Building an autonomous quadrotor using a Parrot Bebop structure, ArduCopter and a Raspberry Pi Zero W

Cours correspondant : Robotique pratique (http://www.ensta-bretagne.fr/lebars/robotique_pratique.pdf).

Rendre sous Moodle dans un fichier de la forme DATE_NOM1_NOM2_..._NOMN.zip tous les documents utiles permettant de montrer le travail effectué : codes, fichiers de config, logs, données brutes de capteurs, infos précises sur les versions logicielles et matérielles utilisées, schémas en CAO ou manuels, photos et vidéos à différentes étapes, etc.

Abbreviations:

pi0w: Raspberry Pi Zero W
px: Pixhawk autopilot or other ArduCopter-compatible
px4f: PX4Flow optical flow sensor
cx-of: Cheerson optical flow sensor
Tx: Transmitter
Rx: Receiver
RC: Radio Control

Overview

The purpose of this activity is to build an autonomous quadrotor able to avoid obstacles around it using the mechanical parts of a standard Parrot Bebop drone, an autopilot using ArduCopter firmware and a Raspberry Pi Zero to get the data of ultrasonic telemeters and send them to the autopilot. Depending on the parts available, some options can be added or removed to e.g. improve horizontal stability with optical flow, vertical stability using LIDAR, etc. Most of the necessary algorithms are already provided by ArduCopter, the main work will be to assemble the hardware and configure the software consistently.

Parts 1, 2, 3 can more or less be prepared in parallel, then 4, 5, 6 also…

Shopping list/Bill of materials:

Bebop central structure: https://www.amazon.fr/Parrot-PF070076-Centrale-contr%C3%B4l%C3%A9-Smartphone/dp/B00R105G60/
Bebop motors: https://www.amazon.fr/Parrot-PF070079-Moteur-contr%C3%B4l%C3%A9-Smartphone/dp/B00R10CW5I/
4 connectors compatible with the motors (optional, cables can be just cut and soldered): https://fr.rs-online.com/web/p/embases-de-circuit-imprime/8201431/
Bebop propellers: https://www.amazon.fr/Parrot-PF070078-H%C3%A9lice-BeBop-Drone/dp/B00R10GUOC/
Bebop landing gear (optional): https://www.amazon.fr/Parrot-PF070112-PF070082AA/dp/B00QNE2REE/

Bebop side protections (optional, recommended to be able to install easily the telemeters for obstacle avoidance): https://www.amazon.fr/Parrot-PF070077-Car%C3%A8ne-externe-BeBop/dp/B00R10AFGG/
Bebop tools (optional, mandatory if you do not have the correct screwdrivers): https://www.amazon.fr/Parrot-PF070234AA-Bo%C3%AEte-%C3%A0-Outils/dp/B06XFXJRF2/
Servo tester (optional, to test easily the correct rotating direction for the motors depending on the propeller) : https://www.amazon.fr/TOOGOO-4-8-6V-Appareil-barre-Bleu/dp/B00HE5BYXQ
4 ESC 3S at least 10 A (for each): https://www.flashrc.com/dualsky/8462-controleur_brushless_22a_xc_22_lite_dualsky.html (tested), https://hobbyking.com/fr_fr/afro-30a-race-spec-mini-esc-w-bec.html (tested only on the ground), https://hobbyking.com/fr_fr/flycolor-x-cross-bl-32-4-in-1-40a-brushless-esc.html (contains 4 ESC and 1 BEC), https://hobbyking.com/fr_fr/flycolor-x-cross-bl-32-35a-brushless-esc.html (not tested yet, does not have a BEC)
BEC (optional, mandatory if the chosen ESC are not able to provide at least 5 V @ 0.5 A for the autopilot and 5 V @ 0.5 A for the pi0w (more might be necessary depending on the devices chosen)): https://hobbyking.com/fr_fr/turnigy-5a-8-26v-sbec-for-lipo.html or https://www.flashrc.com/castle_creations/7178-switch_bec_10a_out_48_9v_in_6s_max_castle_creations.html
1 Li-Po 3S 2.2 with XT60 connector: https://hobbyking.com/fr_fr/turnigy-2200mah-3s-25c-lipo-pack.html
XT60 connector and cable: https://store.drotek.com/xt60-male-connector
Li-Po battery charger with XT60 (for main battery) **and JST** (for RC battery, or other depending on the model, Arduino cables might fit for some connectors) **charge cables**: e.g. https://www.amazon.fr/Gwendoll-Chargeur-%C3%89quilibreur-D%C3%A9chargeur-H%C3%A9licopt%C3%A8re/dp/B07GPC14W5
ArduCopter-compatible autopilot HKPilot32, Dropix v2 or Pixhawk 3 Pro: https://hobbyking.com/fr_fr/hkpilot32-autonomous-vehicle-32bit-control-set-w-power-module.html (tested), https://store.drotek.com/dropix (choose with connectors and case, optionally with https://store.drotek.com/buzzer-piezo (helps to know the state of the autopilot) and https://store.drotek.com/led-pushbutton-switch (safety button)) or https://store.drotek.com/pixhawk-pro-autopilot + https://store.drotek.com/all-in-one-Pixhawk + https://store.drotek.com/jst-gh-to-jwt-28awg-4pins-cable + https://store.drotek.com/jst-gh-to-jwt-28awg-6pins-cable
Zener diode 1N5339 (to put reversed, e.g. https://fr.rs-online.com/web/p/diodes-zener/6547549/) **and capacitor 100-1000 uF** (polarized, e.g. 470 uF, e.g. https://fr.rs-online.com/web/p/condensateurs-aluminium/7111283/ or https://www.lextronic.fr/condensateurs-chimiques/2710-assortiment-de-120-condensateurs-electrolytiques.html), see recommendation http://ardupilot.org/copter/docs/common-powering-the-pixhawk.html
GPS (optional, connector to change to Arduino cables (JWT) for Dropix v2, JST-GH 6 pins for Pixhawk 3 Pro) : https://hobbyking.com/fr_fr/nano-px-gps-with-compass-for-pixhawk-px4.html
Vertical LIDAR to measure distance to the ground (optional, recommended for obstacle avoidance especially if not using any optical flow sensor): https://www.amazon.fr/Capteur-Distance-Laser-LIDAR-Lite-3/dp/B01MF845AB/ +470 Ohm resistor (e.g. https://fr.rs-online.com/web/p/resistances-traversantes/2671761/ or https://www.lextronic.fr/resistances-carbones/2428-lot-de-480-resistances-serie-e3.html), https://www.amazon.fr/Pixhawk-

