Name: Ramprasad Anand Kulkarni (UAV-Intern)

Project report.

Project Name: Drone with Autonomous Flight capabilities, object detection and Drone-kit.

Components Used:

NO	Part Name	Quantity	Description and Specification
1)	F330 Drone Frame	1	Compact drone frame
2)	BLDC Motor	4	Ready To Sky 920KV DJI motors for Extended battery life
4)	4 in ESC 30A	1	Mamba 4 in one ESC
3)	Propellers	4(2CW,2CWW)	DJI Style Propellers 8045
4)	Pixhawk flight controller	1	2.4.8
5)	ESP8266	1	Wireless Telemetry
6)	Status LED and port extender Module	1	Extended Pixhawk USB port and status LED
7)	FS-IA6B Radio Receiver	1	Fly-sky Radio Receiver
8)	FS-i6 Radio Transmitter	1	Fly-sky Radio Transmitter
9)	Raspberry-pi 4B	1	On-Board SBC for video streaming and controlling Pixhawk with Drone-kit
10)	OLED-Screen	1	Displaying Pi IP address and vitals
11)	PI - CAM V2	1	8MP Sony IMX sensor camera.
12)	GPS with stand	1	M8N GPS
13)	I2C Splitter	1	Split I2C lanes from one to five
14)	Li-PO	1	Orange 3000mAh 30C 3S Battery to power the drone
15)	Li-ION With Battery Shield (BMS)	1	Orange 2500mAh 1S 8C Battery to power the raspberry-pi
16)	Pixhawk Power-module	1	Monitor Pixhawk Battery Status and Current Draw
17)	Drone Landing Gear	4	Used as landing Gear
18)	Cables	8	DF-13 and jumper cables

Project Description:

A1) Drone Frame and Drone building process (Previous version):

-Assembled F330 Frame with base plate and top plate used M3 screws for this purpose.

-Installed 2600KV Racing Drone motors on the frame with mamba MK2 mini flight controller and 30A 4 in 1 ESC with 1000mAh 2S 20C Orange Li-PO battery and 5-inch 5045 propellers.

-Installed radio receiver on the drone and configured it.

-Installed M3 standoffs above the Pixhawk flight controller.

-Installed Jetson nano with RPi Cam V2.

-Installed HC-05 Bluetooth.

-Installed GPS module.

-Installed FS-i6 receiver.

-Installed I2C splitter.

Result: High KV rating of the BLDC motors and small battery resulted in very low flight time, without barometer Flying the drone was very difficult.

A2) Drone Frame and Drone building process (Current Version)

-Replaced 2600KV motors with 920KV 2212 ready to sky motors.

-Replaced Mamba 405 with Pixhawk 2.4.8. configured it according to the needs with PID tunings.

-Replaced 1000mAh 2S 20C Orange Li-PO with 3000mAh 3s 30C orange Li-PO.

-Replaced Jeton Nano with Raspberry-pi and cooling-fan.

-Replaced Bluetooth with ESP-8266.

-Added an extra 18650 battery for Raspberry-PI.

-Installed OLED-Screen.

-Done PID tunings and other optimizations for stable flight.

Result: High flight time, automated flight capabilities, very high flight stability, LOITER mode support with raspberry PI.

B1) Flight Controller (Previous Version)

-Assembled drone with Mamba F405 MK2 mini fight controller and 4 in one ESC stack.

B2) flight Controller (Current Version):

-Replaced Mamba F405 MK2 mini flight controller with Pixhawk 2.4.8 flight controller.

Result: Open source Ardupilot flight controller with lot more feature than F405 ex. Open-source and fully configurable, MAVLink support and Telemetry 2 support.

C1) Telemetry (Previous Version):

-Installed and configured HC-05 Bluetooth.

Result: Very low bandwidth Bluetooth telemetry with low data speeds.

C2) Telemetry (Current version)

-Replaced and configured NodeMCU (ESP8266).

Result: Very High-Speed telemetry and drone Wi-Fi.

D1.1) Video streaming (Previous Version):

-Jetson nano was a powerful SBC it was able to stream the processed object detection streaming using G-streamer.

-After replacing Jetson nano with Raspberry-Pi Raspberry pi was producing 4 – 5 FPS output with OpenCV and caffe models to overcome this issue the Raspberry-Pi was only used for video streaming without any processing.

-Used Internet webcam streaming services to stream data over the internet and then receive it on the processing station(laptop).

Result: Very high latency (5 – 6 Sec.) with very low quality and bit-rate.

D1.2) Video streaming (Previous Version):

-Installed OpenCV with default package with all the dependencies like numpy and pyshine.

-Created a LAN with high bandwidth and low latency using home router with 2.4 GHz wi-fi.

-Installed dependencies for MJPEG video streaming server.

-Created a python Code to streaming server with raspberry-pi IP as its streaming IP with 720P MJPEG stream.

Result: Fairly low latency (2 – 3 Sec.) MJPEG streaming using LAN network with very frequent LAG in the stream as it was unoptimized.

D2) Video Streaming (Current Version):

-To increase the data transfer rates and reduce the latency switched from 2.4 GHz to 5 GHz wi-fi.

-Created a python code for video streaming server used default IP address of the raspberry-pi as video streaming

-To optimize the stream used OpenCV API to change the resolution, framerate, bitrate, Auto-exposure, orientation, Buffer-size, Autofocus, zoom, camera roll and other required features of the raspberry- Pi camera.

Result: very low latency 720p 60FPS well optimized video streaming server without lag and stuttering.

E1) Object Detection (Previous Version):

-Used Jetson nano(2GB) SBC as on bord flight controller, with jetson nano Drone was able to do on board processing and provided a 60FPS Object detection support and much more capabilities.

(Jetson nano stopped working at some point after connecting it the laptop so I had to switch to Raspberry-Pi)

-Replaced jetson nano with raspberry-Pi SBC as onboard companion computer.

-Installed RPi-Cam V2 on Raspberry pi.

-Installed OpenCV and all the dependencies on Raspberry-Pi

-Installed G-streamer with online video streaming with on raspberry-pi for video streaming.

Result: 60FPS object detection and powerful on-board computer (Jetson nano). (Switched to Raspberry-Pi) Raspberry-Pi added great community support with some major drawbacks like very low computational power and very low FPS in OpenCV Object Detection.

E2) Objection Detection (Current Version):

-As the Raspberry-Pi has very low computational power Decided to use laptop as a processing station for object detection.

-Installed second windows-10 OS as a final deployment OS.

-Installed all the dependencies like CMake, latest Nvidia Drivers, etc.

-Installed pip3, numpy, putty, OpenCV with OpenCV-contribute CUDA CUDNN world, CMake, Anaconda, Visual Studio, Visual Studio Code, Latest NVidia drivers and other required dependencies.

-Optimized OpenCV python code for the use of CUDA and DNN drivers.

Result: 60 FPS object detection with CUDA acceleration with latest Nvidia Drivers.