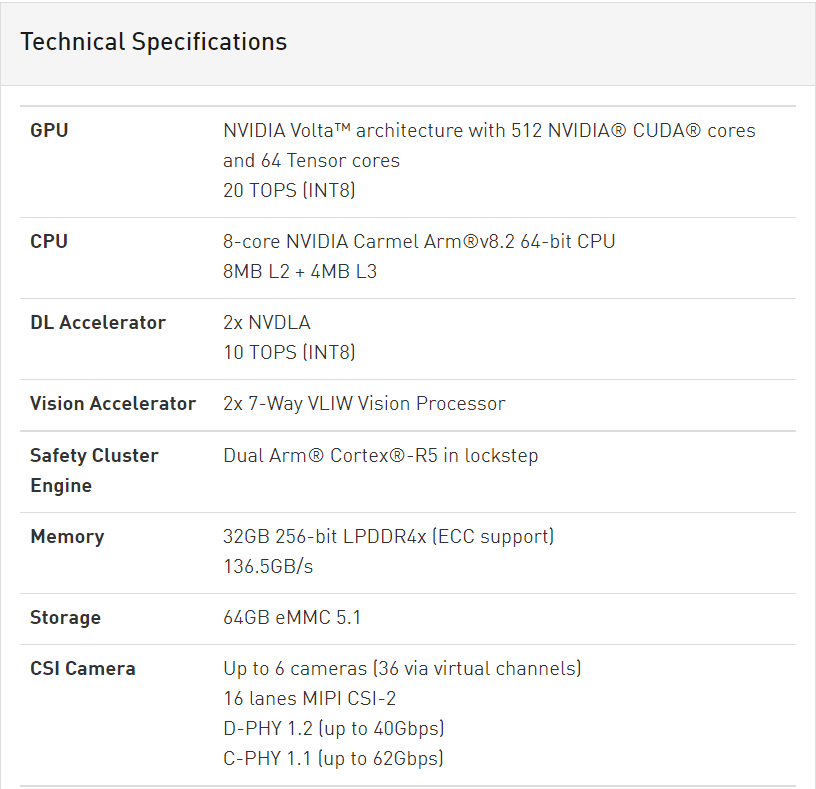


* + 1. Our approach was to run a Deep Learning model similar to we would run in production
    2. We were using [YOLOv5s](https://github.com/ultralytics/yolov5) object detection here. In the final code there would be more complexities as well as some higher order logic but the deep learning process would be the most compute intensive part of the project. So we tried to benchmark it.
    3. Once our deep learning model was ready we optimized it by pruning/sparsification and quantization using [Nvidia TensorRT](https://developer.nvidia.com/tensorrt).
    4. Following are the resource utilization when running YOLOv5s on Tx2

