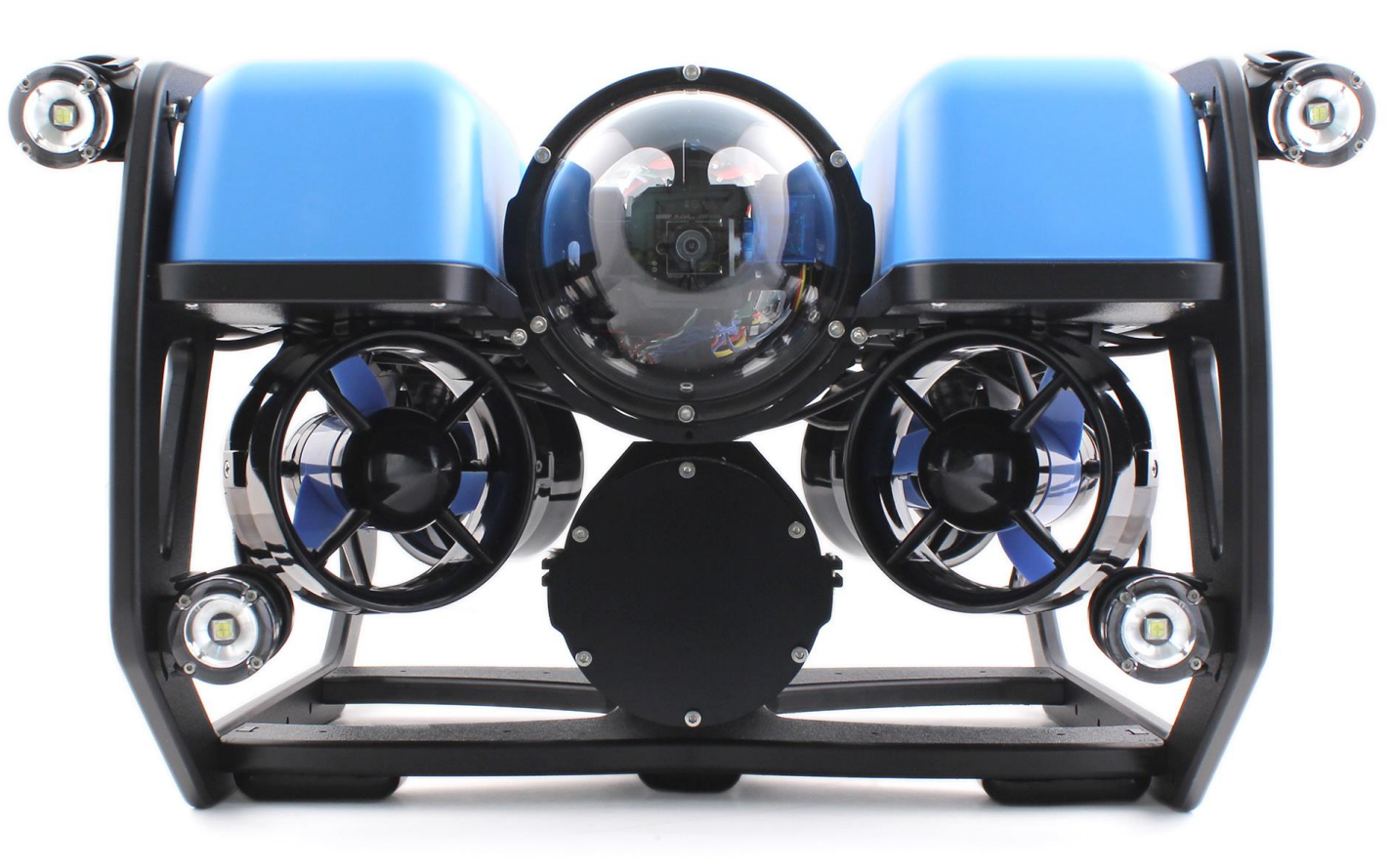
BlueROV2



BlueROV2

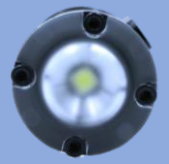
- Robot sous-marin téléguidé (ROV)
- Open-source et modulaire
- Contrôle par un PC en surface avec une manette Xbox
- Transmission de données par cordon ombilical (ethernet)
- Batterie
- Caméra

