



NXP Drone accompanied by HAWK vision

Challenge2: Help drones help others



Team members: Peter Vrbancic, Saso Pavlic (-5°C with strong wind, extreme cold test passed ✓)



1) Please name every participant. Please mark those who are not NXP. Optionally note the roles of each of the participants.

Sašo Pavlič (NXP Freelancer student),
Peter Vrbančič (NXP Freelancer student).

2) Project title:

NXP drone accompanied by hawk vision (FPV camera on gimbal) with gesture control.

3) Please give a one-sentence description of your project

Change your vision from ground to sky. See more, feel more and increase your senses.

4) Highlight any interesting use of additional components

Navq - RRDRONE 8MMNavQ Linux companion computer platform with Vision for Mobile robotics based on NXP i.MX 8M Mini Soc with Google Coral Camera. The camera is taking frames as input into a program running on Navq, which detects if there is a human on a frame with the help of ML algorithm and follows him.

FPV Camera - gives us additional eyes in the sky. Used also for panoramic shoots.

2-Axis Gimbal - An Light 2 Axis Brushless camera gimbal with 8-bit BGC 3.0 controller used for stabilizing the Google Coral and FPV camera. The main function is to get clear recordings and better detection results. In addition, it adds 2 more dimensions, which are helping us to see more when it's necessary. With that our drone can hover on the spot and we are still able to move the camera left/right, up/down.

FPV Goggles - It helps us to see what the drone observes, this device was very useful and handy during debugging and building the final application. Goggles help the pilot to fly over the horizon without losing control.

14.8V 60C 5200 mAh Lipo Battery - After adding so much energy-consuming components it was time for the most powerful battery.

5) Please introduce your story: How does your drone help during pandemics?

The majority of times we think about drones as extended machines that can complete a variety of different tasks, which means they need to be self-operable. Another possible application, to use them as an extension of our own abilities which gives us a more detailed and comprehensive view and information above our horizon, in this case, we are the ones who control them. What use-cases can pop-out if we use both concepts?

The main idea of our project was the use of the drone as a self-flying robot that will follow us where we go (master-slave functionality). When a situation comes in which we want to control them, special hands mimic (gestures) will switch from the follow-me mode into the search mode. With this functionality (master-tool) drone will become our sixth sense and allow us to get the full awareness of the area.

Our principal goal and intentions are to create the most widely used resource-platform in Slovenia, which can be used for a variety of different use-cases:

- Pandemic (helps to locate big groups of people, the messenger of information)
- Natural disaster (identifying potentially dangerous POI)
- Search on difficult terrains (people, pets, objects)
- Hobbies (search for mushrooms, special vegetation, an overview of landscape)




- Fun :)

Summary (practical example): Drone follows us as we move. If we want to extend our vision (left/right 100 meters) we extend our hand in that direction, the drone will detect hands-mimic and initialize a pre-defined mission (object of interest) if that object is found it will notify the master and give him the possibility to observe it through FPV goggle, if not it will return to master and continue with follow-me mode.

Initial plan:

- To build a self-driving drone that will automatically detect a person and lock to it.
- Application for mobile device to enter parameters (object to the search, area, altitude)

Workflow:

1. detection of the master user  ♂
2. tracking a master-user in each direction (follow-me mode)
 - if the master-user is lost, return to the detection
3. extension of arm triggers a directional search of that area
 - fly to search area (100-150m away from the master user)
 - detection of the predefined custom object (pet, object, fire)
 - if an object is found, the master user will receive the notification with GPS coordinates of it
 - if not, return to master user
4. continue with follow-me mode



Final solution:

Workflow:

1. detection of the master-user (✓ **completed**)
2. tracking the master-user in each direction (follow-me mode) (✓ **completed**)
 - if the master-user is lost, return to the detection (✗)
3. extension of arm triggers = directional search of area (**the video was too shaky and Navq processing power is too low, it only detects extension of both arms, which we used as a circle path search**)
 - fly to search area (100-150m away from the master-user) (✓ **completed in the simulator, in reality - MAVLINK doesn't work always**)
 - detection of the predefined custom object (pet, object, fire) (✗)
 - if an object is found, notify the master user with GPS coordinates of it (✗)
4. if not, return to the master-user and continue with follow-me mode (✗)

Due to difficulties of object detection during flying, we decided to buy and attach a gimbal with FPV camera and goggles, which helped us with stabilization of the camera and a better view of what the drone sees and observes. **At the end we still used Google Coral camera, we only added FPV camera next to it for easier debugging (initialization of gimbal's angle,..It's easier to set up the device in the air if you actually see it 😊 in our case through FPV goggles).

