Architecture Overview

ArduPilot Mega (APM)

The APM board can be seen as an interface to the peripheral devices installed in the sailing boat. It is connect via USB to the Jetson Nano and communication is handled via the MAVLink protocol.

This way the servo motors for the rudder and the sail can be set manually in `MANUAL` mode or set automatically by internal control loops which can be controlled by external parameters in `AUTOMATIC` mode.

Jetson Nano

The Jetson Nano acts as mastermind that gives sailing instructions to the APM board. The final goal of the project is, that the Jetson Nano is capable of:

- Detecting a specific object via the camera
- Sending the updated route instructions to the APM board which fires the motor servos.

Peripheral I/O devices

- Wind sensor that delivers readings **relative** to the direction of the sail
- Motor servos for controlling the rudder and the sail
- Camera for detecting objects on the water surface, to be used for controlling the sailing course

Hardware Overview

Stereo Camera

HBV-1780-2 Camera board with two OV9732 sensors and 72° field of view. Uses only one USB output (UVC protocol), which provides both video frames horizontally joined.

Product on AliExpress Image Sensor

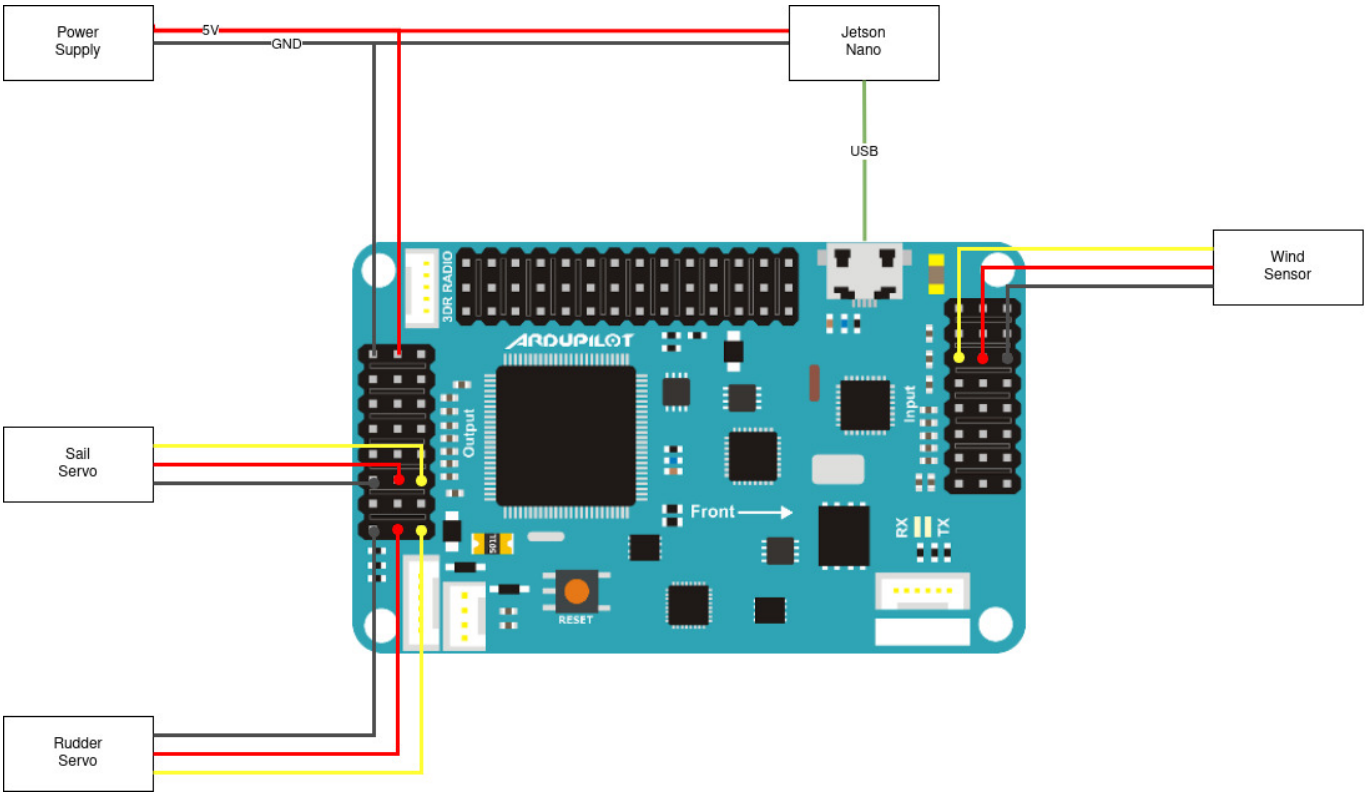
APM

Serial communication

The current APM project is configured to use a baudrate of 115200.

Wiring

Note: you need to supply external supply voltage to the servos connected to the output pins.



Inputs

- 6: Wind Sensor

Outputs

- 1: Rudder
- 3: Sail

Remote Control

- The RC channel columns 1, 2, 3 of the RC receiver have to be connected to the APM inputs 1, 2, 3.
- Connect **GND** and **VCC** from the **BAT** column of the RC receiver with **GND** and **VCC** of the APM board.
- The rows of the RC receiver consist of **GND**, **VCC** and **SIGNAL**, where **GND** is at the edge of the RC receiver board.

Software Overview

Getting started using the project from the repository

Here, it is expected that you are provided with a Jetson Nano as we left it behind in the summer of 2022.

It should be running:

- Ubuntu 20.04
- ROS Noetic Ninjemys

If this is not the case, please follow the `Getting started from scratch` instructions.

Connecting to the Jetson Nano via SSH

To initially check if the Jetson Nano is running as expected, we suggest to boot it up with a screen, keyboard and mouse connected.