t%C3%A9l%C3%A9m%C3%A9trie-monopoint-compatible-MakerHawk/dp/B0778B15G7 (not tested)
Optical flow sensor to limit horizontal drift inside in some flight modes (optional, connector to change to Arduino cables (JWT) for Dropix v2, JST-GH 4 pins for Pixhawk 3 Pro): https://store.drotek.com/optical-flow-kit-with-sonar (there is also a version without sonar but it was not tested), https://www.amazon.fr/KINGDUO-Cheerson-Quadcopter-Rechange-Optique/dp/B07K6HCJVV/ (not tested)
Micro USB cable for px and px4f config: https://www.amazon.fr/C%C3%A2ble-Micro-USB-6-5ft-Rampow/dp/B01GJC4YMC
PPM-compatible RC Rx and Tx (mandatory if not using pi0w, otherwise recommended as a backup in case of a problem with the Wi-Fi control): Best performance and safety: https://hobbyking.com/fr_fr/tbs-crossfire-micro-receiver-v2.html + https://hobbyking.com/fr_fr/tbs-crossfire-long-range-tx.html (or cheaper but not tested https://www.studiosport.fr/emetteur-crossfire-micro-tbs-a13915.html but without MAVLink telemetry through the 868 MHz link and Bluetooth, can still be sent by Wi-Fi through the pi0w)+ https://hobbyking.com/fr_fr/frsky-2-4ghz-accst-taranis-x9d-plus-digital-telemetry-transmitter-mode-2-eu-version.html + https://hobbyking.com/fr_fr/frsky-eva-carry-case-for-taranis-x9d-x9d-plus-transmitters.html + https://hobbyking.com/fr_fr/turnigy-nano-tech-1500mah-life-3s-9-9v-transmitter-pack-taranis-compatible-1.html ; Lower performance (not tested, without MAVLink telemetry through the 2.4 GHz link, can still be sent by Wi-Fi through the pi0w but perturbations of the Wi-Fi by the RC may occur): https://hobbyking.com/fr_fr/turnigy-tgy-i6-afhds-transmitter-and-6ch-receiver-mode-2.html + https://hobbyking.com/fr_fr/ppm-encoder-module-hkpilot-32.html + 4 AA batteries
Soldering and cutting tools, heat shrink, tape, etc.
Drill, 3D printer (optional, might help to attach the camera and ultrasonic telemeters, that kind of part might help also: https://www.leroymerlin.fr/v3/p/produits/corniere-pvc-blanc-20-x-20-mm-l-2-6-m-e1400914364)
Lightweight long belts (to help attaching some parts, e.g. ESC, autopilot): https://hobbyking.com/fr_fr/cable-ties-160mm-x-2-5mm-black-100pcs.html
Velcro to easily attach and detach the battery and the sensors: https://hobbyking.com/fr_fr/graphene-velcro-battery-strap-20-300mm-3pcs-bag.html + https://fr.rs-online.com/web/p/rubans-auto-agrippant/0549921/ + https://fr.rs-online.com/web/p/rubans-auto-agrippant/0549937/
Pins to solder on ultrasonic telemeters: https://www.lextronic.fr/barrettes-malesfemelles/152-connecteur-male-double-rangee-2x5.html or https://fr.rs-online.com/web/p/embases-de-circuit-imprime/6689684/
Long and short Arduino cables (JWT): https://www.lextronic.fr/straps/26303-jeu-de-100-straps-flexibles-f-f-15-cm.html + https://www.lextronic.fr/straps/25968-jeu-de-100-straps-flexibles-f-f-30-cm.html + https://www.lextronic.fr/straps/25645-jeu-de-100-straps-flexibles-m-m-15-cm.html + https://www.lextronic.fr/straps/25971-jeu-de-100-straps-flexibles-m-m-30-cm.html
Servos connectors and cables (optional, to gather some cables instead of using only Arduino cables): https://store.drotek.com/jwt-3-pin-connectors-5pcs + https://hobbyking.com/fr_fr/20cm-jr-22awg-twisted-extension-lead-m-to-f-5pcs.html
16 AWG black, red and yellow cable (optional, might be needed for ESC and motors if they are not provided with enough cable): https://hobbyking.com/fr_fr/turnigy-high-quality-16awg-silicone-wire-1m-black.html + https://hobbyking.com/fr_fr/turnigy-high-quality-16awg-silicone-wire-1m-red.html + https://hobbyking.com/fr_fr/turnigy-high-quality-16awg-silicone-wire-1m-yellow.html

JST male connector for potential additional payload, e.g. FPV transmitter+camera and OSD, lights to distinguish the front and the rear from far away https://hobbyking.com/fr_fr/led-red-lights-strip-with-jst-connector-200mm.html, etc. (optional): https://hobbyking.com/fr_fr/jst-male-2-pin-connectors-set-10pcs-set.html and https://hobbyking.com/fr_fr/jst-female-2-pin-connectors-set-10pcs-set.html , https://hobbyking.com/fr_fr/male-jst-battery-pigtail-12cm-length-10pcs-bag.html and https://hobbyking.com/fr_fr/female-jst-battery-pigtail-10cm-length-10pcs-bag.html
1 mm PVC plaque (can be cut using a knife to create supports for e.g. the autopilot, pi0w): e.g. https://fr.rs-online.com/web/p/plaques-en-plastique-pleines/4386370/
Raspberry Pi Zero W (mandatory if not using a PPM-compatible RC Rx and Tx and for obstacle avoidance) and its camera (optional): https://www.ldlc.com/fiche/PB00245076.html + https://www.amazon.fr/Raspberry-Pi-1080p-Module-Cam%C3%A9ra/dp/B01ER2SKFS + https://www.ldlc.com/fiche/PB00245339.html
Micro SD 32 GB (mandatory if using pi0w): https://www.amazon.fr/Kingston-SDCG2-32GB-adaptateur-Capturer/dp/B079SKDN2J/
Micro USB to female USB, USB keyboard and mouse and hub, screen with mini-HDMI adapter for pi0w configuration (optional, recommended if using pi0w)
A computer with Wi-Fi, Internet and USB ports (needs also **Bluetooth** if telemetry needs to be received through the 868 MHz Tx)
Micro SD reader for computer (mandatory if using pi0w)
Screws and spacers M2.5 for pi0w, **M2** for camera (optional) : https://www.amazon.fr/HSeaMall-Entretoise-Standoff-Accessoires-Assortiment/dp/B07CJGT93C/ + https://www.conrad.fr/p/rondelle-toolcraft-27-d9021-poly-194727-interieur-27-mm-m25-plastique-100-pcs-521722 + https://www.conrad.fr/p/rondelle-toolcraft-105343-na-interieur-22-mm-acier-200-pcs-105343
Logitech Gamepad F310S or compatible (optional, recommended if not using a PPM-compatible RC Rx and Tx): https://www.amazon.fr/Logitech-F310-Manette-pour-Bleu/dp/B00CJAF3ME
5 SRF02 ultrasonic telemeters for obstacle avoidance (optional, 4 horizontal and 1 up, for down either a LIDAR-Lite is used or in FlowHold mode with a cx-of or px4f the distance to the ground can be estimated (or measured by the integrated telemeter of the px4f if available) or the barometer pressure when arming is used): https://www.lextronic.fr/capteurs-ultrason-hyper/1183-telemetre-ultrason-srf02.html
Logic level converter (converts 3.3V I2C bus from pi0w to 5V I2C bus from SRF02 ultrasonic telemeters, choose a logic converter that has at least 2 bidirectional channels): https://www.lextronic.fr/conversion-niveau/29893-convertisseur-niveau-logique-3-3v-5v.html
Capacitor 100 uF (optional, to smooth the power lines of the SRF02): e.g. https://fr.rs-online.com/web/p/condensateurs-aluminium/5194059/
Nano servomotor to control camera orientation from one of the autopilot output (not tested, choose a servomotor with enough course to be able to go more than fully down and a little bit up to compensate the pitch angle that the drone will have when it goes forward or backward)

