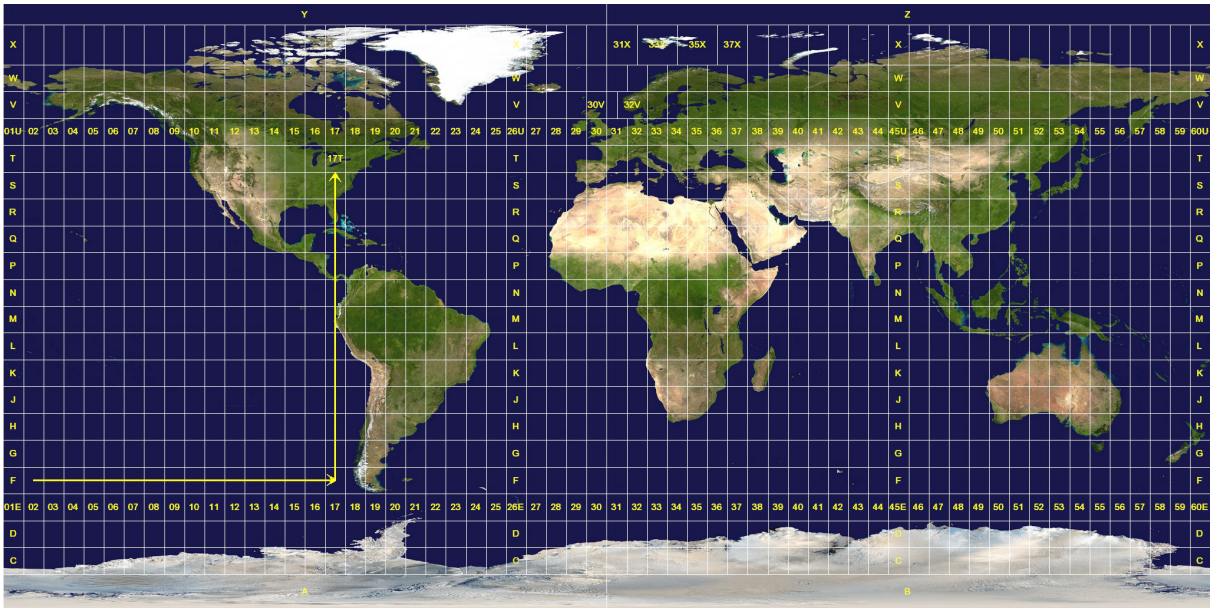
NMEA → UTM(meter)

환경 : python, ublox_gps, 9-axis IMU

UTM 좌표란?



위의 그림과 같이 지구를 격자로 쪼개어 구역으로 나누어 둔 좌표

NMEA(latitude, longitude) → meter 좌표계 변환에 **pyproj** 라이브러리 사용

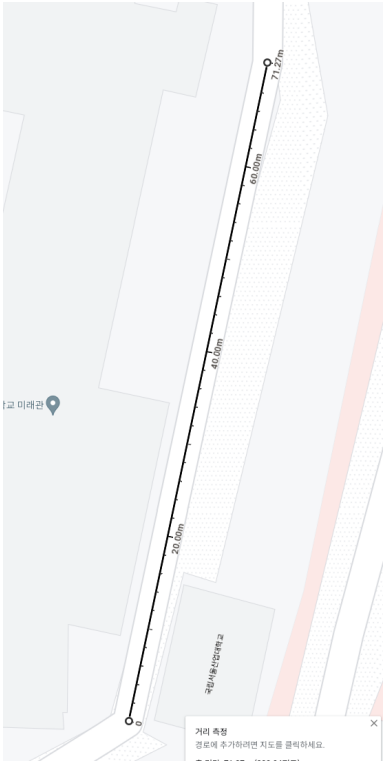
PyProj 사용법

```
from pyproj import Transformer

latitude = 37.630054799999996
longitude = 127.0775683

transformer = Transformer.from_crs('EPSG:4326', 'EPSG:32752') # EPSG:4326 = GPS 좌표계, EPSG:32752 = 서울의 UTM 좌표
x, y = transformer.transform(latitude, longitude)
print(x, y)

#result = 330363.09393319703 14166508.085438043
```