Hardware

Capteur de pression

I²C

Lampes



Moteurs

ESC
(contrôleur de vitesse)

Contrôleur de vol PixHawk

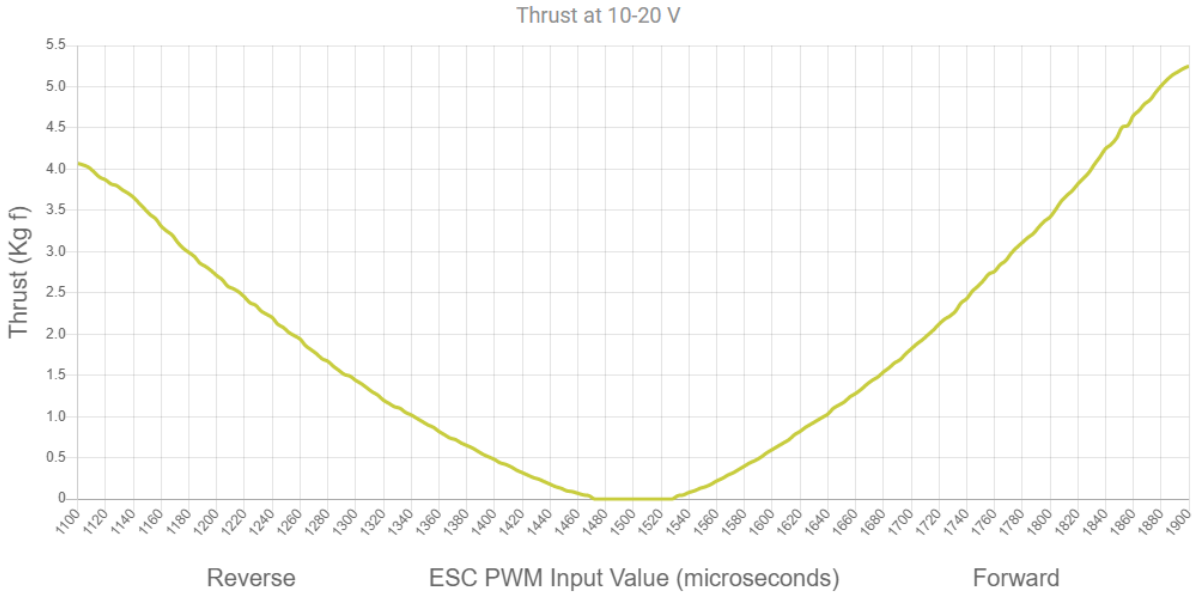
Contrôle la vitesse des moteurs en fonction des commandes utilisateur et des capteurs

Capteurs intégrés :
- Gyroscope
- Accéléromètre
- Magnétomètre
- Baromètre

Documentation

Propulseur et ESC

Le PixHawk envoie un signal de commande PWM vers l'ESC qui fournit la tension triphasée au propulseur.



Propulseur T200

Moteur brushless triphasé
Plage de fonctionnement : 7-20V (Nomimal 16V)

Documentation

Basic ESC

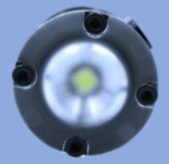
Contrôleur de vitesse pour moteur brushless

Fonctionnement PWM (cf graph.) :
Arrêt à 1500 ± 25 µs
Vitesse avant max : 1900 µs
Vitesse arrière max : 1100 µs

Ne fournit pas de télémétrie

Documentation

Hardware



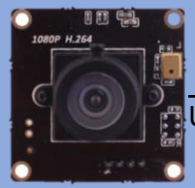
Contrôleur de vol PixHawk

Contrôle la vitesse des moteurs en fonction des commandes utilisateur et des capteurs