Examples of possible configurations

Cheapest, without obstacle avoidance: px+pi0w+joystick (via Wi-Fi socat or MAVProxy on pi0w)
Advanced: px+TBS Crossfire Rx and Tx with Taranis RC+pi0w+camera+5 SRF02+px4f/cx-of+lidar-lite+nano GPS

Part 1 : Mechanical structure, ESC, motors, propellers

For the placement of the ESC on the structure, several constraints need to be considered:

- The **ESC, battery and motors** might generate **magnetic perturbations** so it would be good to have them as far as possible from the **autopilot** to avoid perturbations on its internal compass, or it is possible to use an **external compass** (often integrated with a GPS).
- It is good to ensure the ESC are cooled by the air flow, however we must ensure their placement does not reduce the lift produced by propellers thrust…
- The shorter the cables are, the lighter the weight, the lower the power consumption (power is lost just due to the length of the cables for high currents)…

Belts can be used to attach them to the structure. **Do not cut or drill too much the structure** since it might affect its rigidity when the motors are providing thrust!

The battery should be provided with an XT60 connector (big yellow connector). Solder all the power inputs of the ESC together as well as a JST male connector for potential additional payload to the male XT60 connector that is supposed to mate with the one from the battery. Ensure you have a length of cable compatible with the future placement of the devices (not to short, not to long, pass through some holes in the drone structure before soldering, etc.). Do not forget to put heat shrinks before soldering to always avoid non-protected electrical contacts later.

Then, find how to correctly solder the ESC to the brushless motors (flux dispender might be necessary to help soldering the wires coming from the motors if they are cut, or use a corresponding connector instead if available). 2 types of motors and corresponding propellers are available: the propellers with a **central hole** correspond to motors with a **central pin** and must turn **clockwise**. The others must turn counter-clockwise (you can confirm that by checking the shape of the blades). For the clockwise motors an **Afro ESC** must be connected this way: black+red, white+yellow, green+black; a **Dualsky 22 A ESC**: white+blue, green+red, black+black. Use a servo tester to check if the motor is turning in the correct direction, change 2 wires if it is not (as long as the 3 cables are connected without bad contacts, only the direction is changed if wires are exchanged, change only when power is off). Find the correct convention using this method for the rest.

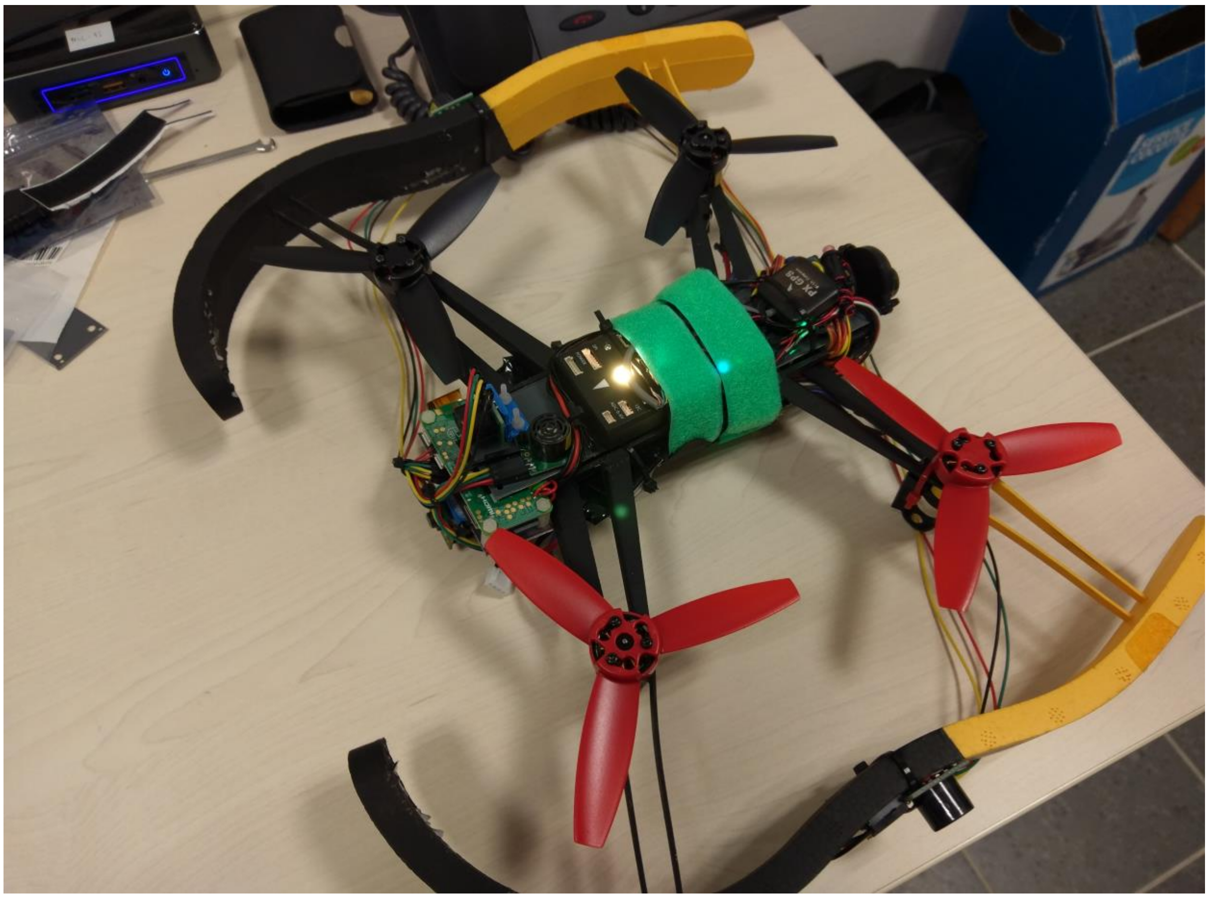
Be careful to ensure that **nothing is touching the rotating parts of the motors** (e.g. the cable from the motor can be clamped in the landing gear parts). **Use protection gloves** if needed when using the propellers.