The login details are:

```
user: nano
password: jetson
```

In order to be able to connect to the Jetson via SSH, you need to be connected to a non-restricted network unlike eduroam, which blocks SSH connections by default.

When working with the Jetson at TUHH, we used the following setup:

- A Wi-Fi connection to the TP-Link mini access point `(SSID: TP-Link_9C4A, password: 94588637)`
- A LAN connection to eduroam

Now, you should be able to SSH to the Jetson via the TP-Link access point, while having internet access via eduroam. You can look up the IP address of the Wi-Fi interface by running `ip a`

Setting up the ROS environment

The current project uses the Robot Operating System (ROS) framework.

Here, functionality is encapsulated in so-called *nodes* which communicate with each other via *topics*. Nodes can either send, i.e. *publish*, messages to topics or receive messages from topics by *subscribing* to them.

In order to have access to ROS commands, the following files should be sourced:

```
$ source /opt/ros/noetic/setup.bash # The global ROS installation
$ source ~/gitlab_repo/jetson_nano/catkin_ws/devel/setup.bash # The Catkin
workspace where all functionality is implemented
$ source
~/gitlab_repo/jetson_nano/catkin_ws_mavros/devel_isolated/setup.bash # The
Catkin workspace which is solely used for a custom MAVROS/MAVLINK version
```

Since these lines are at the end of the `~/.bashrc` file, it should happen automatically for every shell session.

Boat Control Package

Communication with the APM board is handled via MAVROS, the ROS implementation of the MAVLink protocol.

First, a single instance of the MAVROS master node must be launched. It offers a set of topics and services which can be used by other nodes to communicate the APM. **NOTE**: Since this calibrates the Inertial navigation system (INS), the boat must stand straight and still.

Launch it by running:

```
$ roslaunch mavros apm.launch fcu_url:=serial:///dev/ttyACM0:115200
fcu_protocol:=v1.0
```

or via our custom launch script:

```
$ roslaunch mavros master.launch
```

At the end of the console output, you should see `Ready to drive`.

Attachment - APM ready to drive

The next step is to calibrate the wind sensor and the motor servos for the rudder and sail.

Launch the calibration node by running

```
$ rosrun boat_control calibrate
```

The output of the calibration is a `.json` file at `/home/nano/gitlab_repo/jetson_nano/catkin_ws/src/boat_control/src/boat_control/calibration_data.json`

Extending the Boat Control Package

Currently, the Boat Control Package has only a single node called `calibrate`, however in the future you will have to implement further functionality like the sailing logic.

There is a directory for your python code in `catkin_ws/src/boat_control/src/boat_control`, where you can implement, for instance, your sailing node called `sail.py`. In order to be able to start this node, you must create a "python launch file" in `catkin_ws/src/boat_control/scripts`, e.g. a file called `sail`. Take a look at how we do it with the calibration node.

You could then launch the sailing node via `rosrun boat_control sail`.

apm.py

The apm.py file abstracts the MAVROS communication with the APM board and contains utility functions to:

- get and set APM parameters
- set APM Mode (`MANUAL`, `AUTOMATIC`)
- read and set RC channels aka servos and sensors

It can be imported in python files inside the `boat_control` package. See example usage in `calibrate.py`.

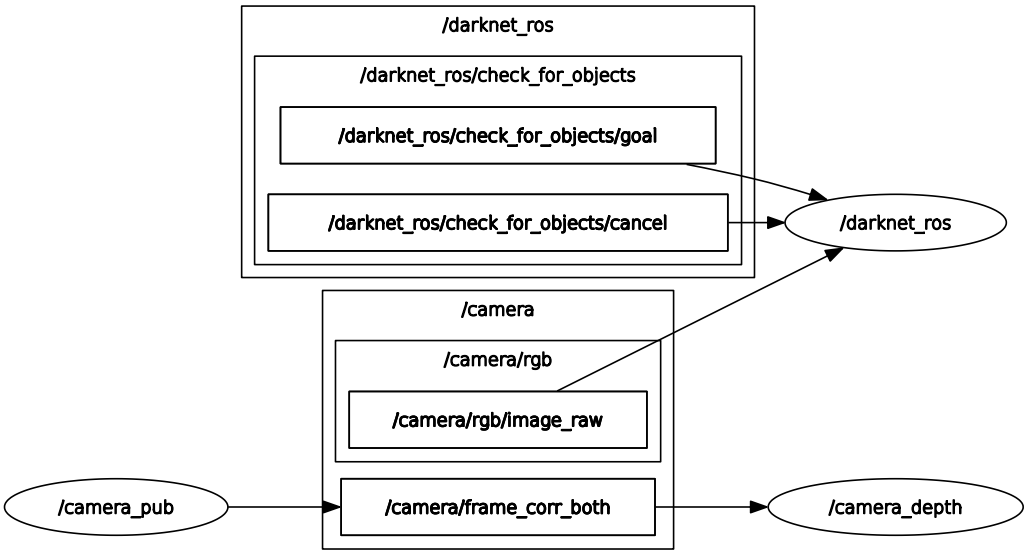
Camera Node

camera project ROS lanuchfiles

runnable by `roslaunch camera <launchfile>`

cam_corr_depth.launch

Runs the camera publisher node. Uses raw images for ROS darknet (YOLO) and the corrected frames for the depth map generation.



calibration.launch

Runs the calibration python script.

camera python scripts

Camera calibration (camera_calib.py)

Takes pictures from camera input if key `s` is pressed. After all images are taken, pressing `x` ends the image collection process. Afterwards a `stereoMap.xml` file is generated and saved. All selected images are saved in `/left/video_shot_{index}` and `/right/video_shot_{index}`. If multiple cameras are connected, change `camport` variable to desired port number. Its recommended to use at least 10 sample images of the calibration chessboard.