Capteurs intégrés :
- Gyroscope
- Accéléromètre
- Magnétomètre
- Baromètre

Documentation

Caméra



USB

Télémétrie

USB

Commandes

USB utilisateur

Raspberry Pi

Ordinateur « Compagnon »

Firmware ArduSub :
- Contrôle de stabilité
- Maintient en position
- Navigation autonome

Communication MAVLink

OpenCV

BlueRov

PixHawk

Raspberry Pi

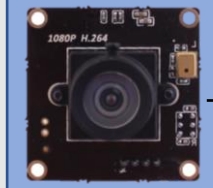
Ordinateur « Compagnon »

Firmware ArduSub :

- Contrôle de stabilité
- Maintient en position
- Navigation autonome

Communication MAVLink

OpenCV



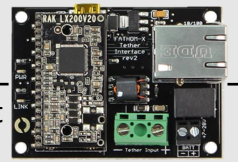
Télémétrie

USB

Commandes utilisateur

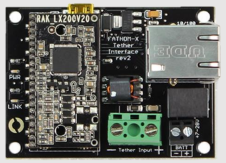
USB

Ethernet



Contrôle en surface

Fathom-X Topside Board



Ethernet

Fathom Tether (cordon ombilical)

QGroundControl

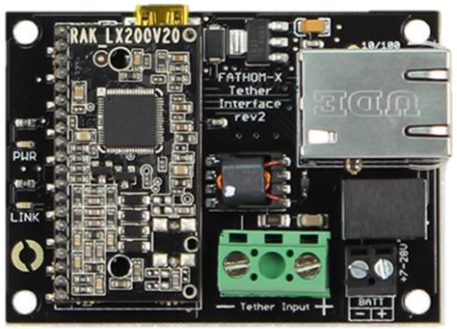
Cordon ombilical et connexion

Fathom Tether

Câble flottant de 25 à 300m
Composé de 4 paires torsadées non blindées (UTC)
Arrangé comme un câble Cat5

Connectique :