Check Bebop pictures to determine where is the front (rear is less vertical than front when viewed from the side) and where should be placed the clockwise propellers.

Vibrations should be low on a small drone, however we need to be aware that they can cause problems to the internal accelerometers of the autopilot especially for bigger drones, see http://ardupilot.org/copter/docs/common-measuring-vibration.html.

Be careful to install the autopilot in the correct direction w.r.t. the Bebop structure. You might want to use PVC plaques, spacers, 3D printed parts to prepare room for the other components (pi0w camera position: connector towards the bottom). Note that the different connectors on the autopilot and pi0w should be left available and the battery needs to be easily removable.

Although not mandatory, it is good to have the autopilot buzzer (it will give useful information about the state of the autopilot) and safety button.



Part 2 : Preparing the autopilot

Receiver power consumption

There is usually a RC row on the autopilot that is supposed to be connected to the receiver (most of the autopilots need to receive PPM signals from the receiver, which is a kind of serialization of the PWM channels, note that the wireless communication between the transmitter and the receiver might be also in PPM, but with a higher frequency). Autopilots such as the Pixhawk/HKPilot32 are supposed to provide 5 V to the receiver through this row, but with a very limited current. For powerful receivers, it is better to avoid using the 5 V coming from the RC row from the autopilot and use a 5 V coming from elsewhere (e.g. a BEC or a dedicated 5 V battery) instead. See http://ardupilot.org/copter/docs/common-powering-the-pixhawk.html#common-powering-the-pixhawk , http://ardupilot.org/copter/docs/common-pixhawk-wiring-and-quick-start.html

Installing a Zener diode and a capacitor to limit the risks of currents spikes: see http://www.icarusfpv.com/2014/11/installing-a-zener-diode-and-capacitor-for-hkpilot32-pixhawk/ .

Configuring the transmitter and the receiver

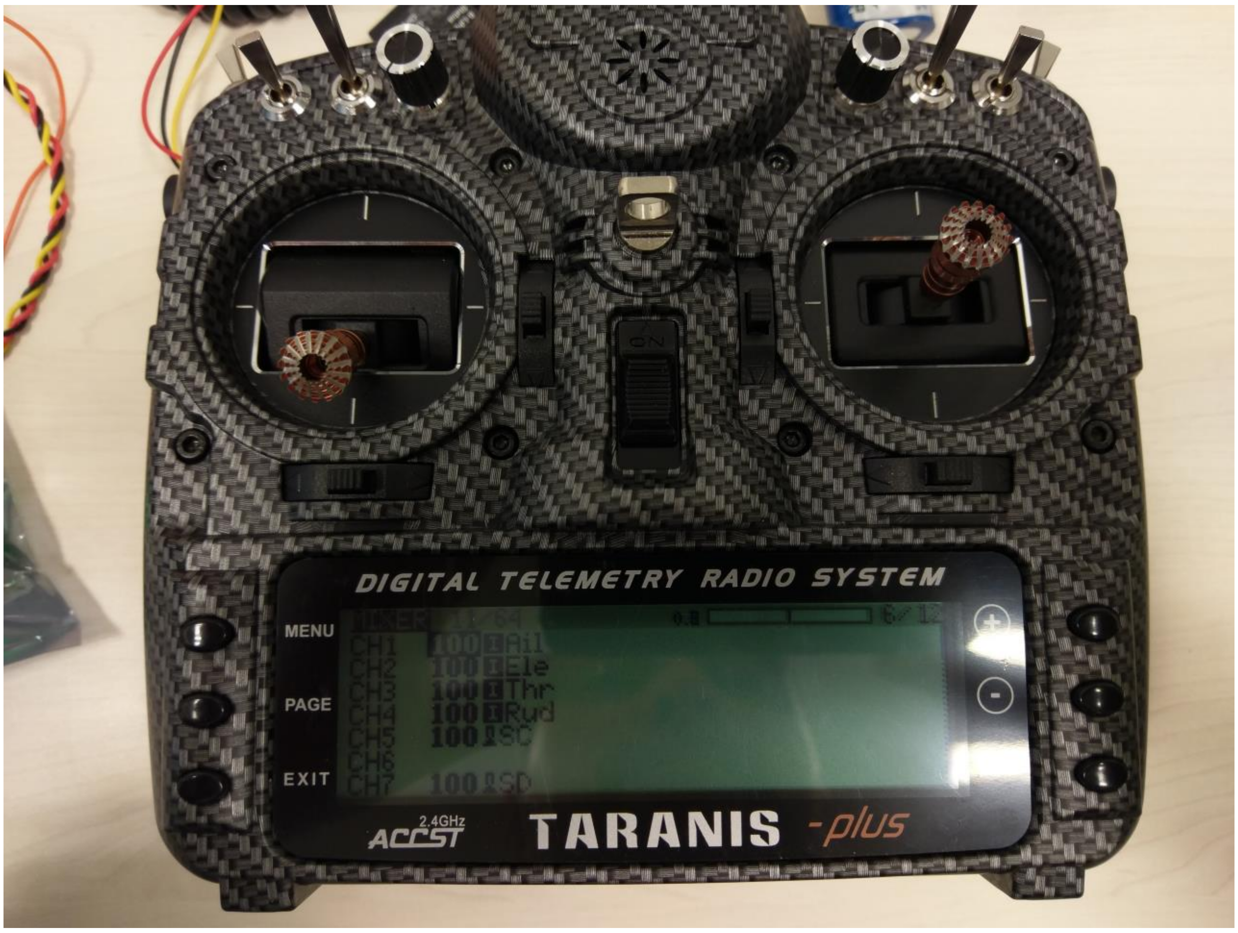
Follow their documentation… For TBS Crossfire, check specific info in https://www.ensta-bretagne.fr/lebars/Share/drone_take-off_check-list.txt .

Taranis X9D Plus transmitter

Long press on MENU to access Radio Setup, decrease volume, disable inactivity alarm, disable splash screen, check low battery voltage level, update date and time.

