드론 세미나 4회차

HandS 2022년 1학기 드론 세미나

목차

- Position Control
- Velocity Control
- Multithreading
- OpenCV

Position Control

- vehicle.simple_goto()
  - dronekit.LocationGlobalRelative class의 instance
  - groundspeed(수평방향 속력)
- haversine 모듈로 (위도, 경도) 좌표를 이용하여 남은 거리 계산

```python
def goto_position(vehicle, targetLocation, groundspeed = 1):
    ########## How to use haversine ##########
    # currentLocation = vehicle.location.global_relative_frame
    # a = (currentLocation.lat, currentLocation.lon)
    # b = (targetLocation.lat, targetLocation.lon)
    # targetDistance = haversine(a, b, unit = 'm')   # distance between the current position and target position

    for key, value in point_dict.items():
        if targetLocation == value:
            target = key
            break
        else:
            target = 'target not in point_dict'

    print("Heading to the target location")
    print("Target:", target)
    print("Target location: lat={}, lon={}".format(targetLocation.lat, targetLocation.lon))

    print("Groundspeed set to {} m/s".format(groundspeed))
    vehicle.simple_goto(targetLocation, groundspeed=groundspeed)

    while vehicle.mode.name=="GUIDED": # Halt if no longer in guided mode.
        b = (targetLocation.lat, targetLocation.lon)
        c = (vehicle.location.global_relative_frame.lat, vehicle.location.global_relative_frame.lon)
        remainingDistance = haversine(c, b, unit = 'm')
        print("\tDistance to target: {:.2f}m".format(remainingDistance))

        if remainingDistance <= 0.5: # Just below target, in case of undershoot.
            print("Reached target\n\n")
            break

        time.sleep(1)
```

Position Control

- `vehicle.message_factory.set_position_target_local_ned_encoder()`
  - MAVLink 프로토콜에 따라서 드론에게 내릴 명령을 binary code로 encoding 해줌
- 변위 벡터로 드론을 이동시킴

```python
def goto_position_target_local_ned(vehicle, north, east, down):
    msg = vehicle.message_factory.set_position_target_local_ned_encode(
        0,       # time_boot_ms (not used)
        0, 0,    # target system, target component
        mavutil.mavlink.MAV_FRAME_BODY_OFFSET_NED, # frame
        # mavutil.mavlink.MAV_FRAME_BODY_FRD,   # for Copter versions released after 2019-08
        # https://mavlink.io/en/messages/common.html#MAV_FRAME_BODY_OFFSET_NED
        0b0000111111111000, # type_mask (only positions enabled)
        north, east, down, # x, y, z positions (or North, East, Down in the MAV_FRAME_BODY_NED frame)
        0, 0, 0, # x, y, z velocity in m/s  (not used)
        0, 0, 0, # x, y, z acceleration (not supported yet, ignored in GCS_Mavlink)
        0, 0)    # yaw, yaw_rate (not supported yet, ignored in GCS_Mavlink)
    # send command to vehicle
    vehicle.send_mavlink(msg)
```

https://dronekit-python.readthedocs.io/en/latest/guide/copter/guided_mode.html
=> Guided mode commands 부분 확인

Position Control

- 드론 작동 시 최초로 GPS 신호를 포착한 장소의 좌표는 home location으로 저장됨
- RTL 모드로 전환하면 자동으로 home location으로 복귀

```python
def rtl(vehicle):
    print ("Returning home")
    vehicle.parameters['RTL_ALT'] = 0
    time.sleep(1)

    vehicle.mode = VehicleMode("RTL")
    while vehicle.mode.name != 'RTL':
        vehicle.mode = VehicleMode("RTL")
        time.sleep(1)

    print("Mode: {}".format(vehicle.mode.name))

    while vehicle.mode.name == "RTL":
        h = (vehicle.home_location.lat, vehicle.home_location.lon)
        c = (vehicle.location.global_relative_frame.lat, vehicle.location.global_relative_frame.lon)
        remainingDistance = haversine(c, h, unit = 'm')
        print("\tDistance to target: {:.2f}m".format(remainingDistance))

        if remainingDistance <= 0.5: # Just below target, in case of undershoot.
            print("Arrived home")
            break

        time.sleep(1)

    while vehicle.location.global_relative_frame.alt - vehicle.home_location.alt > 0.5:
        print("\tLanding...")
        time.sleep(1)

    print("Vehicle on the ground")
    print("Close vehicle object")
    vehicle.close()
```