Documentation



Fathom-X Topside Board

2 Cartes : une dans le ROV et une sur terre

Le Fathom Tether s'y connecte via le connecteur vert

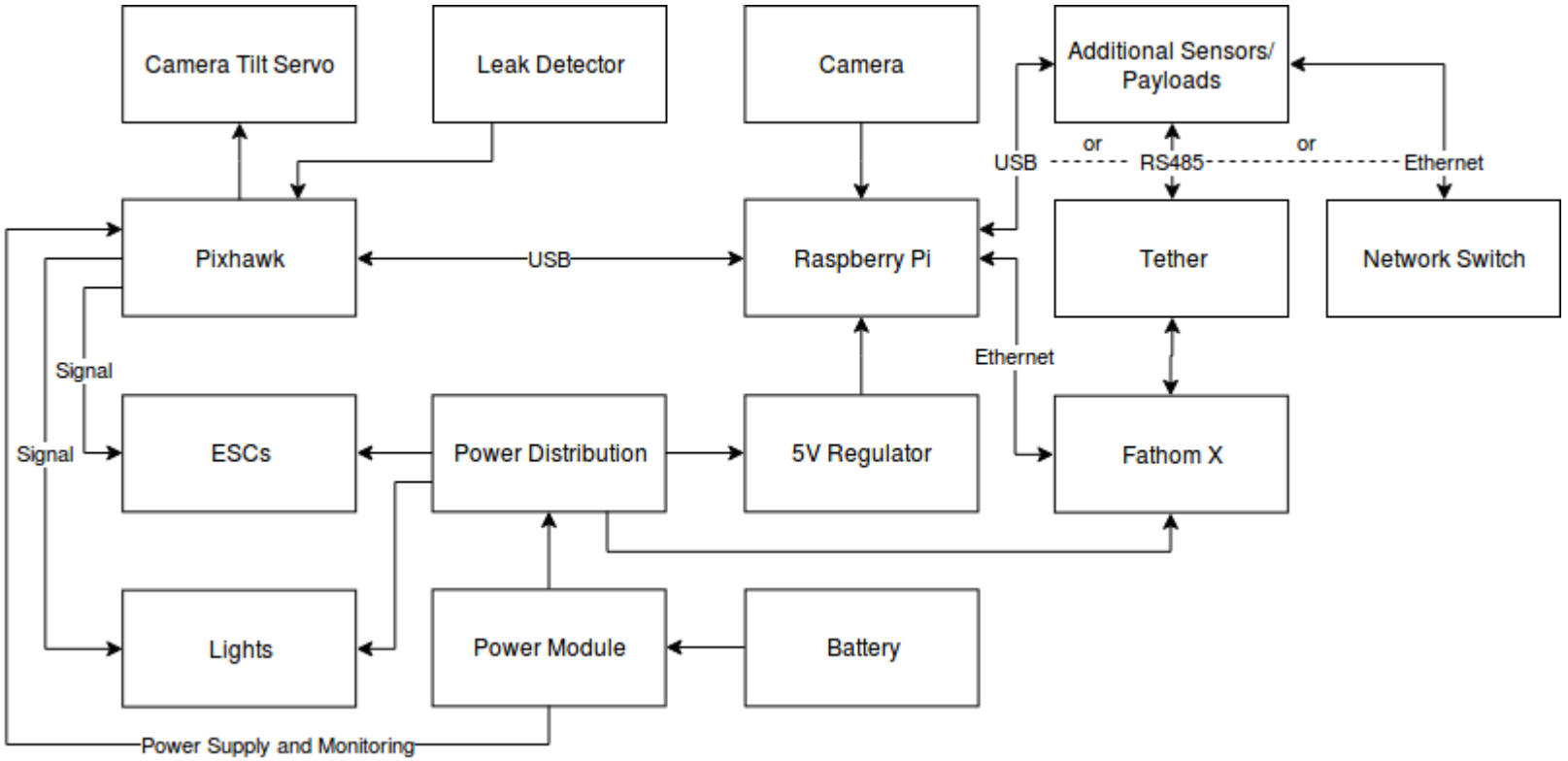
Agit en ampli de puissance + modulation

Offre de l'ethernet en sortie : connexion à la RPi et à l'ordinateur de contrôle (QGC)

Technologie Homeplug (CPL)