Short press on MENU to access Model Setup, disable internal RF and set external RF to PPM (especially if using TBS Crossfire transmitter), disable switch warning, use Mixer to change channel order (usually Ail, Ele, Thr, Rud) and add switches, e.g.:

- CH1 Ail
- CH2 Ele
- CH3 Thr
- CH4 Rud
- CH5 SC
- CH7 SD
- CH8 SG
- CH10 SA
- CH11 SB
- CH12 LS

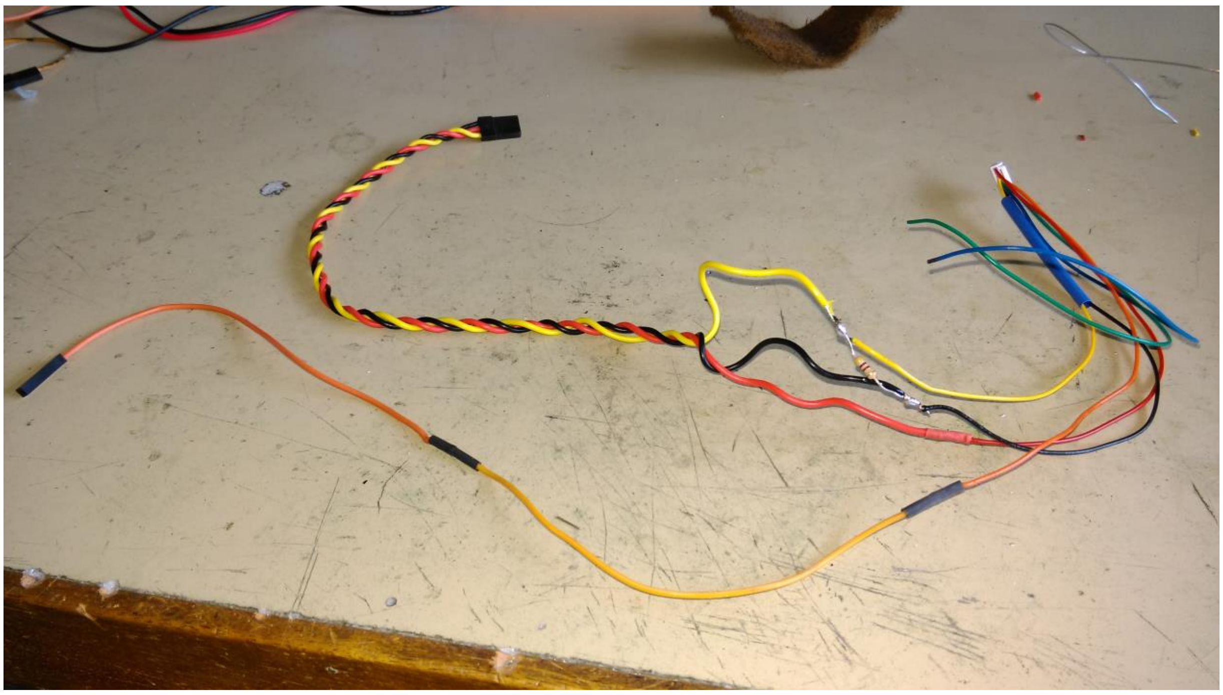


LIDAR-Lite v3 connections

A vertical LIDAR towards the bottom is typically used to get an accurate estimation of the distance to the floor.

The LIDAR-Lite should send **PWM** through its yellow cable to autopilot **AUX5** input with a **470 Ohm resistor** on the ground in parallel. To save power if needed, the orange cable can be connected to **AUX6** if the autopilot has an available **AUX6**, or **AUX4** (change **RNGFND_STOP_PIN** parameter accordingly, see later in Mission Planner). It should be powered with 5 V through its red wire and the ground should be on its black wire.

Warning : **LIDAR-Lite v3** has a different and globally inverted connector compared to **v2**!



Optical flow sensor

An optical flow sensor can be used by the autopilot to limit horizontal drift inside in some flight modes.

Follow http://ardupilot.org/copter/docs/common-cheerson-cxof.html or http://ardupilot.org/copter/docs/common-px4flow-overview.html (implies to configure also a LIDAR-Lite, note that it is not mandatory in FlowHold mode as this mode will try to estimate the distance to the ground from the optical flow data). Alternatively, the PX4FLOW could by connected to the pi0w instead of directly to the autopilot. This way, you could handle by yourself its data to stabilize horizontally the drone, if the autopilot does not handle it reliably enough.

Configure external compass and GPS

Note that when only an internal compass is available, it will get the first ID in Mission Planner. However, when we add an external compass (often mounted with the GPS), the external will get the first ID while the internal will get the second after rebooting the autopilot… Be sure to disable one or the other in Mission Planner if it is expected to be unreliable (see later, when checking the compass calibration).

ArduCopter configuration through Mission Planner

As seen in https://www.ensta-bretagne.fr/lebars/Share/TD_robots_sensors_actuators.pdf , **Mission Planner** should be used to configure the autopilot with **ArduCopter** firmware and all the necessary calibrations should be done (compass calibration should be done again when everything has been mounted on the drone).

Here are the modes that should be configured to match the transmitter channels configuration:

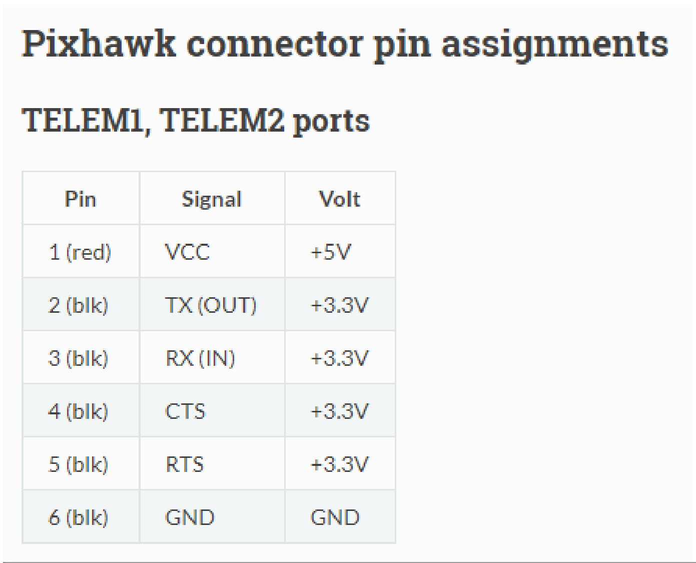
- CH5 **Land** or **FlowHold** or **Loiter** (low position of the corresponding 3 position switch on the radio, FlowHold needs an optical flow sensor, Loiter needs in addition a vertical LIDAR), **Stabilize** (middle switch position), **AltHold** (up switch position)
- CH7 **Motor Emergency Stop**
- CH8 **Smart_RTL**
- CH9 **Land**
- CH10 **AutoTune**
- CH11 **Object Avoidance** (need to send **DISTANCE_SENSOR** mavlink messages)
- CH12 **Camera Trigger**

Check additional configuration info in https://www.ensta-bretagne.fr/lebars/Share/drone_take-off_check-list.txt .

Part 3 : Soldering the necessary devices on the Raspberry Pi

Connect the camera to the pi0w.