Future steps:

1. Acquire Navq with i.MX 8M Plus processor and 16GB of RAM
2. Optimize and further develop Python script to control FMUK66
3. Connect computer/phone wirelessly to Navq for easier parameter setup and controlling instead of physical SSH
4. Wait that MAVSDK-Python is available for Aarch64-v8 or convert our Python detection script to C++ (MAVSDK for C++ is fully working)
5. Perform more test-flights
6. Develop the platform for the mobile device to enter parameters



Problems we were facing:

- **Software**

- Unable to use pyelQ library on NavQ using 8M Mini. After quite a bit of time, we discovered that our NavQ has only 4GB of RAM plus it was not designed to use pyelQ. (Solution): We used OpenCV2 and TensorFlow models to detect objects.
- MAVSDK - Python. We focused on developing in Python (more experienced). Later one when we wanted to test the same code on NavQ we discovered that the library is not supporting Aarch64v8 architecture ([link](#)). Solution: We made communication between Python (object detection) script and C++ (controlling drone).
- FPS rate was too slow for actual detection in the air with NavQ. (Solution): Instead of object detection (Caffe lib) on the frame by frame, we decided to detect it only on the first 30 frames and then switch to object tracking, which should be more power-efficient. In the end, we switch back to object detection with TensorFlow lib. Object tracking is not reliable when an object is lost.
- Difficulties while developing: It becomes annoying to always transport code from your PC to Navq. We used a nice [VS code plugin](#) to synchronize code via SFTP on LAN.

- **Hardware**

- Malfunctional RS 2212 920kV brushless motors. After trying the motors for the first time two were spinning but the other two were spinning and stopping at the same time(the correct screws were used). Got new ones from Amazon but one of them started smoking the other caught fire (Yes). We got the additional one from amazon that was not broken and some from NXP.
- Wrong PWM port connection from the FMUK66 to the ESC motor controller caused the drone to tip-off right after taking off. After posting the log review to PX4 Flight Review found out the cause.
- After synching the RC Transmitter with QGroundControl found out one of the 3-way switches was broken. After asking NXP contact for some switches so we could resolder them we just got a new RC transmitter(really happy about that solution)
- We had to resolder an XT60 connector to the battery cause the original had an T-connector(we could just use an adapter but couldn't wait anymore)
- We had to change the placement for different electronic parts on the drone, to have it more compact.
- After testing Google Coral Camera in action the problem was obvious that the drone was shaking to much and the picture was not even close to getting a stabile object detection
- After adding a 2-axis gimbal ordered from Bangood(the only option in that time) the FPV and Google Coral camera pictures were much better. But because we ordered it from an untrustworthy supplier the gimbal stopped working all of the sudden (3 days prior submission). The only option we had was to flash a new bootloader and firmware on the atmega328 chip that was controlling the gimbal motors. After finding the correct pins on the gimbal pcb board we were able to flash



a bootloader, some untrustworthy firmware we found, and surprisingly it was compatible.

- The Google Coral Camera 25 pin flat cable was too short, so we had to put the navq above the gimbal which changed the center of gravity of the whole drone. The solution was to place the battery more to the opposite side.
- The 3d printed antenna rails bracket broke.
- **Meteorological**
 - Shorten battery life because of too cold weather. During the flying test, the cold was a cause of many difficulties and problems with power supplies. Devices were turning off faster. It was very challenging for us to make videos. The battery of each device became less efficient. We had less time and even with the second battery.
- **Pilot's skills**
 - We started our project with zero flying skills to operate this drone. Our first tests finished with broken propellers. It took many hours to become more confident drone-users. We don't need to mention how many other parts we broke.

6) If available, please provide any schematics or links. Also share CAD files if created into your submission folder on teams. (Thingiverse?)

[Scheme](#)

7) Please share a link to your bitbucket repository or where your code can be found.

Source code added to TEAMS folder. BitBucket access was not available.

TEAMS folder: [link](#)

8) Link to your published videos or other online resourced you contributed to. (youtube, vimeo, git, thingiverse)

[\(You-Tube\) Hawk vision simulation in jMAVSim with SITL \(NavQ controlling PX4 via MAVLink\)](#)

[\(Hackster\) NXP drone accompanied by HAWK vision](#)

[\(You-Tube\) NXP drone accompanied by HAWK vision](#)



9) Collaboration

Because we know there is no real progress in humanity if there is no collaboration we are adding some information on how we helped others.

- [Active on GitHub for library MAVSDK-Python](#)
- [Saso Pavlic: Hello, I would need help with GStreamer. When starting Gstr...](#)
- [Saso Pavlic: Does someone know what it means error code \(E.11\) on attached charger Ho...](#)
- [Saso Pavlic: Using MAVSDK-Python for controlling drone with Navq Does anyone know...](#)
- [Saso Pavlic: Matthias Viertler thanks a lot for your explanation and simple guidelin...](#)
- [Saso Pavlic: Flashing imx-image-full-imx8mpevk.wic to SD card in order to use pyelQ](#)
- [Hackster project description](#)