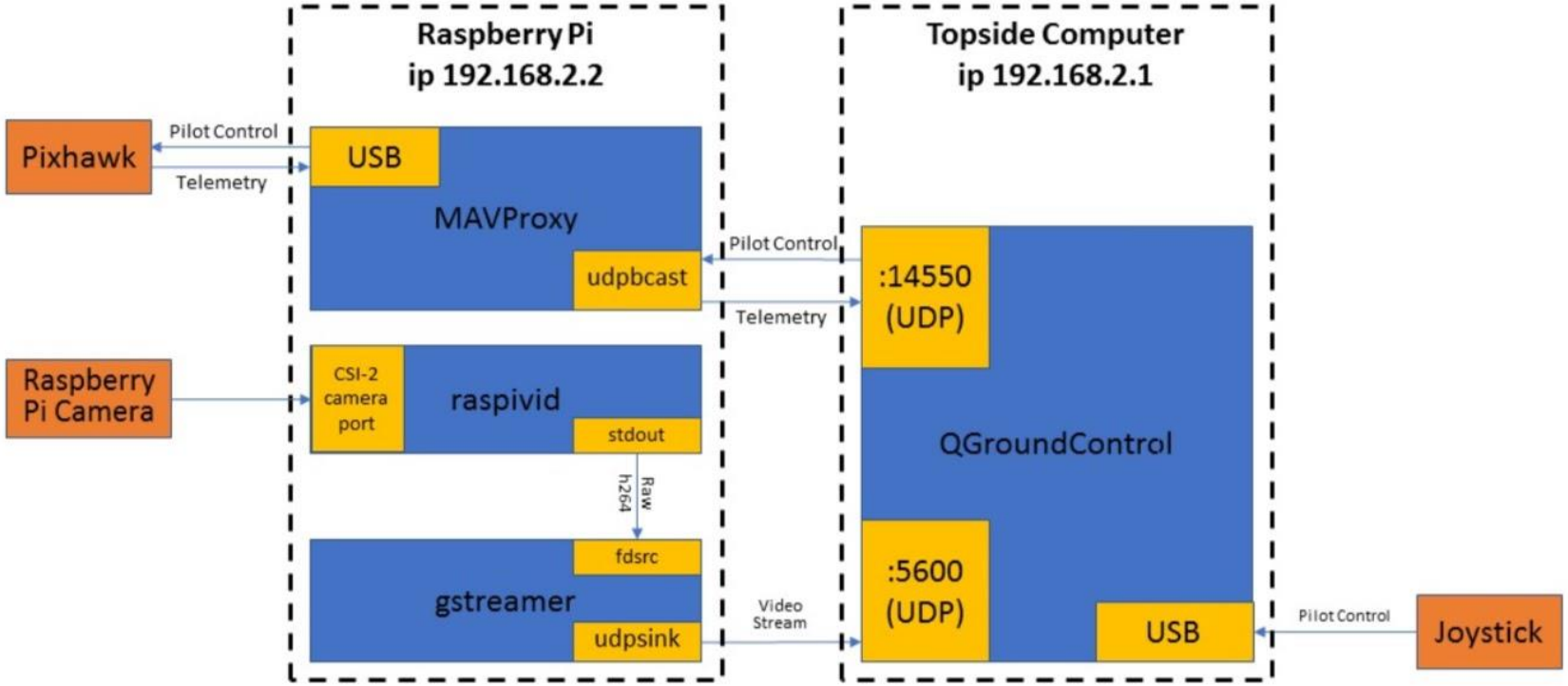
Documentation

Documentation et synthèse hardware



- Datasheet
- Composants hardware
- Assemblage du BlueRov

Software



Raspberry Pi :

- Equipée du firmware ArduSub : cerveau du ROV
- Relaie les communications entre autopilot et QGC
- Streame la vidéo vers QGC

QGroundControl (QGC) permet :

- Contrôle et mise en place autopilot
- Vidéo live et enregistrement
- Gérer les paramètres du ROV

MAVLink

- Protocole de communication pour drones
- Messages définis dans fichiers XML
- En autopilot : envoi d'un signal heartbeat pour assurer la bonne connexion

Software In The Loop (SITL)

- Simule le ROV : permet de faire des tests de commande, autopilot..
- Installation sur Linux (VM)
- Tutoriel : https://gitlab.polytech.umontpellier.fr/prj-semestriels/2019-2020-s6/02-robot-bluerov/-/wikis/Tutoriel-SITL

Documentation Software

- Pymavlink (implantation Python)
- Documentation générale
- OpenCV
- ArduSub
- QGroundControl

Contrôle du ROV autonome

1. Importer Pymavlink depuis le compagnon (Rpi)
2. Créer une connexion (udpout) et attendre le heartbeat

```
# Import mavutil
from pymavlink import mavutil

# Create the connection
master = mavutil.mavlink_connection('udpout:0.0.0.0:9000')

# Send a ping to start connection and wait for any reply.
wait_conn()

# Get some information !
while True:
    try:
        print(master.recv_match().to_dict())
    except:
        pass
    time.sleep(0.1)
```

Contrôle du ROV autonome

Exemple de possibilités :

- Changement du mode de navigation (manuel, auto, etc.)

```
# Choose a mode
mode = 'STABILIZE'

# Check if mode is available
if mode not in master.mode_mapping():
    print('Unknown mode : {}'.format(mode))
    print('Try:', list(master.mode_mapping().keys()))
    sys.exit(1)

# Get mode ID
mode_id = master.mode_mapping()[mode]
# Set new mode
master.mav.set_mode_send(
    master.target_system,
    mavutil.mavlink.MAV_MODE_FLAG_CUSTOM_MODE_ENABLED,
    mode_id)

while True:
    # Wait for ACK command
    ack_msg = master.recv_match(type='COMMAND_ACK', blocking=True)
    ack_msg = ack_msg.to_dict()

    # Check if command in the same in `set_mode`
    if ack_msg['command'] !=
mavutil.mavlink.MAVLINK_MSG_ID_SET_MODE:
        continue

    # Print the ACK result !
    print(mavutil.mavlink.enums['MAV_RESULT'][ack_msg['re-
sult']].description)
    break
```

