Tello Driver V

Pedro Arias

Novedades TelloDriver

- Topics
- Mask
- Coordination Frame
- Control por velocidad
- Estudio latencia

PoseStamped

position.x = self.x
position.y = self.y
position.z = h
orientation.x = qx
orientation.y = qy
orientation.z = qz
orientation.w = qw

```
mask = msg.type_mask
mask = "{0:012b}".format(int(mask))
if not bool(int(mask[-1])):...
if not bool(int(mask[-2])):...
if not bool(int(mask[-3])):...
if not bool(int(mask[-4])):...
if not bool(int(mask[-5])):...
if not bool(int(mask[-6])):...
if not bool(int(mask[-7])):...
if not bool(int(mask[-8])):...
if not bool(int(mask[-9])):...
if not bool(int(mask[-10])):...
if not bool(int(mask[-11])):...
if not bool(int(mask[-12])):...
```

```
uint8 coordinate_frame
uint8 FRAME_LOCAL_NED = 1
uint8 FRAME_LOCAL_OFFSET_NED = 7
uint8 FRAME_BODY_NED = 8
uint8 FRAME_BODY_OFFSET_NED = 9
```

⇒ **FRAME_LOCAL_FRD**

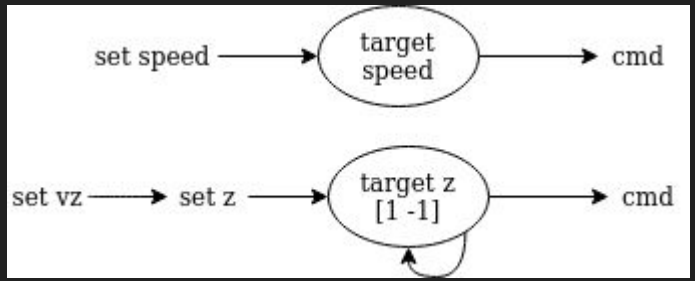
⇒ **FRAME_BODY_FRD**

¿OFFSET?