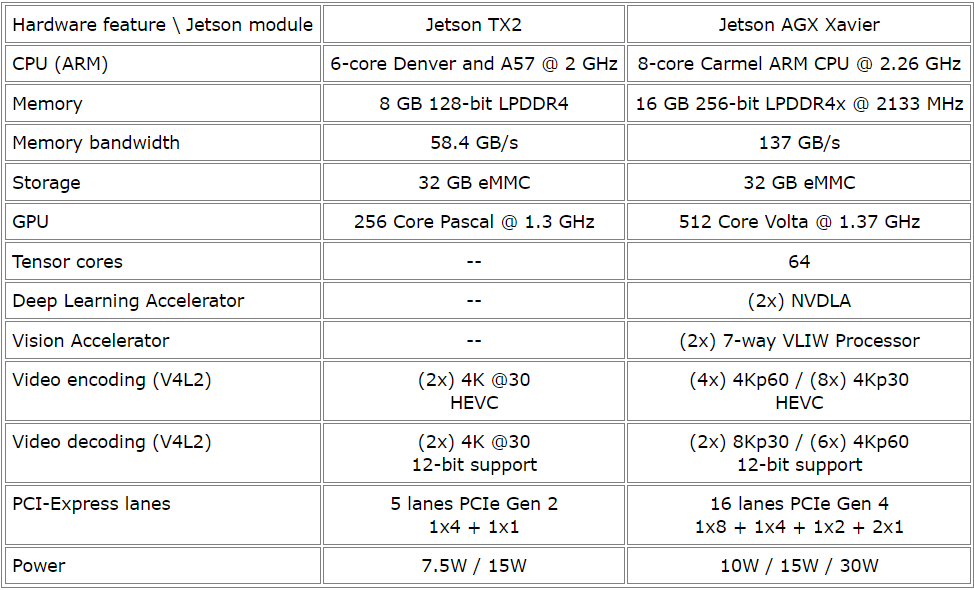




* + 1. Finally we after all the optimizations we were getting 37 FPS on Nvidia Jetson TX2.
    2. We figured we needed an embedded device with a more powerful GPU as that was the bottleneck for us in order to achieve 50FPS and remember in this we were getting data from saved video file whereas in actual condition we will be getting data from an IR camera live. Also, more code complexity would be added.
    3. So based on these we identified the following hardware: [Nvidia Jetson Xavier AGX Industrial](https://developer.nvidia.com/embedded/jetson-agx-xavier-i). We feel that this hardware has enough specification to take the load of our approach.
    4. Following are its specifications:



* + 1. Following is a broad comparison between our hardware we used for benchmarking (Nvidia Jetson Tx2) vs the hardware we feel would be able to solve our purpose (Nvidia Jetson Xavier AGX):



* + 1. So the objective of this phase was to find out the appropriate hardware that would solve the problem and we came up with one.

1. **Problems with existing system:**
   1. Now our model is running with good accuracy but we cannot deploy this system due to the following points
      1. Physical constraint (size):
         1. The size of our system should fit in either a 70 mm diameter PCB or a length could be extended to 115mm but width has to be in that 70mm range.
         2. Now, this Jetson AGX is of the size 100mm\*87mm and already very dense so there is hardly any scope for reduction of its size.
         3. Although, we can find out individual parts of this hardware like the GPU and CPU and build a custom PCB.
      2. General purpose vs specialized hardware:
         1. Now, the Jetson AGX is a general purpose hardware which is not built specifically for our purpose.
         2. A point has been raised by the hardware team that in such a case the reliability in terms of operation of the system is hard to determine.
         3. Also, the reliability in different environmental conditions is also hard to determine and require lots of test in different conditions.
      3. Booting time:
         1. The booting time of the complete system which is the time from which the system boots up, the program starts, the YOLOV5s model loads and inference starts on Jetson TX2 is around 10-15 seconds.
         2. Here, we do assume we will get a better boot up time of the coplete system with Jetson AGX but we still think it would be a very challenging task to optimize it to 2-3 seconds.
2. **Proposed system by the hardware team:**
   1. Proposal:
      1. Will find out the absolute resources required by current software. Amount of instructions sent to GPU per cycle for parallel processing, RAM consumed, CPU utilization as well as other key resources by individual layers of our deep learning model (YOLOV5s)
      2. Once we have the complete resource utilization will replace GPU with FPGAs and DSPs for parallel processing.
      3. Based on that a custom specialized hardware will be designed.
   2. Problems:
      1. Finding out the absolute requirements is a complex task and would require a lot of time whereas we are on a constrained timeline
      2. The amount of community support and frameworks/libraries available for Nvidia GPU support is way way more compared to FPGAs, where either we drop the idea of deep learning to drop FPGA or we have to reinvent the wheel by writing multiple custom codes/libraries for parallel processing where we lack both in terms of proficiency and time.
3. **Help required in following:**
   1. Designing a hardware which could run a YOLOv5s model or any deep learning model for that purpose.
   2. The hardware should abide by the physical constraints as well as it should be powerful enough to process the 50 FPS.
   3. If both these things are not possible we will have to drop the deep learning approach and shift to a pure image processing based system. But then solving the UAV detection problem with purely image processing robustly is not possible or would not fit in our time frame of developing the system in 6 months. We can also, process the complete deep learning on CPUs but that also requires a lot of engineering and would not be possible in time frame and is not guaranteed to work.
   4. Also, if a Nvidia based hardware doesn’t work, help is required in designing a FPGA based hardware for faster processing. Figuring out following related to FPGA:
      1. No. of I/Os
      2. Manufacturer
      3. Level of complexity
      4. Heatsinks, etc.
4. **If we go from an approach other than a Nvidia GPU based approach, what would be the impact.**
   1. The core reason behind adopting a Nvidia based system is that it has direct compatibility with high level deep learning frameworks like Pytorch (we are using this right now) or Tensorflow.
   2. It uses Cuda Toolkit and Cudnn which interacts with our framework to manage all the GPU cores and assign task to it rapidly.
   3. The system is extremely tightly bounded and works in real-time
   4. If we move to any other hardware for parallelization the major problem is we loose the compatibility and we have to design custom libraries which takes a lot of time.
   5. We have seen some implementations of FPGAs with deep learning but they are still in a very nascent stage.