Velocity Control

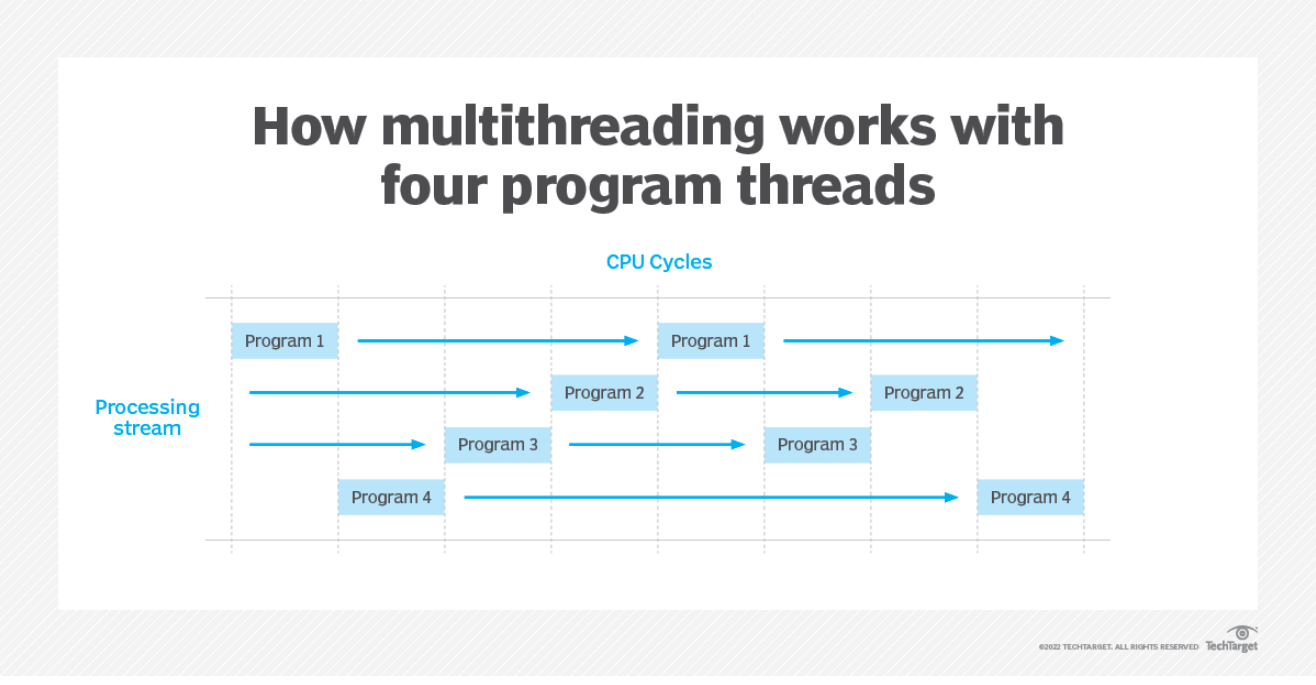
- vehicle.message_factory.set_position_target_local_ned_encoder()
- 속도 벡터로 드론을 이동시킴

```python
def send_ned_velocity(vehicle, velocity_x, velocity_y, velocity_z, duration):
    # Set up velocity mappings
    # velocity_x > 0 => fly North
    # velocity_x < 0 => fly South
    # velocity_y > 0 => fly East
    # velocity_y < 0 => fly West
    # velocity_z < 0 => ascend
    # velocity_z > 0 => descend

    msg = vehicle.message_factory.set_position_target_local_ned_encode(
        0,       # time_boot_ms (not used)
        0, 0,    # target system, target component
        mavutil.mavlink.MAV_FRAME_BODY_OFFSET_NED, # frame
        # mavutil.mavlink.MAV_FRAME_BODY_FRD,   # for Copter versions released after 2019-08
        # https://mavlink.io/en/messages/common.html#MAV_FRAME_BODY_OFFSET_NED
        0b0000111111000111, # type_mask (only speeds enabled)
        0, 0, 0, # x, y, z positions (not used)
        velocity_x, velocity_y, velocity_z, # x, y, z velocity in m/s
        0, 0, 0, # x, y, z acceleration (not supported yet, ignored in GCS_Mavlink)
        0, 0)    # yaw, yaw_rate (not supported yet, ignored in GCS_Mavlink)

    msg_immediate_stop = vehicle.message_factory.set_position_target_local_ned_encode(
        0,       # time_boot_ms (not used)
        0, 0,    # target system, target component
        mavutil.mavlink.MAV_FRAME_BODY_OFFSET_NED, # frame
        # mavutil.mavlink.MAV_FRAME_BODY_FRD,   # for Copter versions released after 2019-08
        # https://mavlink.io/en/messages/common.html#MAV_FRAME_BODY_OFFSET_NED
        0b0000111111000111, # type_mask (only speeds enabled)
        0, 0, 0, # x, y, z positions (not used)
        0, 0, 0, # x, y, z velocity in m/s
        0, 0, 0, # x, y, z acceleration (not supported yet, ignored in GCS_Mavlink)
        0, 0)    # yaw, yaw_rate (not supported yet, ignored in GCS_Mavlink)

    # The message is re-sent every second during the specified duration.
    # From Copter 3.3 the vehicle will stop moving if a new message is not received in approximately 3 seconds.
    for x in range(0, duration):
        vehicle.send_mavlink(msg)
        time.sleep(1)

    # Send a stop messeage twice in case of not being received.
    vehicle.send_mavlink(msg_immediate_stop)
    vehicle.send_mavlink(msg_immediate_stop)
```