Timer: cmd vel

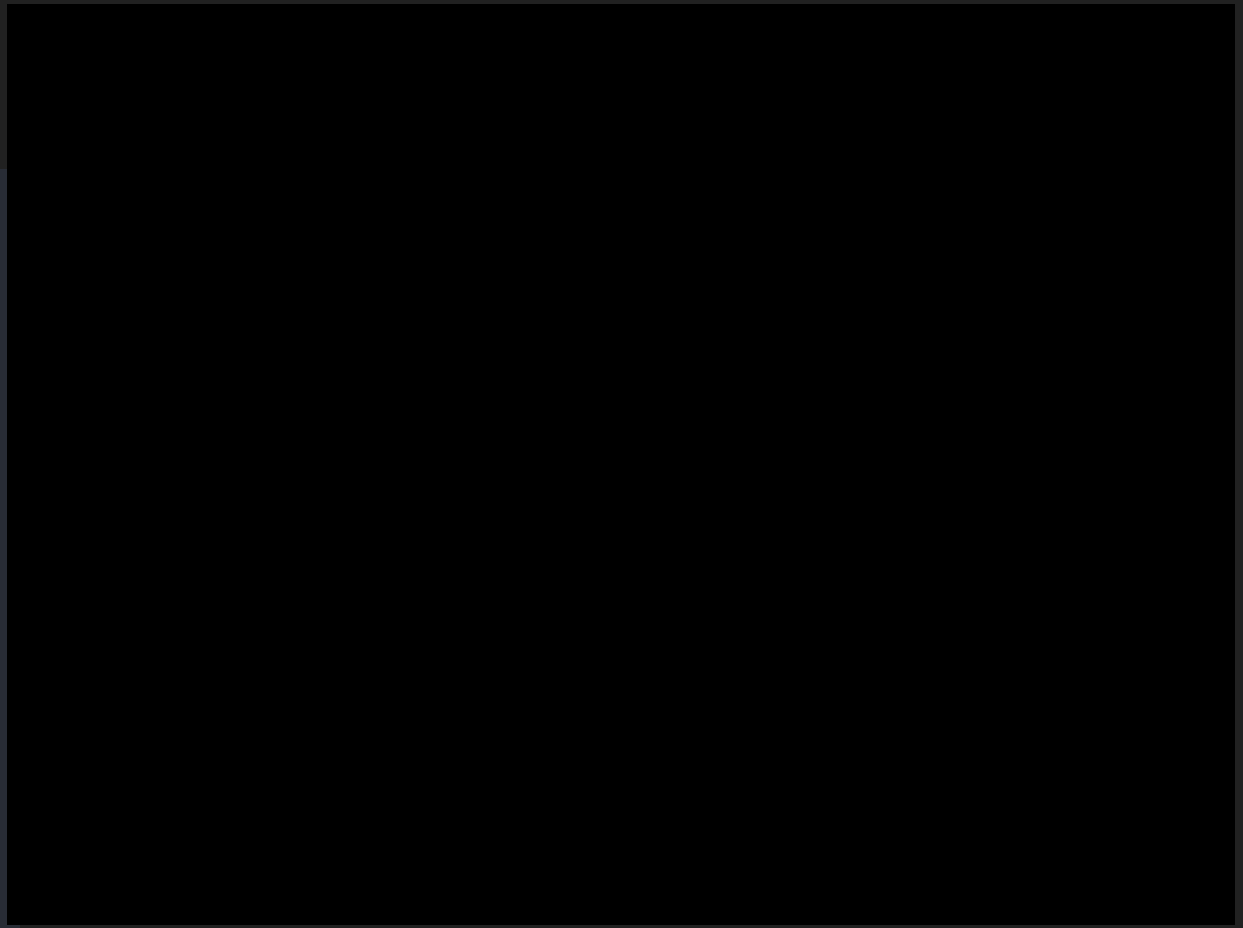
```
if not bool(int(mask[-11])):
    if is_abs:
        target_yaw = degrees(msg.yaw)
        yaw = target_yaw - self.__yaw
        self.__yaw = target_yaw
    else:
        target_yaw = degrees(msg.yaw)
        yaw = target_yaw
        self.__yaw += target_yaw

    if yaw > 0:
        self.__send_cmd("cw {}".format(abs(yaw)), False)  # degrees
    elif yaw < 0:
        self.__send_cmd("ccw {}".format(abs(yaw)), False)  # degrees
    else:
        pass  # already at target yaw
        # print("Already at target yaw")
if not bool(int(mask[-12])):
    yaw_rate = degrees(msg.yaw_rate)
    self.__yaw_rate = yaw_rate
    self.__hold_vel()
```

```
def __send_fake_vel(self, event):
    vx = self.__vx
    vy = self.__vy
    vz = self.__vz
    yaw_rate = radians(self.__yaw_rate)

    mask = 3064 + 1 if vx == 0 else 0 + 2 if vy == 0 else 0 + 4 if vz == 0 else 0 + 1024 if yaw_rate == 0 else 0
    if mask == 4095:
        self.fake_vel_timer.shutdown()
        self.fake_vel_status = False
        return
    setpoint = PositionTarget()
    setpoint.coordinate_frame = 12
    setpoint.type_mask = mask
    setpoint.position.x = vx
    setpoint.position.y = vy
    setpoint.position.z = vz
    setpoint.yaw = yaw_rate
    self.fake_vel_pub.publish(setpoint)

def __hold_vel(self):
    if not self.fake_vel_status:
        self.fake_vel_timer = rospy.Timer(rospy.Duration(secs=1), self.__send_fake_vel)
        self.fake_vel_status = True
```