- Explanations and tutorial of the calibration process YouTube: Camera Calibration
- Used source code from niconielsen32: ComputerVision
- Mathematical details OpenCV: Camera Calibration

Camera correction *deprecated* (camera_corr.py)

Subscribes to *video_frames* ROS topic and corrects video feed by `stereoMap.xml` file. The video feed then is shown in its raw and corrected form split into its left and right parts.

Camera depthmap (camera_depth.py)

Subscribes to *camera/frame_corr_both* ROS topic and generates a depth map. The following 3 parameters can be changed while the depth map is shown:

- Number of disparities (from 1 to 17)
- Block size (from 5 to 50)
- Minimum number of disparities (from 5 to 25)
- More information about these maps at learnopencv: depth perception
- Mathematical details OpenCV: Depth Map

Camera publisher (camera_pub.py)

`camera_pub.py` reads an image from the camera, applies the correction via the `stereoMap.xml` file and publishes both frames (the corrected and original image) to the topics `/camera/frame_corr_both` and `/camera/frame_both`, respectively. The camera publisher can be launched by

```
$ roslaunch camera camera_py.launch
```

Additionally, the optional arguments

```
  sample_rate (default "10.0"): Frequency in Hz to publish images from the
camera
  verbose (default "true"): Verbose output
```

can be supplied for instance by

```
$ roslaunch camera camera_py.launch sample_rate:=0.5 verbose:=false
```

For debugging and other purposes the published images may be displayed with

```
$ rosrun image_view image_view image:=/camera/frame_both
```

or

```
$ rosrun image_view image_view image:=/camera/frame_corr_both
```

(Resizing the window may be necessary to see the entire image.)

When viewing the published image, note that the color channels may be swapped. This is intended behavior due to a quirk in YOLO.

YOLO mediator (yolo_mediator.py)

The mediator is situated between the camera publisher and the YOLO node. Based on user-specified launch arguments, it processes the image published by `camera_pub.py` to either the `/camera/frame_corr_both` or `/camera/frame_both` topic and publishes it to the `/camera/rgb/image_raw` topic at which point the YOLO node picks it up. The mediator can be launched by

```
$ roslaunch camera yolo_mediator.launch
```

Optional arguments can be supplied by:

```
$ roslaunch camera yolo_mediator.launch arg1:=value1 arg2:=value2
```

To modify the behavior of the mediator, the following arguments are available to the user:

```
  calib (default "true"): Use calibrated camera frame
  frame_slice (default "right"): [left, right] camera for object detection
  log_detection (default "false"): Print YOLO's detections with bounding
box to terminal
  scale_factor (default "1.0"): Factor to scale frame by
  thresh (default "0.3"): Probability threshold for log_detection in range
[0, 1]
  verbose (default "true"): Verbose output
```

In order to obtain the above output:

```
$ roslaunch camera yolo_mediator.launch --ros-args
```

The default values can be changed by modifying the launch file `isp-2022/jetson_nano/catkin_ws/src/camera/launch/yolo_mediator.launch`.

The `scale_factor` argument was implemented with the assumption that by varying the scaling of the image supplied to YOLO the speed/FPS of the detection is affected. This, however, does not appear to be

the case. The speed of the object detection seems to be independent of the input image's resolution since scaling the input image by 0.5 or even 0.25 leads to no change in speed. Hence, the default value of `1.0`.

YOLO - You Only Look Once - Object Detection

The YOLO node carries out the object detection and runs on the Darknet framework. The input images to the node are supplied by publishing images to the `/camera/rgb/image_raw` topic. The YOLO node can be launched by

```
$ roslaunch darknet_ros yolo_v3.launch
```

for detection with the YOLOv3 model or by

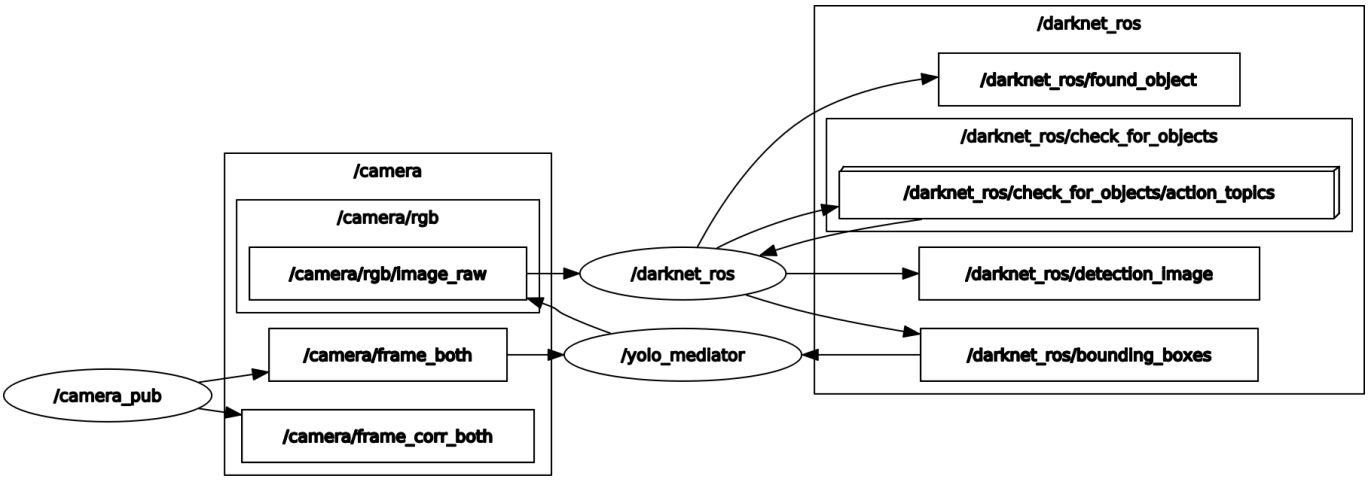
```
$ roslaunch darknet_ros yolo_v3_tiny.launch
```

for detection with the YOLOv3-tiny model.