Multithreading

http://pythonstudy.xyz/python/article/24-%EC%93%B0%EB%A0%88%EB%93%9C-Thread
https://wikidocs.net/82581
https://eunjinii.tistory.com/m/41
https://murphymoon.tistory.com/entry/%EB%A9%80%ED%8B%B0%ED%94%84%EB%A1%9C%EC%84%B8%EC%8B%B1multiprocessing%EA%B3%BC-%EB%A9%80%ED%8B%B0%EC%8A%A4%EB%A0%88%EB%94%A9multithreading%EC%9D%98-%EC%B0%A8%EC%9D%B4%EC%A0%90-OS-%EB%A9%B4%EC%A0%91%EC%A7%88%EB%AC%B8-2
https://www.youtube.com/watch?v=QmtYKZC0lMU

- 프로그램을 병렬적으로 실행하는 방법
- OpenCV는 while loop로 비디오 프레임을 종료 시점까지 계속 불러와서 처리하므로 dronekit-python이 실행 될 여유가 없음
- dronekit-python thread와 OpenCV thread를 병렬적으로 실행해야 함



Multithreading

- threading 모듈 사용
- 쓰레드에서 실행할 작업을 미리 함수화해서 `threading.Thread()` 함수의 `target`이라는 파라미터에 입력
- 쓰레드 생성 후 start로 쓰레드 실행

```python
import threading
import time

def thread_1():
    print("thread_1 forks")
    for i in range(10):
        print("thread_1: ", i)
        time.sleep(0.5)
    print("thread_1 done")

def thread_2():
    print("thread_2 forks")
    for i in range(10):
        print("thread_2: ", i)
        time.sleep(0.5)
    print("thread_2 done")

t1 = threading.Thread(target=thread_1)
t2 = threading.Thread(target=thread_2)

# fork
print("main thread forks")
t1.start()
t2.start()
print("main thread done")
```

Multithreading

- 기본적으로 메인 쓰레드는 자신의 작업이 다 끝나도 서브 쓰레드의 작업이 모두 종료될 때 까지 기다림

```
main thread forks
thread_1 forks
thread_1:  0
thread_2 forks
main thread done
thread_2:  0
thread_1:  1
thread_2:  1
thread_2:  2
thread_1:  2
thread_2:  3
thread_1:  3
thread_1:  4
thread_2:  4
thread_1:  5
thread_2:  5
thread_1:  6
thread_2:  6
thread_1:  7
thread_2:  7
thread_1:  8
thread_2:  8
thread_1:  9
thread_2:  9
thread_1 done
thread_2 done
```

Multithreading

- 데몬(daemon) 쓰레드
  - 메인 쓰레드가 종료될 때 자신의 실행 상태와 상관없이 종료되는 서브 쓰레드
  - daemon=True로 설정

```python
import threading
import time

def thread_1():
    print("thread_1 forks")
    for i in range(10):
        print("thread_1: ", i)
        time.sleep(0.5)
    print("thread_1 done")

def thread_2():
    print("thread_2 forks")
    for i in range(5):
        print("thread_2: ", i)
        time.sleep(0.5)
    print("thread_2 done")

t1 = threading.Thread(target=thread_1, daemon=True)
t2 = threading.Thread(target=thread_2)

# fork
print("main thread forks")
t1.start()
t2.start()
print("main thread done")
```

Multithreading

- 메인 쓰레드는 thread_2가 끝날 때 까지 기다렸다가 종료
- thread_1의 종료 여부는 고려하지 않음

```
main thread forks
thread_1 forks
thread_1:  0
thread_2 forks
thread_2:  0
main thread done
thread_1:  1
thread_2:  1
thread_1:  2
thread_2:  2
thread_2:  3
thread_1:  3
thread_1:  4
thread_2:  4
thread_1:  5
thread_2 done
```

Multithreading

- 모든 쓰레드가 종료 후 그 결과를 모아서 활용해야 한다면 or 모든 쓰레드가 종료되고 나서 실행해야 할 코드가 있다면 join()을 사용해야함

```python
import threading
import time

def thread_1():
    print("thread_1 forks")
    for i in range(10):
        print("thread_1: ", i)
        time.sleep(0.5)
    print("thread_1 done")

def thread_2():
    print("thread_2 forks")
    for i in range(5):
        print("thread_2: ", i)
        time.sleep(0.5)
    print("thread_2 done")

t1 = threading.Thread(target=thread_1)
t2 = threading.Thread(target=thread_2)

# fork
print("main thread forks")
t1.start()
t2.start()
print("main thread done")

t1.join()
t2.join()

print("program successfully finished...")
```