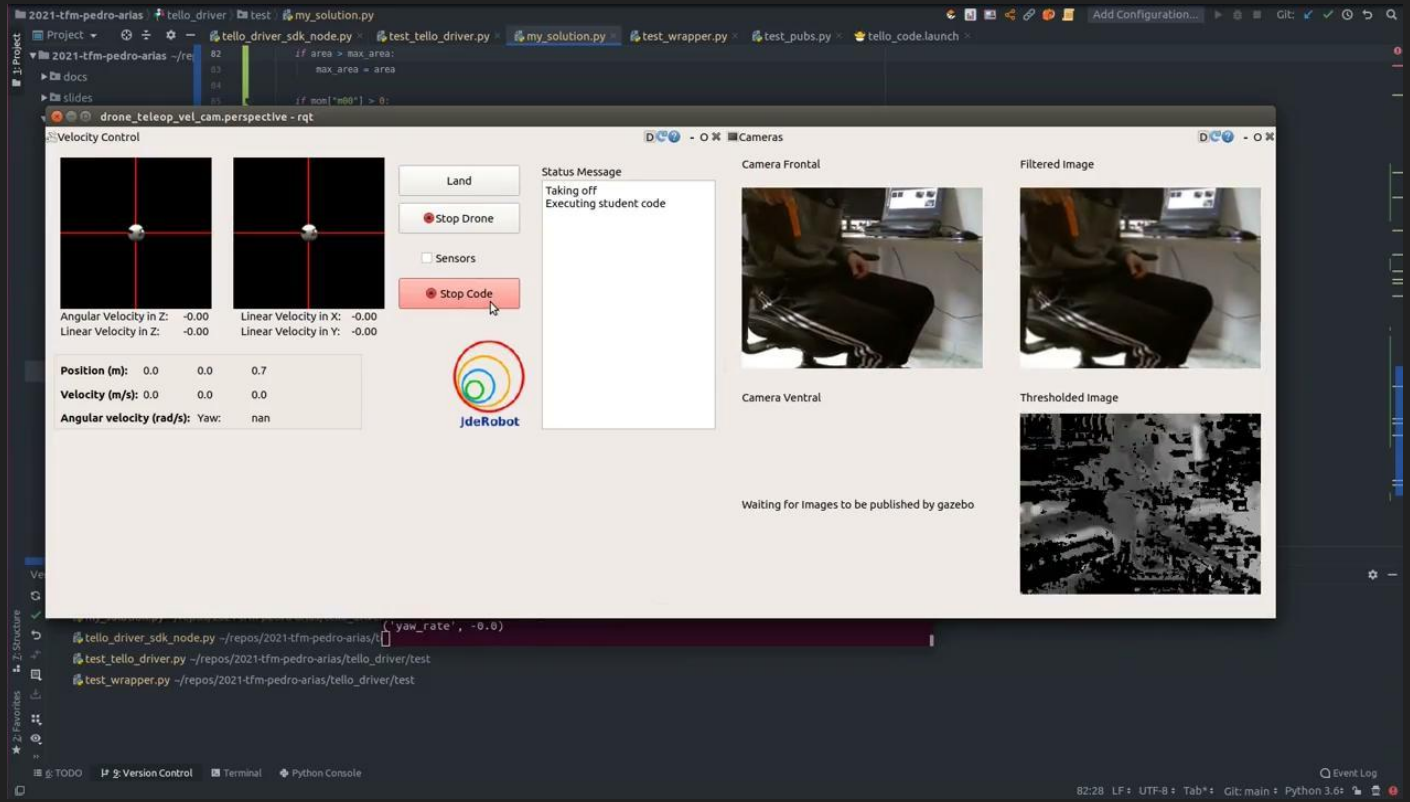


Latencia

```
msg = PositionTarget()
msg.coordinate_frame = 8
msg.type_mask = 3064  # xyz yaw

msg.position.x = 0
msg.position.y = 0
msg.position.z = 0
msg.yaw = 2

for i in range(1000):
    setpoint_raw_publisher.publish(msg)
    time.sleep(0.01)
time.sleep(1)

for i in range(100):
    setpoint_raw_publisher.publish(msg)
    time.sleep(0.1)
time.sleep(1)

for i in range(10):
    setpoint_raw_publisher.publish(msg)
    time.sleep(1)
time.sleep(1)
```