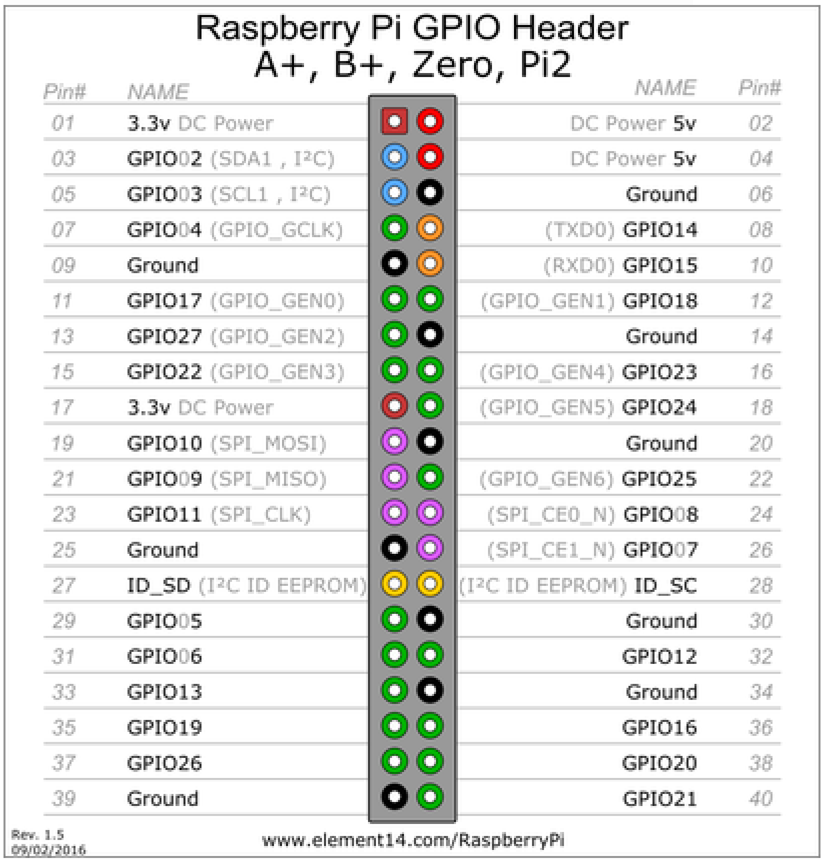
Check the type of connector used as telemetry port on the autopilot. The **RX**, **TX**, **GND** signals will need to be connected to the pi0w UART (by default **/dev/ttyS0** for a Raspberry Pi Zero W, usually baudrate **57600** from the autopilot telemetry port, however this UART has limitations on the possible baudrates so it is better to use **/dev/ttyAMA0** UART originally dedicated to the BlueTooth, by adding **dtoverlay=pi3-disable-bt** in **/boot/config.txt** and run **sudo systemctl disable hciuart**, be sure also to choose **enable serial port** and **disable console on serial port** in **raspi-config**). Double-check the logical levels of the RX and TX signals on both sides (autopilot and pi0w), if they are not between 0 and 3.3 V you will need to add a level converter. Remember that the TX from the autopilot should be connected to the RX of the pi0w and conversely. Do not rely only on the colors of the cables and connectors, they might vary from one brand to another.



Pixhawk connector pin assignments

TELEM1, TELEM2 ports

Pin	Signal	Volt
1 (red)	VCC	+5V
2 (blk)	TX (OUT)	+3.3V
3 (blk)	RX (IN)	+3.3V
4 (blk)	CTS	+3.3V
5 (blk)	RTS	+3.3V
6 (blk)	GND	GND



Raspberry Pi GPIO Header
A+, B+, Zero, Pi2

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Rev. 1.5
09/02/2016

www.element14.com/RaspberryPi

Pin 8 (TXD0, GPIO14) pi0w
Pin 9(GND) pi0w
Pin 10 (RXD0, GPIO15) pi0w

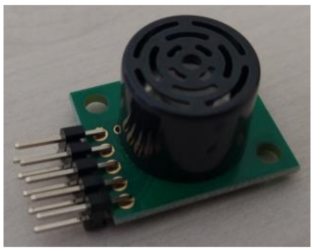
The pi0w power supply needs to be 5 V with at least 1 A. One of the BEC from the ESC or a dedicated BEC can be used for that.

Pin 4 (5 V) pi0w
Pin 6 (GND) pi0w

Soldering the telemeters on the Raspberry Pi

All the SRF02 ultrasonic telemeters can be connected to the pi0w on its I2C bus 1 (see Pin 3 (SDA1, GPIO02), Pin 5 (SCL1, GPIO03) on the pi0w). However, for the telemeters the I2C bus logic levels are between 0 and 5 V while on the pi0w it is 0 to 3.3 V, so we need a level converter (see Pin 1, Pin 2, Pin 14 on the pi0w). It is advised to add a 100 uF capacitor on the power lines since the telemeters generate spikes when emitting….

They need to be removable since their default address will need to be changed, for that a connector can be soldered like in the picture below.



We can use Arduino cables to connect all of them together.

Part 4 : Configuring the Raspberry Pi and the telemeters

Modifying pi0w micro SD (without dedicated screen, mouse and keyboard) directly by mounting it on Ubuntu

Get an existing Raspbian micro SD image, e.g. from https://www.ensta-bretagne.fr/lebars/Share/pi.gz , extract the image (.gz corresponds to a compressed file) and install it on the micro SD using the **dd** command, e.g.
dd if=pi of=/dev/mmcblk bs=2M
(change **/dev/mmcblk** with the correct name corresponding to the micro SD, double-check that you are not overwriting your hard disk or USB drive!).

Then, mount the micro SD, add a file named **ssh** in the **boot** partition to enable ssh at first startup and change the necessary network configuration files in the **rootfs** partition so that the pi0w connects to a desired Wi-Fi network with a desired IP address and hostname (see e.g. section about **/etc/wpa_supplicant/wpa_supplicant.conf** and **/etc/dhcpcd.conf** in https://www.ensta-bretagne.fr/lebars/Share/Raspbian.txt as well as section about hostname in https://www.ensta-bretagne.fr/lebars/Share/Ubuntu.txt to be able to find easily the pi0w IP address if set by DHCP…). This should avoid the use of a mouse, keyboard, screen, etc.

Information about the configuration used in the provided image are here:
https://www.ensta-bretagne.fr/lebars/Share/Raspbian.txt
https://www.ensta-bretagne.fr/lebars/Share/Ubuntu.txt
https://www.ensta-bretagne.fr/lebars/Share/unattended.sh

Note: it is possible also to directly edit the image before installing it on the micro SD. Change offset depending on the start position (to convert in bytes) of the ext4 partition given by
fdisk -l pi
sudo mkdir -p /mnt/rootfs
sudo mount -r -v -o offset=67108864 -t ext4 pi /mnt/rootfs

Everything is in **/mnt/rootfs**.
On Windows, it is partly possible (there might be bugs, problems of permissions, case, etc.) to do the same using an **ext4** driver such as **Ext2Fsd** (in case of hanging problems, right click on the partition in **Ext2 Volume Manager** and flush cache, also preferably remove the drive letter before trying to eject the corresponding device…) and **OSFMount** tool (choose to mount entire image as virtual disk, removable, direct write mode) to mount the image as a disk…

After the first startup, resize the ext4 partition to the whole SD card (see **Advanced Options** in **raspi-config**…).