Multithreading

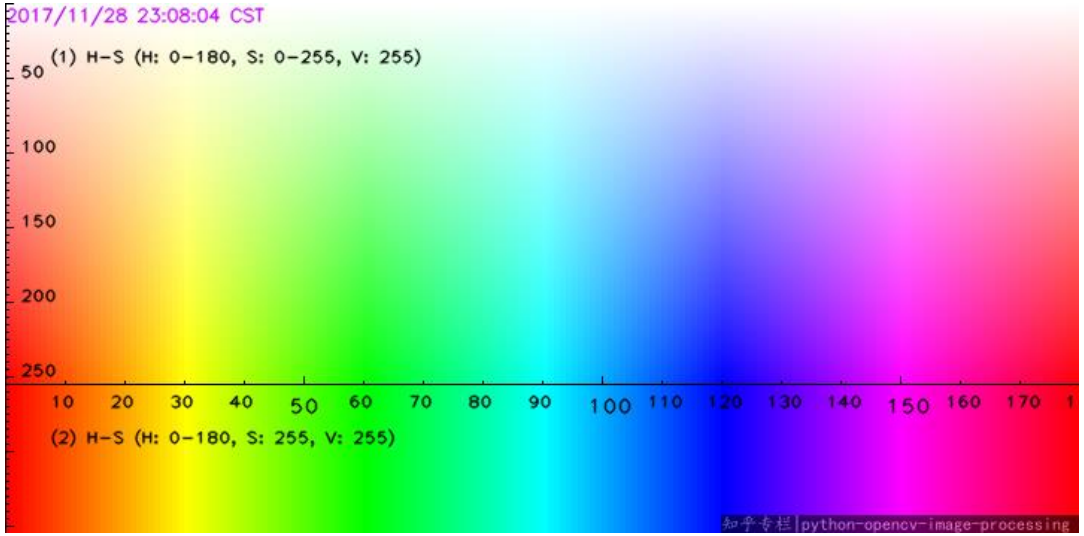
- 모든 쓰레드가 `join()` 하고 그 이후의 코드가 실행됨

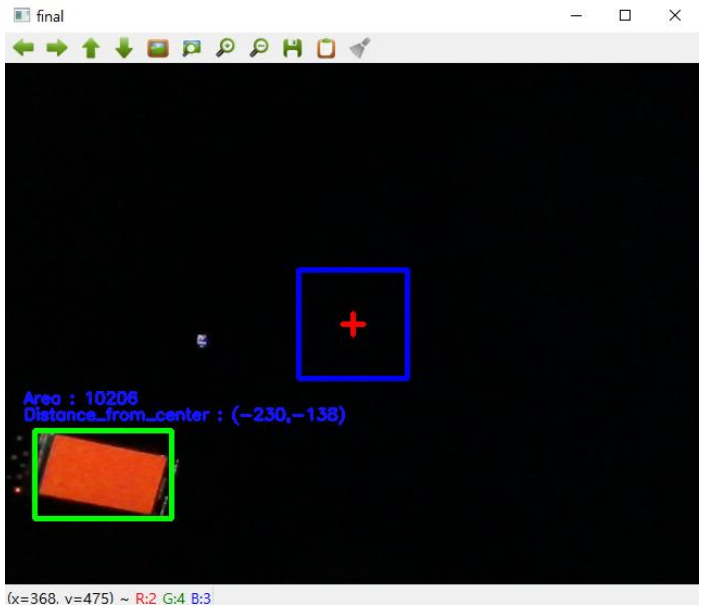
```
main thread forks
thread_1 forks
thread_1:  0
thread_2 forks
main thread done
thread_2:  0
thread_2:  1
thread_1:  1
thread_1:  2
thread_2:  2
thread_1:  3
thread_2:  3
thread_2:  4
thread_1:  4
thread_1:  5
thread_2 done
thread_1:  6
thread_1:  7
thread_1:  8
thread_1:  9
thread_1 done
program successfully finished...
```

사진 출처: https://cvexplained.wordpress.com/2020/04/28/color-detection-hsv/

OpenCV

- 설치
  - `conda install -c conda-forge opencv`
  - `pip install opencv-python`
- 예제 코드는 빨간색 ~ 주황색 물체를 탐지하도록 설정
  - `u_b, l_b` 조정해서 다른 색 탐지 가능





과제

- OpenCV 윈도우 내에서 주황색 물체의 위치에 반응하는 드론 프로그램 만들기

- 전역 변수 or queue 사용
  - https://infinity-infor-age.tistory.com/entry/python-inter-thread-comm