Filtro color + control en guiñada

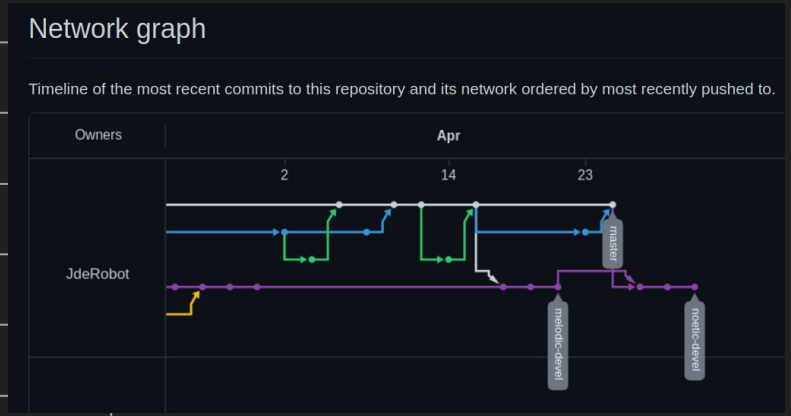


DroneWrapper

- Eliminada dependencia de gazebo_ros [1]
- Versión Noetic

/ros/rosdistro

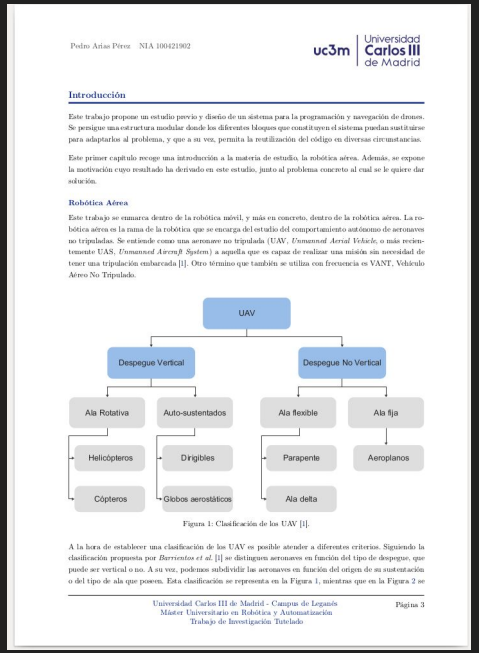
cv_bridge	converts between ROS Image messages and OpenCV images.
geometry_msgs	msgs for geometric primitives such as points, vectors and poses.
mavros_msgs	messages for MAVROS.
rospy	Python client library for ROS.
sensor_msgs	msgs for commonly used sensors: cameras.
tf	keep track of multiple coordinate frames over time.
mavros	MAVLink extendable communication node for ROS with proxy for GCS.



Trabajo de Investigación Tutelado

- Introducción
  - Robótica Aérea
  - Motivación
  - Problema
- Estado del Arte
  - Herramientas
  - Plataformas aéreas

- Github [1]
- Drive



Pedro Arias Pérez NIA 100421902

uc3m | Universidad Carlos III de Madrid

Introducción

Este trabajo propone un estudio previo y diseño de un sistema para la programación y navegación de drones. Se persigue una estructura modular donde los diferentes bloques que constituyen el sistema puedan sustituirse para adaptarlos al problema, y que a su vez, permita la reutilización del código en diversas circunstancias.

Este primer capítulo recoge una introducción a la materia de estudio, la robótica aérea. Además, se expone la motivación cuyo resultado ha derivado en este estudio, junto al problema concreto al cual se le quiere dar solución.

Robótica Aérea

Este trabajo se enmarca dentro de la robótica móvil, y más en concreto, dentro de la robótica aérea. La robótica aérea es la rama de la robótica que se encarga del estudio del comportamiento autónomo de aeronaves no tripuladas. Se entiende como una aeronave no tripulada (UAV, *Unmanned Aerial Vehicle*, o más recientemente UAS, *Unmanned Aircraft System*) a aquella que es capaz de realizar una misión sin necesidad de tener una tripulación embarcada [1]. Otro término que también se utiliza con frecuencia es VANT, Vehículo Aéreo No Tripulado.

Figura 1: Clasificación de los UAV [1].

A la hora de establecer una clasificación de los UAV es posible atender a diferentes criterios. Siguiendo la clasificación propuesta por *Barrientos et al.* [1] se distinguen aeronaves en función del tipo de despegue, que puede ser vertical o no. A su vez, podemos subdividir las aeronaves en función del origen de su sustentación o del tipo de ala que posean. Esta clasificación se representa en la Figura 1, mientras que en la Figura 2 se