Annex 1: Flying Eye Code

```
import asyncio
import picamera
import os
import cv2
import numpy as np

from mavsdk import System

async def read_mission_data():

    global mission1_data     # Mission for road control
    global mission3_data     # Mission for return to base station
    rows, cols = (10, 10)
    global mission2_data = [[0] * cols] * rows       # Mission for take pictures
    global photo_loc_count

    f = open('/home/pi/yol_kontrol_mission.txt', 'r')
    content = f.readline()
    mission1_data = content.split(",")
    f.close()

    i = 0
    f = open('/home/pi/yol_foto_mission.txt', 'r')
    while True:
        content = f.readline()
        if content == 'EOF':
            break
        mission2_data[i] = content.split(",")
        i = i + 1
    f.close()
    photo_loc_count = i

    f = open('/home/pi/donus_mission.txt', 'r')
    content = f.readline()
    mission3_data = content.split(",")
    f.close()

async def connect_and_startup_controls():

    # Init the drone
    drone = System(mavsdk_server_address='127.0.0.1', port=50051)
    #await drone.connect(system_address="serial:///dev/ttyACM0")
    await drone.connect()
    print("waiting drone to connect ...", flush=True)

    async for state in drone.core.connection_state():
        if state.is_connected:
            print(f"Drone discovered with UUID: {state.uuid}", flush=True)
            break

    #  Global Position Estimate kontrolu
    print("Waiting GPS home position", flush=True)
    async for global_lock in drone.telemetry.health():
        if global_lock.is_home_position_ok == True:
            break
        await asyncio.sleep(2)
```

```
    # GPS kontrolu ( en az 14 uydu gorene kadar bekle )
    print("Waiting GPS Satellites", flush=True)
    gps_sat__num = await check_gps_info(drone)
    print(gps_sat__num)

    return drone

async def check_gps_info(drone):
    async for gps_info in drone.telemetry.gps_info():
        if gps_info.num_satellites >= 14:
            return gps_info.num_satellites
        await asyncio.sleep(1)

    return drone


async def get_telemetry_pos(drone):
        async for position in drone.telemetry.position():
            return position

#
#      Picamera ile 1024 x 768 pixel fotograf cekimi
#
#      type = 'M' ise goruntu isleme icin fotograf memory stream'e alinir
#      type = 'D' ise fotograf disk'e kaydedilir.
#
async def take_photo(type,fname):

    camera = picamera.PiCamera()
    my_stream = BytesIO()
    camera.start_preview()
    #camera warm up
    await asyncio.sleep(1)
    camera.hflip = True
    camera.vflip = True
    camera.resolution = (1024,768)
    camera.exposure_mode = 'antishake'
    if type == 'M':
        camera.capture(my_stream,'jpeg')
        camera.close()
        return my_stream
    else:
        camera.capture(fname, 'jpeg')
        camera.close()
        return
    return my_stream

async def get_telemetry_bat(drone):
    async for position in drone.telemetry.battery():
        return position

#
#      Problem tespit edilene kadar veya pil bitene kadar goruntu isleme
#
async def image_process(drone):

    count = 0
```

```
    return_status = False

    # Initiate SIFT detector
    sift = cv2.xfeatures2d.SIFT_create()
    stream1 = await take_photo('M')
    kp1, des1 = sift.detectAndCompute(stream1, None)
    await asyncio.sleep(1)

    while True:
        stream2 = await take_photo('M')
        # find the keypoints and descriptors with SIFT

        kp2, des2 = sift.detectAndCompute(stream2, None)

        distance = abs(len(kp1) - len(kp2))  # keypoint length distance

        if distance >= 120:
            count = 0   # resimler farkliysa counter sifirlanir
        else:
            count = count + 1  # resimler ayniysa counter artar

        stream1 = stream2
        kp1 = kp2
        await asyncio.sleep(1)

        battery = await get_telemetry_bat(drone)

        if battery.remaining_percent <= 0.30: # pil bitiyorsa donus yapilir
            return_status = False
            break

        if count >= 600 :
            return_status = True  # resimler 10 dakika boyunca ayniysa problem tespit
edilmistir
            break

    return return_status

#
#      Yol kontrol noktasina gidis ve image processing islemleri
#
async def road_control_mission(drone):

    mission_items = []
    mission_items.append(MissionItem(float(mission1_data[0]),