To feed images to YOLO, the camera publisher and the mediator have to launched as well:

```
$ roslaunch camera camera_py.launch
$ roslaunch camera yolo_mediator.launch
```

The corresponding ROS graph may look like this:



(Image generated with the `rqt_graph` utility)

To ensure that the Jetson is running with the highest possible performance, set the Jetson to 10W (MAXN) performance mode:

```
$ sudo nvpmodel -m 0
$ sudo jetson_clocks
```

When accessing the Jetson over SSH the `-X` flag can be used `$ ssh -X nano@jetson-nano` to see detected object's bounding boxes.

Steps after cloning the repo for the first time

After cloning the repository for the very first time additional steps have to be taken to run the YOLO node.

Compiling

The Darknet binaries have to be compiled in order to utilize the Jetson's GPU. To this end, invoke the following commands:

```
$ cd isp-2022/jetson_nano/catkin_ws/src/darknet_ros/darknet/
$ make
```

(The required changes to the `Makefile` and further changes can be referred to by commit 1d35b94a).

Finally, build the ROS package in *Release* mode to increase performance:

```
$ cd isp-2022/jetson_nano/catkin_ws
$ ./release-build.sh
$ source devel/setup.bash
```

Downloading weights

Further, the weights of the YOLOv3 and YOLOv3-tiny models have to download. For this, see the file `how_to_download_weights.txt` in the directory `isp-2022/jetson_nano/catkin_ws/src/darknet_ros/darknet_ros/yolo_network_config/weights` and place the weight files in the same directory.

YOLO Darknet Official Website YOLO Darknet + ROS GitHub Repository

YOLO with TensorRT

Not in a working state! There exists an additional ROS package `isp-2022/jetson_nano/catkin_ws/src/yolov4_trt_ros` as a submodule. It is currently not runnable due to numerous version/depencency incompatabilities. The root cause of the issues appears to be the fact that we are running Ubuntu 20.04 while the CUDA and TensorRT packages are based on Ubuntu 18. However, the advantes of running YOLO with TensoRT are significant. According to this blog post, YOLOv3 runs with ~4 FPS on a Jetson Nano. A substantial increase over the current 1.3-1.5 FPS when running YOLO with Darknet. For further details see `isp-2022/jetson_nano/catkin_ws/src/yolov4_trt_ros/ISSUES.md`

YOEO - You Only Encode Once

Dependencies on ARCH64

- CMake needs to be 3.22 or newer. package *onnxoptimizer* needs cmake 3.22 or higher. Thus if needed a supported CMake Version is available in /YOEO/dep.

CMake Downloads

Installation:

```
./configure
make
sudo make install
```

`cmake --version` should now show version 3.24 (or the version which was installed)

- onxoptimizer 0.2.6 which is required in the default poetry.lock file is not available on ARCH64. Thus, version 0.3.0 was set in the changed .lock file.
- the same applies to onnxruntime-gpu (1.12.1 instead of 1.10.0). Of course a newer version can be tried out.
- If the user want to select a specific version himself, he also needs to add the sha265 hash of the setup archive.
- The old .lock file indicated by "_old" can be used to compare version differences.

Introduction

Instead of generating bouding boxes in YOLO, the approach of segmentation is persued in YOEO. This can be helpful to separate certain areas. A practical example would be the segmentation of water and sky.

YOEO GitHub repository

Publication

> **Abstract** Fast and accurate visual perception utilizing a robot's limited hardware resources is necessary for many mobile robot applications. We are presenting YOEO, a novel hybrid CNN which unifies previous object detection and semantic segmentation approaches using one shared encoder backbone to increase performance and accuracy. We show that it outperforms previous approaches on the TORSO-21 and Cityscapes datasets.

[ResearchGate] [Download]

```
@inproceedings{vahlyoeo,
  title={YOEO – You Only Encode Once: A CNN for Embedded Object Detection
and Semantic Segmentation},
  author={Vahl, Florian and Gutsche, Jan and Bestmann, Marc and Zhang,
Jianwei},
  year={2021},
  organization={IEEE},
  booktitle={2021 IEEE International Conference on Robotics and
Biomimetics (ROBIO)}
}
```

Getting started from scratch (behind the scenes)

You need a Jetson Nano running the following software:

- Ubuntu 20.04
- ROS Noetic Ninjemys
- MAVROS 0.32.0
- MAVLink 2020.2.2-1

Here we provide detailed step-by-step instructions on how we set up everything. This might be helpful for troubleshooting future problems.

Installing and configuring Ubuntu 20.04

Ubuntu 20.04 is required to run the latest ROS (ROS Noetic Ninjemys as of now). Unfortunately, the official Jetson Nano Developer Kit SD Card Image (4.6.1) uses Ubuntu 18.04 for the near future.

Hence, we use an unofficial image based on Xubuntu 20.04.

Instructions

Download sdcard image

https://github.com/Discombobulated88/Xubuntu-20.04-L4T-32.3.1/releases/download/v1.0/Xubuntu-20.04-l4t-r32.3.1.tar.tbz2

Flash image to sdcard

Extract the downloaded `.tar.tbz2` and flash the image to the sdcard using your favorite tool, e.g. balenaEtcher.