Testing an I2C Devantech SRF02 ultrasonic telemeter on the pi0w

```
sudo i2cdetect -y 1
```

to see the address of the devices currently connected to the I2C bus 1 of the pi0w.
To send a distance measurement request and read the answer (high byte and low byte, in cm):

```
sudo i2cset -y 1 0x70 0 0x51
sleep 1
sudo i2cget -y 1 0x70 2
sudo i2cget -y 1 0x70 3
```

To change the default address:

```
sudo i2cset -y 1 0x70 0 0xA0
sudo i2cset -y 1 0x70 0 0xAA
sudo i2cset -y 1 0x70 0 0xA5
# Last parameter is new address << 1...
#sudo i2cset -y 1 0x70 0 0xE2
#sudo i2cset -y 1 0x70 0 0xE4
#sudo i2cset -y 1 0x70 0 0xE6
#sudo i2cset -y 1 0x70 0 0xE8
sudo i2cset -y 1 0x70 0 0xEA
#sudo i2cset -y 1 0x70 0 0xEC
sudo i2cdetect -y 1
```

See https://www.robot-electronics.co.uk/htm/srf02tech.htm for more info.

Part 5 : MAVLink tests

Be careful when running code with a real drone, it might take off!

Before doing obstacle avoidance, check that you can control the drone from the pi0w. The MAVLink protocol is used by ArduCopter to communicate with Mission Planner and any other application. For example, **pymavlink** Python package can be used to control the drone using MAVLink messages: see http://ardupilot.org/dev/docs/copter-commands-in-guided-mode.html , https://mavlink.io/en/messages/common.html and the sample https://www.ensta-bretagne.fr/lebars/Share/nogps_takeoff_land_pymavlink.py . It is also possible to see sent/received MAVLink messages with Mission Planner in **Config/Tuning\Planner\Layout\Testing screen\MAVLink Inspector**. **Dronekit** is also a higher level package on top of pymavlink but at the moment most of its functionalities require GPS fix.

Before trying to communicate directly with the drone, it is also better to test the Python code with a simulated autopilot (**SITL**), see Simulation in Mission Planner (you can use **vJoy Feeder** software to simulate a joystick or a real one to control manually the simulated drone, see **Flight Data\Actions\Joystick** in Mission Planner to enable and configure the joystick). You will probably have to disable some pre-arm checks in **Config/Tuning\Standard Params** for the simulation, and change the parameter **SIM_GPS_DISABLE** to simulate an indoor drone without GPS fix.

Part 6 : First tests

For the first flight, it should be better to have the least possible devices on the drone to limit the damages in case things go wrong. Unless there is no RC receiver available, the pi0w and its devices (except the autopilot) should not be necessary (i.e. remove pi0w, camera, telemeters, optical flow sensor). If no RC receiver is available, we will need the pi0w, use **socat** (e.g. **socat -d -d /dev/ttyAMA0,b57600,raw,echo=0,mode=666 tcp-listen:5760,reuseaddr** maps the serial port **/dev/ttyAMA0** to a TCP server running on TCP port **5760**) or **mavproxy** (e.g. **mavproxy.py --master=/dev/ttyAMA0,57600 --out=udpbcast:192.168.2.255:14550 --out=tcpin:0.0.0.0:5760 --out=tcpin:0.0.0.0:5762 --out=tcpin:0.0.0.0:5763** for an UDP connection (replace 192.168.2.255 with the broadcast address of your network, or the IP address of the laptop) and several TCP connections, however it has a high processor usage, https://github.com/intel/mavlink-router might be a good replacement for some of its functionalities) commands to send telemetry to a laptop running Mission Planner through Wi-Fi (in Mission Planner, choose TCP (e.g. if using **socat**) or UDP (e.g. if using **mavproxy**) in the top left to choose how to connect), connect a joystick to the laptop and configure it in Mission Planner (see **Flight Data\Actions\Joystick** in Mission Planner to enable and configure the joystick).

For a small drone like the Parrot Bebop, it is possible to keep it in one hand (with **protection gloves**) for a first test, to check that it tries to stabilize its roll and pitch, that it tries to go in the correct direction, etc. Follow https://www.ensta-bretagne.fr/lebars/Share/drone_take-off_check-list.txt to ensure you are ready for this test.

If successful, you are ready for a first very short fly to check vibrations (as suggested in https://www.ensta-bretagne.fr/lebars/Share/drone_take-off_check-list.txt). For this test, the drone should be attached with a safety cable so that it does not try to fly far away in case things go wrong. Remember that if it has too many vibrations, the **Land** mode often make it **fly away at full speed** since its vertical speed estimation obtained from its accelerometers will be wrong! Therefore, the idea will be to take off only a little bit enough to leave the ground (we need to have no contact with anything (except the safety cable if any) for the vibration test to be meaningful), stop using the **Motor Emergency Stop** switch (probably the most reliable way to stop immediately the motors), then disarm, and check the dataflash log files as described in http://ardupilot.org/copter/docs/common-measuring-vibration.html . Do not forget to ensure the safety cable does not fly into the propellers with the air flow (e.g. add a little mass at 25 cm from the drone and try to leave the cable so that it is a little bit tight).

Part 7 : Obstacle avoidance

Write a code on the pi0w to provide distance measurements from the I2C Devantech SRF02 ultrasonic telemeters to the autopilot through the MAVLink protocol. See http://ardupilot.org/dev/docs/code-overview-object-avoidance.html for the MAVLink messages to send and first test the influence of the messages with fake distances in SITL (we can also send the **ground_distance** value from the ultrasonic telemeter integrated in px4f if there is no other downward telemeter for altitude above ground measurement (otherwise it is not used by the autopilot)…).

Part 8 : Remote video and telemetry

If a camera is connected to the pi0w, **gstreamer** (see e.g. the last part of https://www.ensta-bretagne.fr/lebars/tutorials/TD_pi.pdf for additional information) can be run on the pi0w with
gst-launch-1.0 -v v4l2src device=/dev/video0 do-timestamp=true ! video/x-h264, width=1920, height=1080, framerate=30/1 ! h264parse ! queue ! rtph264pay config-interval=10 pt=96 ! udpsink host=192.168.2.1 port=5600
(replace 192.168.2.1 with the address of the laptop that should receive the video) to send the video and **QGroundControl** (tested with v3.2.7, test its different compatibility shortcuts if needed) can be launched on the laptop to display the video (if it does not work by default, check in its settings that **Video Source** is set to **UDP Video Stream** at **UDP Port 5600**, note that it is also possible to record the videos from it in the laptop user folder). Note that **QGroundControl** can replace also **Mission Planner** to get the telemetry and use a joystick, and **Mission Planner** can be also configured to display the video (extract http://firmware.eu.ardupilot.org/Tools/MissionPlanner/gstreamer/ so that you get C:\gstreamer\1.0\x86_64\bin , if you have a 32 bit-only OS it might not work, try to use gstreamer-1.0-x86-1.9.2.zip instead and rename C:\gstreamer\1.0\x86 to C:\gstreamer\1.0\x86_64, there might be a loading time of several minutes the first time **Mission Planner** detects a video stream…).

