**CODES**

**A python program to display the UAV or Drone Health status**

from dronekit import connect, VehicleMode, LocationGlobalRelative,APIException

import time

import socket

import math

import argparse

def connectMyCopter():

parser = argparse.ArgumentParser(description='Commands')

parser.add\_argument('--connect')

args = parser.parse\_args()

connection\_string = args.connect

baud\_rate=57600

vehicle = connect(connection\_string,baud=baud\_rate, wait\_ready=True)

# Get some vehicle attributes (state)

print("Get some vehicle attribute values:")

print( " GPS: %s" % vehicle.gps\_0)

print( " Battery: %s" % vehicle.battery)

print( " Last Heartbeat: %s" % vehicle.last\_heartbeat)

print( " Is Armable?: %s" % vehicle.is\_armable)

print( " System status: %s" % vehicle.system\_status.state)

print( " Mode: %s" % vehicle.mode.name ) # settable

return vehicle

vehicle=connectMyCopter()

**A python program to arm the motors of UAV or Drone**

from dronekit import connect, VehicleMode, LocationGlobalRelative,APIException

import time

import socket

import math

import argparse

def connectMyCopter():

parser = argparse.ArgumentParser(description='Commands')

parser.add\_argument('--connect')

args = parser.parse\_args()

connection\_string = args.connect

baud\_rate=57600

vehicle = connect(connection\_string,baud=baud\_rate, wait\_ready=True)

# Get some vehicle attributes (state)

print("Get some vehicle attribute values:")

print( " GPS: %s" % vehicle.gps\_0)

print( " Battery: %s" % vehicle.battery)

print( " Last Heartbeat: %s" % vehicle.last\_heartbeat)

print( " Is Armable?: %s" % vehicle.is\_armable)

print( " System status: %s" % vehicle.system\_status.state)

print( " Mode: %s" % vehicle.mode.name ) # settable

return vehicle

def arm():

parser = argparse.ArgumentParser(description='Commands')

parser.add\_argument('--connect')

args = parser.parse\_args()

connection\_string = args.connect

baud\_rate=57600

vehicle = connect(connection\_string,baud=baud\_rate, wait\_ready=True)

print ("Basic pre-arm checks")

# Don't try to arm until autopilot is ready

print ("Arming motors")

# Copter should arm in GUIDED mode

vehicle.mode = VehicleMode("STABILIZE")

vehicle.armed = True

# Confirm vehicle armed before attempting to take off

while not vehicle.armed:

print (" Waiting for arming...")

time.sleep(1)

'''vehicle=connectMyCopter()'''

arm()

**A python program to run a basic mission of takeoff and land in GUIDED Mode**

from dronekit import connect, VehicleMode, LocationGlobalRelative,APIException

import time

import socket

import math

import argparse

#Function to connect script to drone

def connectMyCopter():

parser = argparse.ArgumentParser(description='Commands')

parser.add\_argument('--connect')

args = parser.parse\_args()

connection\_string = args.connect

baud\_rate=57600

vehicle = connect(connection\_string,baud=baud\_rate, wait\_ready=True)

return vehicle

#Function to arm the drone and takeoff into the air

Def arm\_and\_takeoff(aTargetAltitude):

While not vehicle.is\_armable:

Print(“Waiting for vehicle to become armable”)

time.sleep(1)

#Switch vehicle to GUIDED mode and wait for change

vehicle.mode = VehicleMode("GUIDED")

while vehicle.mode!="GUIDED":

print("Waiting for vehicle to enter GUIDED mode")

time.sleep(1)

#Arm vehicle once GUIDED mode is confirmed

vehicle.armed=True

while vehicle.armed==False:

print("Waiting for vehicle to become armed")

time.sleep(1)

vehicle.simple\_takeoff(aTargetAltitude)

while True:

print("Current Altitude: %d"%vehicle.location.global\_relative\_frame.alt)

if vehicle.location.global\_relative\_frame.alt>=aTargetAltitude\*.95:

break

time.sleep(1)

print("Target Altitude reached")

return None

# Mission

vehicle=connectMyCopter( )

print("About to takeoff...")

vehicle.mode=VehicleMode("GUIDED")

arm\_and-takeoff(5) # telling the drone to fly 5 meters in the sky

vehicle.mode=VehicleMode("LAND")

time.sleep(2)

print("End of function")

print("Arducopter version: %s"%vehicleversion)

while True:

time.sleep(2)

vehicle.close()