Boot from sdcard and follow instructions

Select keyboard layout, Connect to wifi-hotspot:

SSID: `TP-Link_9C4A` Passwort: `94588637`

and resize the app partition to the full size of the sdcard. Setup a user account. We chose:

user: `nano` password: `jetson` hostname: `jetson-nano`

Connect to the internet

Just plug in an ethernet cable providing Internet access

Upgrade the system packages

Since the custom sdcard image is a bit outdated, we should upgrade all packages first:

```
sudo apt update
sudo apt upgrade
```

There might be another process, i.e. a `unattended-ugrade` script, which blocks the update process. You can look it up by the PID which is provided in the error message:

```
ps aux | grep <PID>
```

You must first wait for it to finish or reboot/kill it, e.g.

```
kill <PID>
```

Setup SHH

```
sudo apt-get install openssh-server
sudo systemctl enable ssh
sudo systemctl start ssh
```

Get the Jetson's ip addr by running

```
ip address
```

Now you can connect via ssh from your machine:

```
ssh <username>@<ipaddress>
```

Using X11 forwarding

```
ssh <username>@<ipaddress> -X
```

Installing and setting up ROS Noetic Ninjemys

Follow instructions at https://wiki.ros.org/noetic/Installation/Ubuntu unitl **2. Tutorials** (select **Desktop-Full install**).

Install mavros

We install a custom MAVLINK/MAVROS version.

Geopgraphiclib must still be installed:

```
sudo apt-get install libgeographic-dev
sudo apt-get install geographiclib-tools
```

Run

```
wget
https://raw.githubusercontent.com/mavlink/mavros/master/mavros/scripts/ins
tall_geographiclib_datasets.sh
sudo bash ./install_geographiclib_datasets.sh
```

In order to have the permissions to access the APM device at `/dev/ttyACM0`, we need to add our user to the `dialout` group and logout/login:

```
sudo adduser $USER dialout
```

Creating Catkin workspace

First of all we have to establish a connection to GitLab in order to clone our repository. Inside the repo we added a root folder `jetson_nano` with a subfolder called `catkin_ws`.

This folder will be symlinked to the home folder, so that we can access the catkin workspace via

```
cd ~/catkin_ws
```

Install CUDA

Installing CUDA is necessary for accelerating the object detection via YOLO. It is advisable to use CUDA 10.0 to avoid dependency and version incompatibility issues. To install CUDA 10.0:

```
$ sudo apt install \
 cuda-command-line-tools-10-0 \
 cuda-compiler-10-0 \
 cuda-curand-dev-10-0 \
 cuda-cublas-dev-10-0 \
 libcudnn7-dev \
 gcc-7 \
 g++-7
```

The older versions of `gcc` and `g++` are required by the CUDA compiler `nvcc`.

Add CUDA to `$PATH` by adding

```
export PATH=/usr/local/cuda-10.0/bin${PATH:+:${PATH}}
```

to `~/.bashrc` and `$ source ~/.bashrc`.

Install OpenCV

Installation

```
$ sudo apt install ros-noetic-vision-opencv
$ sudo apt install ros-noetic-cv-bridge
```

To build OpenCV with CUDA support, look to the directory `isp-2022/jetson_nano/build_opencv`. Further, this blog post and the resources it links to may be of help.

Note that compiling OpenCV from source on the Jetson Nano takes several hours. Hence, it is advisable to start the build job in a `tmux` session from which detaching without stopping the job is possible.

See this link for various camera configuration options that can be set by via OpenCV

```
cap = cv2.VideoCapture(cv2.CAP_ANY)
cap.set(<index>, <value>)
```

Install YOEO

```
git clone https://github.com/bit-bots/YOEO
cd YOEO/
pip3 install poetry --user
PATH=$PATH:/home/parallels/.local/bin
poetry install
```

Facing Problems with MAVLink

Initially, we had issues to establish the MAVLink connection with the APM board. The main issue is that the APM board uses an old firmware that is not compatible with recent MAVLink implementations.

Install compatible MAVROS and MAVLink versions:

Inspired by this GitHub issue we tried the following mavros/mavlink versions:

```
mavros: release 0.32.0
mavlink: release 2020.2.2-1
```

Following the source installation instructions we came up with this `.rosinstall` file:

```
- git:
    local-name: mavlink
    uri: https://github.com/mavlink/mavlink-gbp-release.git
    version: release/melodic/mavlink/2020.2.2-1

- git:
    local-name: mavros
    uri: https://github.com/mavlink/mavros.git
    version: 0.32.0
```

Note: The installation instructions are for an older ROS version. We have to replace `catkin build` with `catkin_make_isolated`. However, running `catkin_make_isolated` showed some errors that had to be fixed first.

Missing python2 package

`future` for python2 was missing, i.e. `python2 -m pip install future` had to be run

A now deprecated boost function was used

To compile, `libboost1.67` was required for the call to `Socket::get_io_service()` to work, but ROS noetic depends on version 1.71. Thus, a downgrade was not an option.

As a workaround, we replaced the calls to the deprecated function `Socket::get_io_service()` in the source file `catkin_ws_mavros/src/mavros/libmavconn/src/tcp.cpp` according to this stackoverflow answer.

Multiple installations of mavros/mavlink

Since we installed mavros through the debian package manager previously, we have to remove these packages to avoid conflicts.

```
sudo apt-get purge ros-noetix-mavros ros-noetic-mavros-extras ros-noetic-
mavlink
```

Testing MAVlink communication

Now we can source the isolated development environment

```
nano@jetson-nano:~/catkin_ws_mavros$ source devel_isolated/setup.bash
```

and connect to the APM board with

```
roslaunch mavros apm.launch fcu_url:=serial:///dev/ttyACM0:115200
fcu_protocol:=v1.0
```

You should see `Ready to drive` at some point, indicating that the connection was successfull.