                                     float(mission1_data[1]),
                                      float(mission1_data[2]),
                                      float(mission1_data[3]),
                                    bool(mission1_data[4]),
                                     float(mission1_data[5]),
                                     float(mission1_data[6]),
                                     MissionItem.CameraAction.NONE,
                                     float(mission1_data[8]),
                                     float(mission1_data[9]),
                                     )

    mission_plan = MissionPlan(mission_items)
```

```
    await drone.mission.set_return_to_launch_after_mission(False)

    await drone.mission.upload_mission(mission_plan)

    await drone.action.arm()
    await asyncio.sleep(1)      # Pervaneler tam dÃ¶nmesi iÃ§in 1 sn bekle

    await drone.mission.start_mission()

    while drone.mission.is_mission_finished == False:
        await asyncio.sleep(1)

    problem_status = await image_process(drone)

    return problem_status
#
#      Donus oncesi yolun belirli noktalarindan foto cekimlerinin yapilmasi
#
async def photo_mission(drone):

    fname = '/home/pi/resim'
    mission_items = []

    for i in range(photo_loc_count):
        mission_items.append(MissionItem(float(mission2_data[i][0]),
                                         float(mission2_data[i][1]),
                                         float(mission2_data[i][2]),
                                         float(mission2_data[i][3]),
                                         bool(mission2_data[i][4]),
                                         float(mission2_data[i][5]),
                                         float(mission2_data[i][6]),
                                         MissionItem.CameraAction.NONE,
                                         float(mission2_data[i][8]),
                                         float(mission2_data[i][9]),
                                         )

        mission_plan = MissionPlan(mission_items)

        await drone.mission.set_return_to_launch_after_mission(False)

        await drone.mission.upload_mission(mission_plan)

        await drone.mission.start_mission()

        while drone.mission.is_mission_finished == False:
            await asyncio.sleep(1)

        fname = fname + str(i) + '.jpg'
        await take_photo('D',fname)

#
#      Ana istasyona donus , gorevin tamamlanmasi
#
async def return_base_station(drone):

    mission_items = []
```

```
    mission_items.append(MissionItem(float(mission3_data[0]),
                                     float(mission3_data[1]),
                                      float(mission3_data[2]),
                                      float(mission3_data[3]),
                                    bool(mission3_data[4]),
                                     float(mission3_data[5]),
                                     float(mission3_data[6]),
                                     MissionItem.CameraAction.NONE,
                                     float(mission3_data[8]),
                                     float(mission3_data[9]),
                                     )

    mission_plan = MissionPlan(mission_items)

    await drone.mission.set_return_to_launch_after_mission(False)

    await drone.mission.upload_mission(mission_plan)

    await drone.mission.start_mission()

    while drone.mission.is_mission_finished == False:
        await asyncio.sleep(1)

    await drone.action.land()

    async for in_air in drone.telemetry.in_air():
        if in_air == False:
            break

    await drone.action.disarm()

#
#      Ana Kod Kismi
#
async def run():

    await read_mission_data()    # KoordinatlarÄ± ve yukseklikleri iceren gorev
dosyalarinin okunmasi

    drone = await connect_and_startup_controls()  # Raspi - Drone baglantisi ve ilk
islemler

    problem_status = await road_control_mission(drone)  # Yol kontrol noktasina gidis ve
yol kontrol islemleri

    if problem_status == True:
        await photo_mission(drone)  # Yolda problem tespit edilmisse fotograflarin
cekilmesi

    await return_base_station(drone)  # Ana istasyona donus


if __name__ == "__main__":

    # Runs the event loop until the program completes
    loop = asyncio.get_event_loop()
    loop.run_until_complete(run())
```

```
loop.close()
```