#End of script

**A python program to run a mission in AUTONOMOUS Mode**

1. In this mode first the hexacopter is made STABLE with the following program:

import time

import socket

import argparse

import math

def connectMycopter():

parser = argparse.ArgumentParser(description='Commands')

parser.add\_argument('--connect')

args = parser.parse\_args()

connection\_string = args.connect

baud\_rate=57600

if not connection\_string:

import dronekit\_sitl

sitl=dronekit\_sitl.start\_default()

connection\_string=sitl.connection\_string()

vehicle = connect(connection\_string,baud=baud\_rate, wait\_ready=True)

return vehicle

def arm():

while vehicle.is\_armable!=True:

print("Waiting for vehicle to become armable.")

time.sleep(1)

print("Vehicle is armable")

# Copter should arm in STABILIZE mode

vehicle.mode=VehicleMode(“STABILIZE")

while vehicle.mode!=“STABILIZE":

print("Waiting for drone to enter STABILIZE flight mode")

time.sleep(1)

print("Vehicle in Stabilize Mode")

vehicle.armed=True

# Confirm vehicle armed before attempting to take off

while vehicle.armed=False:

print("Waiting to be armed")

time.sleep(1)

print("Vehicle is armed")

return None

def get\_distance\_meters(targetLocation,currentLocation):

dLat=targetLocation.lat - currentLocation.lat

dLon=targetLocation.lon - currentLocation.lon

return math.sqrt((dLon\*dLon)\*(dLat\*dlat))

distanceToTargetLocation=get\_distance\_meters(target location,vehicle.location.global-relative\_frame)

vehicle.simple\_goto(targetLocation)

for i in range (2):

wi=LocationGlobalRelative(dronekit.longitude, dronekit.latitute)

# automatic read longitudes and latitudes of the drone using GPS module.

currentDistance=get\_distance\_meters(targetLoction,vehicle.location.global\_re lative\_frame)

if(currentDistance < distanceToTargetLocation):

print("reached target waypoint")

time.sleep(2)

break

time.sleep(1)

return None

vehicle=connectMyCopter()

vehicle.parameters("W\_SPEED")#control speed of the drone in m/s

arm()

vehicle.mode = VehicleMode("RTL") #return to the original launch position

while(vehicle.mode!="RTL":

print("Waiting for drone to enter RTL Flight")

time.sleep(1)

print("Vehicle back to initial take off point")

#End of Script

1. Assigning a mission with three waypoints:

import time

import socket

import argparse

import math

def connectMycopter():

parser = argparse.ArgumentParser(description='Commands')

parser.add\_argument('--connect')

args = parser.parse\_args()

connection\_string = args.connect

baud\_rate=57600

if not connection\_string:

import dronekit\_sitl

sitl=dronekit\_sitl.start\_default()

connection\_string=sitl.connection\_string()

vehicle = connect(connection\_string,baud=baud\_rate, wait\_ready=True)

return vehicle

def arm():

while vehicle.is\_armable!=True:

print("Waiting for vehicle to become armable.")

time.sleep(1)

print("Vehicle is armable")

# Copter should arm in GUIDED mode

vehicle.mode=VehicleMode("GUIDED")

while vehicle.mode!="GUIDED":

print("Waiting for drone to enter GUIDED flight mode")

time.sleep(1)

print("Vehicle in Guided Mode")

vehicle.armed=True

# Confirm vehicle armed before attempting to take off

while vehicle.armed=False:

print("Waiting to be armed")

time.sleep(1)

print("Vehicle is armed")

return None

def get\_distance\_meters(targetLocation,currentLocation):

dLat=targetLocation.lat - currentLocation.lat

dLon=targetLocation.lon - currentLocation.lon

return math.sqrt((dLon\*dLon)\*(dLat\*dlat))

def goto(targetLocation):

distanceToTargetLocation=get\_distance\_meters(target location,vehicle.location.global-relative\_frame)

vehicle.simple\_goto(targetLocation)

while (vehicle.mode.name== "GUIDED"):

currentDistance=get\_distance\_meters(targetLoction,vehicle.location.global\_relative\_frame

if(currentDistance < distanceToTargetLocation):

print("reached target waypoint")

time.sleep(2)

break

time.sleep(1)

return None

#give longitudes and latitudes to the drone

w1=LocationGlobalRelative(17.39730929798652,78.4898615696797)

w2=LocationGlobalRelative(17.397162126123774,78.4901887991777)

w3=LocationGlobalRelative(17.397099418076714,78.48982334820145)

vehicle=connectMyCopter()

vehicle.parameters("W\_SPEED")#control speed of the drone in m/s

arm()

goto(w1)

goto(w2)

goto(w3)

vehicle.mode = VehicleMode("RTL") #return to the original launch position

while(vehicle.mode!="RTL":

print("Waiting for drone to enter RTL Flight")

time.sleep(1)

print("Vehicle in Driving Mode")

#End of script