Now you can **set parameters**, e.g.

```
nano@jetson-
nano:~/gitlab_repo/jetson_nano/catkin_ws_mavros/devel_isolated/mavros/lib/
mavros$ rosrun mavros mavparam set RUDDER_MIN 100
100
```

and **get parameters**, e.g.

```
nano@jetson-
nano:~/gitlab_repo/jetson_nano/catkin_ws_mavros/devel_isolated/mavros/lib/
mavros$ rosrun mavros mavparam get RUDDER_MIN
100
```

Attachments

APM ready to drive

```
    ... logging to /home/nano/.ros/log/3e774a90-17e6-11ed-bbff-
e5481d40a7d3  /roslaunch-jetson-nano-7926.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://jetson-nano:37083/

SUMMARY
========

CLEAR PARAMETERS
 * /mavros/

PARAMETERS
 * /mavros/cmd/use_comp_id_system_control: False
 * /mavros/conn/heartbeat_mav_type: ONBOARD_CONTROLLER
 * /mavros/conn/heartbeat_rate: 1.0
 * /mavros/conn/system_time_rate: 1.0
```

```
 * /mavros/conn/timeout: 10.0
 * /mavros/conn/timesync_rate: 10.0
 * /mavros/distance_sensor/rangefinder_pub/field_of_view: 0.0
 * /mavros/distance_sensor/rangefinder_pub/frame_id: lidar
 * /mavros/distance_sensor/rangefinder_pub/id: 0
 * /mavros/distance_sensor/rangefinder_pub/send_tf: False
 * /mavros/distance_sensor/rangefinder_pub/sensor_position/x: 0.0
 * /mavros/distance_sensor/rangefinder_pub/sensor_position/y: 0.0
 * /mavros/distance_sensor/rangefinder_pub/sensor_position/z: -0.1
 * /mavros/distance_sensor/rangefinder_sub/id: 1
 * /mavros/distance_sensor/rangefinder_sub/orientation: PITCH_270
 * /mavros/distance_sensor/rangefinder_sub/subscriber: True
 * /mavros/fake_gps/eph: 2.0
 * /mavros/fake_gps/epv: 2.0
 * /mavros/fake_gps/fix_type: 3
 * /mavros/fake_gps/geo_origin/alt: 408.0
 * /mavros/fake_gps/geo_origin/lat: 47.3667
 * /mavros/fake_gps/geo_origin/lon: 8.55
 * /mavros/fake_gps/gps_rate: 5.0
 * /mavros/fake_gps/mocap_transform: True
 * /mavros/fake_gps/satellites_visible: 5
 * /mavros/fake_gps/tf/child_frame_id: fix
 * /mavros/fake_gps/tf/frame_id: map
 * /mavros/fake_gps/tf/listen: False
 * /mavros/fake_gps/tf/rate_limit: 10.0
 * /mavros/fake_gps/tf/send: False
 * /mavros/fake_gps/use_mocap: True
 * /mavros/fake_gps/use_vision: False
 * /mavros/fcu_protocol: v1.0
 * /mavros/fcu_url: serial:///dev/tty...
 * /mavros/gcs_url:
 * /mavros/global_position/child_frame_id: base_link
 * /mavros/global_position/frame_id: map
 * /mavros/global_position/gps_uere: 1.0
 * /mavros/global_position/rot_covariance: 99999.0
 * /mavros/global_position/tf/child_frame_id: base_link
 * /mavros/global_position/tf/frame_id: map
 * /mavros/global_position/tf/global_frame_id: earth
 * /mavros/global_position/tf/send: False
 * /mavros/global_position/use_relative_alt: True
 * /mavros/image/frame_id: px4flow
 * /mavros/imu/angular_velocity_stdev: 0.0003490659 // 0...
 * /mavros/imu/frame_id: base_link
 * /mavros/imu/linear_acceleration_stdev: 0.0003
 * /mavros/imu/magnetic_stdev: 0.0
 * /mavros/imu/orientation_stdev: 1.0
 * /mavros/landing_target/camera/fov_x: 2.0071286398
 * /mavros/landing_target/camera/fov_y: 2.0071286398
 * /mavros/landing_target/image/height: 480
 * /mavros/landing_target/image/width: 640
 * /mavros/landing_target/land_target_type: VISION_FIDUCIAL
 * /mavros/landing_target/listen_lt: False
 * /mavros/landing_target/mav_frame: LOCAL_NED
 * /mavros/landing_target/target_size/x: 0.3
```

```
 * /mavros/landing_target/target_size/y: 0.3
 * /mavros/landing_target/tf/child_frame_id: camera_center
 * /mavros/landing_target/tf/frame_id: landing_target
 * /mavros/landing_target/tf/listen: False
 * /mavros/landing_target/tf/rate_limit: 10.0
 * /mavros/landing_target/tf/send: True
 * /mavros/local_position/frame_id: map
 * /mavros/local_position/tf/child_frame_id: base_link
 * /mavros/local_position/tf/frame_id: map
 * /mavros/local_position/tf/send: False
 * /mavros/local_position/tf/send_fcu: False
 * /mavros/mission/pull_after_gcs: True
 * /mavros/mocap/use_pose: True
 * /mavros/mocap/use_tf: False
 * /mavros/odometry/estimator_type: 3
 * /mavros/odometry/frame_tf/desired_frame: ned
 * /mavros/plugin_blacklist: [actuator_contro...
 * /mavros/plugin_whitelist: []
 * /mavros/px4flow/frame_id: px4flow
 * /mavros/px4flow/ranger_fov: 0.118682
 * /mavros/px4flow/ranger_max_range: 5.0
 * /mavros/px4flow/ranger_min_range: 0.3
 * /mavros/safety_area/p1/x: 1.0
 * /mavros/safety_area/p1/y: 1.0
 * /mavros/safety_area/p1/z: 1.0
 * /mavros/safety_area/p2/x: -1.0
 * /mavros/safety_area/p2/y: -1.0
 * /mavros/safety_area/p2/z: -1.0
 * /mavros/setpoint_accel/send_force: False
 * /mavros/setpoint_attitude/reverse_thrust: False
 * /mavros/setpoint_attitude/tf/child_frame_id: target_attitude