If needed to rotate or flip the camera images, it can be done at the hardware level with a command similar to
gst-launch-1.0 -v rpicamsrc bitrate=10000000 preview=false rotation=180 hflip=1 vflip=1 do-timestamp=true ! video/x-h264, width=1920, height=1080, framerate=30/1 ! h264parse ! queue ! rtph264pay config-interval=10 pt=96 ! udpsink host=192.168.2.1 port=5600
(ensure that https://github.com/thaytan/gst-rpicamsrc is installed). The network usage should be around 10 Mbps. Alternatively,
v4l2-ctl --set-ctrl rotate=180 --set-ctrl horizontal_flip=1 --set-ctrl vertical_flip=1
command can be used.

If needed to reduce processor and network usage without losing too much quality, try
gst-launch-1.0 -v rpicamsrc sensor-mode=5 preview=false rotation=180 hflip=1 vflip=1 do-timestamp=true ! video/x-h264, width=1280, height=720, framerate=25/1 ! h264parse ! queue ! rtph264pay config-interval=10 pt=96 ! udpsink host=192.168.2.1 port=5600
Check
gst-device-monitor-1.0
gst-inspect-1.0 rpicamsrc
to get more information on the camera internal capabilities so that the least processing tasks are left to the pi0w processor, note also that the field of view might change depending on the sensor-mode parameter...

If needed to record a file locally as well, try

```
gst-launch-1.0 -e -v rpicamsrc sensor-mode=5 preview=false rotation=180 hflip=1 vflip=1 do-timestamp=true ! video/x-h264, width=1280, height=720, framerate=25/1 ! h264parse ! tee name=t t. ! queue ! rtph264pay config-interval=10 pt=96 ! udpsink host=192.168.2.1 port=5600 t. ! queue ! avimux ! filesink location=test.avi
```

which has **-e**, **! tee name=t t.** and **t. ! queue ! avimux ! filesink location=test.avi** added, use

```
sudo killall -s 2 gst-launch-1.0
```

or **CTRL+C** to stop properly.

If needed to have a separate remote program that uses the camera (e.g. to detect objects with OpenCV) while still sending the video remotely to **QGroundControl** or **Mission Planner**, try

```
gst-launch-1.0 -v rpicamsrc sensor-mode=5 preview=false rotation=180 hflip=1 vflip=1 do-timestamp=true ! video/x-h264, width=1280, height=720, framerate=25/1 ! h264parse ! tee name=t t. ! rtph264pay config-interval=10 pt=96 ! multiudpsink clients=127.0.0.1:5600,192.168.2.1:5600 sync=false async=false
```

(replace **127.0.0.1** with an other IP address), however extra CPU might be used for the duplication of UDP packets (another possibility could be to use broadcast or multicast, but it might be worse in performance if it is not correctly used) and at least **OpenCV 3.4.2** (see https://www.ensta-bretagne.fr/lebars/Share/Raspbian.txt to get a prebuilt version of **OpenCV 4.2.0** if needed) might be necessary to read the UDP data (see https://www.ensta-bretagne.fr/lebars/Share/video_opencv_bluerov.py and https://www.ensta-bretagne.fr/lebars/Share/bluerov_vlc.sdp).

If needed to have an embedded program in the pi0w that uses the camera while still sending the video remotely to **QGroundControl** or **Mission Planner**, it is possible to install **v4l2loopback** driver and do

```
sudo modprobe v4l2loopback devices=1
gst-launch-1.0 -v rpicamsrc sensor-mode=5 preview=false rotation=180 hflip=1 vflip=1 do-timestamp=true ! video/x-raw, format=I420, width=1280, height=720, framerate=25/1 ! tee name=t t. ! omxh264enc control-rate=3 target-bitrate=10000000 ! h264parse ! queue ! rtph264pay config-interval=10 pt=96 ! udpsink host=192.168.1.2 port=5600 sync=false async=false t. ! v4l2sink device=/dev/video1 sync=false async=false
```

however it seems to use 95 % of the pi0w CPU, which does not leave much computing power for the rest. Additionally, at least **OpenCV 3.4.3** (see https://www.ensta-bretagne.fr/lebars/Share/Raspbian.txt to get a prebuilt version of **OpenCV 4.2.0** if needed) might be necessary to read the camera in this configuration (support of I420 format), other alternative options did not seem to work well… Another possibility

```
gst-launch-1.0 -e -v rpicamsrc sensor-mode=5 preview=false rotation=180 hflip=1 vflip=1 do-timestamp=true ! video/x-h264, width=1280, height=720, framerate=25/1 ! h264parse ! tee name=t t. ! queue ! rtph264pay config-interval=10 pt=96 ! udpsink host=192.168.2.1 port=5600 sync=false async=false t. ! omxh264dec ! v4l2sink device=/dev/video1 sync=false async=false
```

seems to use 90% of the pi0w CPU, but do not work for the pi4 and might use more GPU since a decoding step is done, contrary to the first possibility (and the encoding in H264 is normally done somehow in both cases)…

Note that you can use the **screen** command on the pi0w to avoid the commands launched from ssh to stop if there is a temporary network disconnection, or launch them automatically at startup of the pi0w, e.g. in

```
sudo crontab -e
```
add
```
@reboot sh -c "while true; do sudo mux_server.py -d /dev/ttyAMA0 -b 57600 -p 5760; sleep 5; done" &
@reboot sh -c "while true; do sudo socat -d tcp:127.0.0.1:5760 udp4-sendto:192.168.2.1:14550; sleep 5; done" &
@reboot sh -c "while true; do gst-launch-1.0 -v v4l2src device=/dev/video0 do-timestamp=true ! video/x-h264, width=1920, height=1080, framerate=30/1 ! h264parse ! queue ! rtph264pay config-interval=10 pt=96 ! udpsink host=192.168.2.1 port=5600; sleep 5; done" &
```
or especially for commands that would require a GUI, add files like https://www.ensta-bretagne.fr/lebars/Share/telem.desktop , https://www.ensta-bretagne.fr/lebars/Share/udptelem.desktop , https://www.ensta-bretagne.fr/lebars/Share/video.desktop to **~/.config/autostart/**. A **mux_server.py** (get it from https://github.com/lebarsfa/mux_serial/raw/master/mux_server.py) and a **socat** commands have been added to make the telemetry available through multiple isolated TCP connections as well as an UDP connection, this takes less processor usage than the corresponding **mavproxy** command and the UDP should enable better automatic reconnection to **Mission Planner** or **QGroundControl** while the multiple isolated TCP connections could be for other embedded programs that would need to receive or send data to the autopilot without stealing data between each other. **QGroundControl**, **gstreamer**, **mavproxy** through **UDP** commands are inspired by what is used on the **BlueROV2** submarine, that has a similar hardware architecture.