예제 :

```
from pyproj import Transformer
import numpy as np

latitude1 = 37.629620
```

```
longitude1 = 127.081801

latitude2 = 37.628986
longitude2 = 127.081633

transformer = Transformer.from_crs('EPSG:4326', 'EPSG:32752')
x1, y1 = transformer.transform(latitude1, longitude1)
x2, y2 = transformer.transform(latitude2, longitude2)

d1 = x1-x2
d2 = y1-y2

print(np.sqrt(d1**2+d2**2))
```



좌측 그림과 같이 2 점의 거리 측정

윗점 : (37.629620, 127.081801)

아랫점 : (37.628986, 127.081633)

결과 :

E1, N1 = 330735.6400791636, 14166452.192916803

E2, N2 = 330719.37547313655, 14166382.146788798

$$d = \sqrt{(E1 - E2)^2 + (N1 - N2)^2)} = 71.9 \text{ (m)}$$

UTM → Robot_Local

UTM 좌표계를 (N, W, Z)로 설정

초기값 : (N_0, W_0, Z_0)[GPS 값], $\vec{q_0}$ [IMU 값]

IMU의 회전 역시 (N, W, Z) 좌표계를 사용하므로(my_ahrs+ 기준), 초기 쿼터니안 $\vec{q_0}$을 UTM과 robot 좌표계 사이의 회전으로 표현할 수 있음.

그러면, 초기값과 위경도의 차이 $(N - N_0, W - W_0, Z - Z_0)$을 $\vec{q_0}$에서 구해진 회전행렬 R_0의 역행렬에 곱하면 Robot 좌표계에서의 (x, y, z)좌표값을 구할 수 있다.

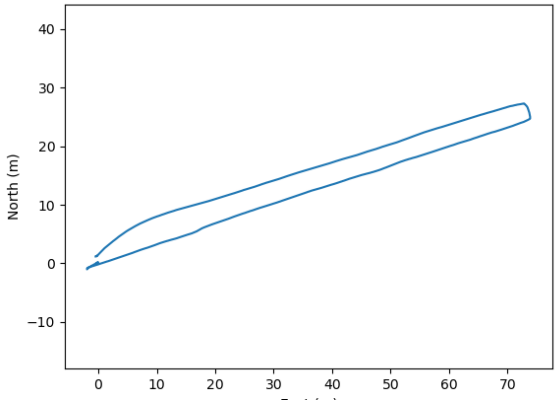
예시 :

```
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import NavSatFix
from pyproj import Transformer
from sensor_msgs.msg import Imu
import numpy as np
from tf_transformations import euler_from_quaternion
from numpy.linalg import inv
from scipy.spatial.transform import Rotation

class FixtoUtmNode(Node):
    def __init__(self):
        super().__init__('fixtoutm')
        self.subscription = self.create_subscription(
            NavSatFix,
            '/ublox_gps_node/fix',
            self.fix_callback,
            10)

        self.imu_subscription = self.create_subscription(
            Imu,
            '/imu/data',
            self.imu_callback,
            10
        )

        self.positions = np.empty((0,3))
        self.init_east = 0
        self.init_north = 0
        self.init_alt = 0
        self.relative_matrix = 0
        self.is_fix_init = True
        self.is_imu_init = True


    def fix_callback(self, msg):
        if (self.is_imu_init == False):
            latitude = msg.latitude
            longitude = msg.longitude
            altitude = msg.altitude

            utm_e, utm_n = self.lat_lon_to_utm(latitude, longitude)

            if (self.is_fix_init == True):
                self.init_east = utm_e
                self.init_north = utm_n
                self.init_alt = altitude
                self.is_fix_init=False

            diff_e = -(utm_e-self.init_east)
            diff_n = utm_n-self.init_north
            diff_a = altitude-self.init_alt

            pos_array = np.array([diff_n, diff_e, diff_a])


            filtered_position = inv(self.relative_matrix) @ pos_array
            print(filtered_position)

            # print('utm_e : ', utm_e, ', diff_n : ', diff_n, ', alt : ', altitude)
            # print('diff_e : ', diff_e, ', diff_n : ', diff_n, ', diff_a : ', diff_a)
            self.positions = np.vstack((self.positions, filtered_position))

    def lat_lon_to_utm(self, latitude, longitude):
        transformer = Transformer.from_crs('EPSG:4326', 'EPSG:32752')
        utm_easting, utm_northing = transformer.transform(latitude, longitude)
        return utm_easting, utm_northing
```