 * /mavros/setpoint_attitude/tf/frame_id: map
 * /mavros/setpoint_attitude/tf/listen: False
 * /mavros/setpoint_attitude/tf/rate_limit: 50.0
 * /mavros/setpoint_attitude/use_quaternion: False
 * /mavros/setpoint_position/mav_frame: LOCAL_NED
 * /mavros/setpoint_position/tf/child_frame_id: target_position
 * /mavros/setpoint_position/tf/frame_id: map
 * /mavros/setpoint_position/tf/listen: False
 * /mavros/setpoint_position/tf/rate_limit: 50.0
 * /mavros/setpoint_raw/thrust_scaling: 1.0
 * /mavros/setpoint_velocity/mav_frame: LOCAL_NED
 * /mavros/startup_px4_usb_quirk: False
 * /mavros/sys/disable_diag: False
 * /mavros/sys/min_voltage: 10.0
 * /mavros/target_component_id: 1
 * /mavros/target_system_id: 1
 * /mavros/tdr_radio/low_rssi: 40
 * /mavros/time/time_ref_source: fcu
 * /mavros/time/timesync_avg_alpha: 0.6
 * /mavros/time/timesync_mode: MAVLINK
 * /mavros/vibration/frame_id: base_link
 * /mavros/vision_pose/tf/child_frame_id: vision_estimate
 * /mavros/vision_pose/tf/frame_id: map
```

```
 * /mavros/vision_pose/tf/listen: False
 * /mavros/vision_pose/tf/rate_limit: 10.0
 * /mavros/vision_speed/listen_twist: True
 * /mavros/vision_speed/twist_cov: True
 * /mavros/wheel_odometry/child_frame_id: base_link
 * /mavros/wheel_odometry/count: 2
 * /mavros/wheel_odometry/frame_id: map
 * /mavros/wheel_odometry/send_raw: True
 * /mavros/wheel_odometry/send_twist: False
 * /mavros/wheel_odometry/tf/child_frame_id: base_link
 * /mavros/wheel_odometry/tf/frame_id: map
 * /mavros/wheel_odometry/tf/send: True
 * /mavros/wheel_odometry/use_rpm: False
 * /mavros/wheel_odometry/vel_error: 0.1
 * /mavros/wheel_odometry/wheel0/radius: 0.05
 * /mavros/wheel_odometry/wheel0/x: 0.0
 * /mavros/wheel_odometry/wheel0/y: -0.15
 * /mavros/wheel_odometry/wheel1/radius: 0.05
 * /mavros/wheel_odometry/wheel1/x: 0.0
 * /mavros/wheel_odometry/wheel1/y: 0.15
 * /rosdistro: noetic
 * /rosversion: 1.15.14

NODES
  /
    mavros (mavros/mavros_node)

auto-starting new master
process[master]: started with pid [7934]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 3e774a90-17e6-11ed-bbff-e5481d40a7d3
process[rosout-1]: started with pid [7946]
started core service [/rosout]
process[mavros-2]: started with pid [7964]
[ INFO] [1660051319.821902237]: FCU URL: serial:///dev/ttyACM0:115200
[ INFO] [1660051319.830922456]: serial0: device: /dev/ttyACM0 @ 115200 bps
[ INFO] [1660051319.831991591]: GCS bridge disabled
[ INFO] [1660051319.868104136]: Plugin 3dr_radio loaded
[ INFO] [1660051319.874696394]: Plugin 3dr_radio initialized
[ INFO] [1660051319.874804573]: Plugin actuator_control blacklisted
[ INFO] [1660051319.885851393]: Plugin adsb loaded
[ INFO] [1660051319.898116881]: Plugin adsb initialized
[ INFO] [1660051319.898216675]: Plugin altitude blacklisted
[ INFO] [1660051319.898606734]: Plugin cam_imu_sync loaded
[ INFO] [1660051319.901147615]: Plugin cam_imu_sync initialized
[ INFO] [1660051319.901605748]: Plugin command loaded
[ INFO] [1660051319.923383236]: Plugin command initialized
[ INFO] [1660051319.923831473]: Plugin companion_process_status loaded
[ INFO] [1660051319.933357900]: Plugin companion_process_status
initialized
[ INFO] [1660051319.933461027]: Plugin debug_value blacklisted
[ INFO] [1660051319.933927651]: Plugin distance_sensor loaded
[ INFO] [1660051319.966066633]: Plugin distance_sensor initialized
```

```
[ INFO] [1660051319.966622789]: Plugin fake_gps loaded
[ INFO] [1660051320.016780023]: Plugin fake_gps initialized
[ INFO] [1660051320.016893306]: Plugin ftp blacklisted
[ INFO] [1660051320.017292376]: Plugin global_position loaded
[ INFO] [1660051320.074467240]: Plugin global_position initialized
[ INFO] [1660051320.075000374]: Plugin gps_rtk loaded
[ INFO] [1660051320.084409664]: Plugin gps_rtk initialized
[ INFO] [1660051320.084517687]: Plugin hil blacklisted
[ INFO] [1660051320.084972955]: Plugin home_position loaded
[ INFO] [1660051320.098885870]: Plugin home_position initialized
[ INFO] [1660051320.099324576]: Plugin imu loaded
[ INFO] [1660051320.127881825]: Plugin imu initialized
[ INFO] [1660051320.128385168]: Plugin landing_target loaded
[ INFO] [1660051320.179948261]: Plugin landing_target initialized
[ INFO] [1660051320.180383737]: Plugin local_position loaded
[ INFO] [1660051320.204632521]: Plugin local_position initialized
[ INFO] [1660051320.205189094]: Plugin log_transfer loaded
[ INFO] [1660051320.217138795]: Plugin log_transfer initialized
[ INFO] [1660051320.217577709]: Plugin manual_control loaded
[ INFO] [1660051320.229106413]: Plugin manual_control initialized