Contrôle du ROV autonome

Exemple de possibilités :

- Armer / Désarmer le ROV

```
# Arm
# master.arducopter_arm() or:
master.mav.command_long_send(
    master.target_system,
    master.target_component,
    mavutil.mavlink.MAV_CMD_COMPONENT_ARM_DISARM,
    0,
    1, 0, 0, 0, 0, 0, 0)

# Disarm
# master.arducopter_disarm() or:
master.mav.command_long_send(
    master.target_system,
    master.target_component,
    mavutil.mavlink.MAV_CMD_COMPONENT_ARM_DISARM,
    0,
    0, 0, 0, 0, 0, 0, 0)
```

Contrôle du ROV autonome

Exemple de possibilités :

- Envoyer des commandes manuelles

```
# Create a function to send RC values
# More information about Joystick channels
# here: https://www.ardusub.com/operators-manual/rc-input-and-output.html#rc-inputs
def set_rc_channel_pwm(channel_id, pwm=1500):
    """ Set RC channel pwm value
    Args:
        channel_id (TYPE): Channel ID
        pwm (int, optional): Channel pwm value 1100-1900
    """
    if channel_id < 1 or channel_id > 18:
        print("Channel does not exist.")
        return

    # Mavlink 2 supports up to 18 channels:
    # https://mavlink.io/en/messages/common.html#RC_CHANNELS_OVERRIDE
    rc_channel_values = [65535 for _ in range(18)]
    rc_channel_values[channel_id - 1] = pwm
    master.mav.rc_channels_override_send(
        master.target_system,                # target_system
        master.target_component,             # target_component
        *rc_channel_values)                  # RC channel list, in microseconds.
```

Contrôle du ROV autonome

Exemple de possibilités :

- Envoyer des commandes manuelles

```
# Set some roll
set_rc_channel_pwm(2, 1600)

# Set some yaw
set_rc_channel_pwm(4, 1600)

# Set channel 12 to 1500us
# This can be used to control a device connected to a servo output by setting the
# SERVO[N]_Function to RCIN12 (Where N is one of the PWM outputs)
set_rc_channel_pwm(12, 1500)
```

```
# Create a function to send RC values
# More information about Joystick channels
# here: https://www.ardusub.com/operators-manual/rc-input-and-output.html#rc-inputs
def set_rc_channel_pwm(channel_id, pwm=1500):
    """ Set RC channel pwm value
    Args:
        channel_id (TYPE): Channel ID
        pwm (int, optional): Channel pwm value 1100-1900
    """
    if channel_id < 1 or channel_id > 18:
        print("Channel does not exist.")
        return

    # Mavlink 2 supports up to 18 channels:
    # https://mavlink.io/en/messages/common.html#RC_CHANNELS_OVERRIDE
    rc_channel_values = [65535 for _ in range(18)]
    rc_channel_values[channel_id - 1] = pwm
    master.mav.rc_channels_override_send(
        master.target_system,                # target_system
        master.target_component,             # target_component
        *rc_channel_values)                  # RC channel list, in microseconds.
```

Contrôle du ROV autonome

Exemple de possibilités :

- Récupération de tous les paramètres

```
# Request all parameters
master.mav.param_request_list_send(
    master.target_system, master.target_component
)
while True:
    time.sleep(0.01)
    try:
        message = master.recv_match(type='PARAM_VALUE', blocking=True).to_dict()
        print('name: %s\tvalue: %d' % (message['param_id'],
                                       message['param_value']))
    except Exception as error:
        print(error)
        sys.exit(0)
```

Contrôle du ROV autonome

+ d’exemples

Gestion des axes de déplacements (RC inputs)