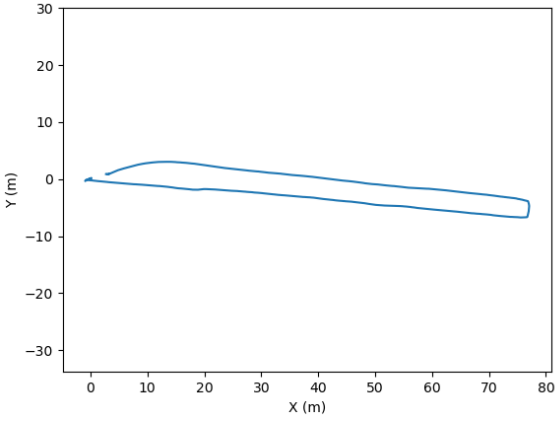
```
    def imu_callback(self, msg):
        if (self.is_imu_init == True):
            q_x = msg.orientation.x
            q_y = msg.orientation.y
            q_z = msg.orientation.z
            q_w = msg.orientation.w

            self.relative_matrix = Rotation.from_quat([q_x, q_y, q_z, q_w]).as_matrix()
            # print(self.relative_matrix)
            self.is_imu_init=False

            self.timer = self.create_timer(1.0, self.timer_callback)

    def timer_callback(self):
        self.timer.cancel()

def main(args=None):
    rclpy.init(args=args)
    fixtutm = FixtoUtmNode()

    try:
        rclpy.spin(fixtutm)
    except KeyboardInterrupt:
        pass

    np.save('UTM_position-0911', fixtutm.positions)
    print('Save Complete!')

    fixtutm.destroy_node()


if __name__ == '__main__':
    main()
```

1. ***imu_callback*** 함수에서 초기 heading 값을 통해 UTM→robot_local사이의 변환행렬 R을 구한다.
2. ***fix_callback*** 에서 GPS data를 받아, 이를 ***lat_lon_to_utm*** 함수에서 UTM 좌표계로 변경하고 초기 UTM 좌표와의 차이값을 계산한다.
3. 이후 이 차이값에 위에서 구한 R의 역행렬에 곱해주면 robot_local좌표계에서의 위치를 구할 수 있음.

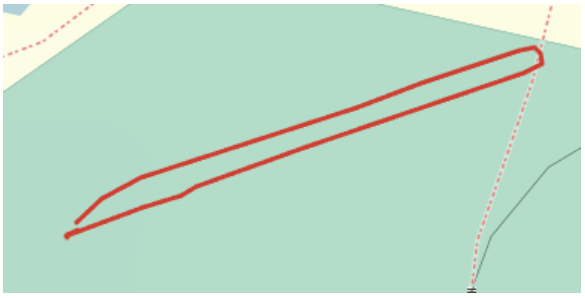
예시 결과



변환 전



변환 후 (초기 시작이 살짝 틀어져 정확히 X방향으로 움직이지는 않음)



GPX파일로 만든 궤적