[ INFO] [1660051320.229595797]: Plugin mocap_pose_estimate loaded
[ INFO] [1660051320.243718663]: Plugin mocap_pose_estimate initialized
[ INFO] [1660051320.244145650]: Plugin mount_control loaded
[ INFO] [1660051320.253465771]: Plugin mount_control initialized
[ INFO] [1660051320.253946041]: Plugin obstacle_distance loaded
[ INFO] [1660051320.263265120]: Plugin obstacle_distance initialized
[ INFO] [1660051320.263741587]: Plugin odom loaded
[ INFO] [1660051320.284322178]: Plugin odom initialized
[ INFO] [1660051320.284946616]: Plugin param loaded
[ INFO] [1660051320.297347211]: Plugin param initialized
[ INFO] [1660051320.297469766]: Plugin px4flow blacklisted
[ INFO] [1660051320.298003734]: Plugin rangefinder loaded
[ INFO] [1660051320.300770243]: Plugin rangefinder initialized
[ INFO] [1660051320.301282440]: Plugin rc_io loaded
[ INFO] [1660051320.315254106]: Plugin rc_io initialized
[ INFO] [1660051320.315365671]: Plugin safety_area blacklisted
[ INFO] [1660051320.315761043]: Plugin setpoint_accel loaded
[ INFO] [1660051320.327766578]: Plugin setpoint_accel initialized
[ INFO] [1660051320.328467320]: Plugin setpoint_attitude loaded
[ INFO] [1660051320.366057081]: Plugin setpoint_attitude initialized
[ INFO] [1660051320.366531464]: Plugin setpoint_position loaded
[ INFO] [1660051320.415987279]: Plugin setpoint_position initialized
[ INFO] [1660051320.416424943]: Plugin setpoint_raw loaded
[ INFO] [1660051320.449987962]: Plugin setpoint_raw initialized
[ INFO] [1660051320.450517034]: Plugin setpoint_velocity loaded
[ INFO] [1660051320.472557756]: Plugin setpoint_velocity initialized
[ INFO] [1660051320.473350323]: Plugin sys_status loaded
[ INFO] [1660051320.515820591]: Plugin sys_status initialized
[ INFO] [1660051320.516317579]: Plugin sys_time loaded
[ INFO] [1660051320.537405523]: TM: Timesync mode: MAVLINK
[ INFO] [1660051320.541956493]: Plugin sys_time initialized
[ INFO] [1660051320.542435824]: Plugin trajectory loaded
[ INFO] [1660051320.562944747]: Plugin trajectory initialized
[ INFO] [1660051320.563420068]: Plugin vfr_hud loaded
```

```
[ INFO] [1660051320.565982355]: Plugin vfr_hud initialized
[ INFO] [1660051320.566114076]: Plugin vibration blacklisted
[ INFO] [1660051320.566535855]: Plugin vision_pose_estimate loaded
[ INFO] [1660051320.595289149]: Plugin vision_pose_estimate initialized
[ INFO] [1660051320.595418996]: Plugin vision_speed_estimate blacklisted
[ INFO] [1660051320.595853691]: Plugin waypoint loaded
[ INFO] [1660051320.613431778]: Plugin waypoint initialized
[ INFO] [1660051320.613561780]: Plugin wheel_odometry blacklisted
[ INFO] [1660051320.613951579]: Plugin wind_estimation loaded
[ INFO] [1660051320.617368205]: Plugin wind_estimation initialized
[ INFO] [1660051320.617639564]: Built-in SIMD instructions: ARM NEON
[ INFO] [1660051320.617821078]: Built-in MAVLink package version:
[ INFO] [1660051320.617928059]: Known MAVLink dialects: common
ardupilotmega ASLUAV autoquad icarous matrixpilot paparazzi slugs standard
uAvionix ualberta
[ INFO] [1660051320.618007332]: MAVROS started. MY ID 1.240, TARGET ID 1.1
[ERROR] [1660051321.543604567]: FCU: Calibrating barometer
[ INFO] [1660051321.545143137]: CON: Got HEARTBEAT, connected. FCU:
ArduPilot
[ERROR] [1660051322.595141531]: FCU: command received:
[ INFO] [1660051322.600939295]: VER: 1.1: Capabilities
0x0000000000000000
[ INFO] [1660051322.603522780]: VER: 1.1: Flight software:     00000080
(e43baa2c)
[ INFO] [1660051322.605092132]: VER: 1.1: Middleware software: 00000000 (
)
[ INFO] [1660051322.606112880]: VER: 1.1: OS software:         00000000 (
)
[ INFO] [1660051322.606480439]: VER: 1.1: Board hardware:      00000000
[ INFO] [1660051322.606892739]: VER: 1.1: VID/PID:             0000:0000
[ INFO] [1660051322.607725775]: VER: 1.1: UID:
0000000000000000
[ WARN] [1660051322.608780586]: CMD: Unexpected command 520, result 0
[ERROR] [1660051322.912485232]: FCU: Warming up ADC...
[ERROR] [1660051323.415141225]: FCU: Beginning INS calibration; do not
move vehicle
[ERROR] [1660051323.542423951]: FCU: <startup_ground> GROUND START
[ERROR] [1660051325.556971315]: FCU: Initialising APM...
[ERROR] [1660051326.659895304]: FCU:

 Ready to drive.
```

Credentials

Laptop User name: `projekt` Password: `mteclabor2022`

Laptop VM User name: `developer` Password: `password`

Jetson Ubuntu 20.04 User name: `nano` Password: `jetson` Mac-Address: `28:D0:EA:88:52:DE`

WiFi SSID: `TP-Link_9C4A` Password: `94588637`

WiFi Admin Panel IP: `192.168.0.1` Password: `mtecsail2022`