```python
# This file should be set as the controller for the DJI Maverick node.
# Please do not alter this file - it may cause the simulation to fail.

# Import Webots libraries
from controller import Robot

import numpy as np
import pickle

# Import evalution functions
from eval import showPlots

# Import functions from other scripts in controller folder
from lqr_controller import LQRController
from adaptive_controller import AdaptiveController

# Instantiate dron driver supervisor
driver = Robot()

# Get the time step of the current world
timestep = int(driver.getBasicTimeStep())

# Set your percent loss of thrust
lossOfThust = 0.5  # Ex2.1 50% loss of thrust
# lossOfThust = 0.67  # Ex2.2 67% loss of thrust

# Instantiate controller and start sensors
# customController = LQRController(driver, lossOfThust)  # LQR controller
customController = AdaptiveController(driver, lossOfThust)  # Adaptive controller
customController.initializeMotors()
customController.startSensors(timestep)

# Initialize state storage vectors
stateHistory = []
referenceHistory = []

```

```python
# flag for motor failure
motor_failure = False

# calculate gain matrix for baseline LQR controller & adaptive controller
customController.initializeGainMatrix()

# start simulation for LQR controller
while driver.step(timestep) != -1:

    current_time = driver.getTime()
    print("Time:", current_time)

    # motor failure after 14 s
    if current_time > 14:
        motor_failure = True

    # reference trajectory
    if current_time < 10:
        r = np.array([0, 0, 2, 0]).reshape(-1,1)
    elif current_time >= 10 and current_time < 20:
        r = np.array([0, 0, 3, 0]).reshape(-1,1)
    elif current_time >= 20 and current_time < 30:
        r = np.array([0, 0, 2, 0]).reshape(-1,1)
    elif current_time >= 30 and current_time < 40:
        r = np.array([0, 0, 4, 0]).reshape(-1,1)
    elif current_time >= 40 and current_time < 50:
        r = np.array([0, 0, 1, 0]).reshape(-1,1)
    elif current_time >= 50 and current_time < 60:
        r = np.array([0, 0, 4, 0]).reshape(-1,1)
    else:
        # end simulation
        break

    # Call control update method
    states, U = customController.update(r)

    # Check failure
    if (states[2] < 0 ):
        print("="*15 + "Drone Crashed" + "="*15)
        print("="*15 + "Your Controller Failed" +
"="*15)
        break;

    # Convert control input to motoespeed
    rotorspeed = customController.convertUtoMotorSpeed(U)

    # set motor speed
    customController.setMotorsSpeed(rotorspeed, motor_failure)

    # collect state history for evaluation
    stateHistory.append(list(states.flatten()))
    referenceHistory.append(list(r.flatten()))

# save data for evaluation
# reference trajectory
np.save("r_hist_ex2",referenceHistory)

# COMMENT one of two following lines to correctly save te state data
if type(customController).__name__ == 'AdaptiveController':
    np.save("x_ad_hist_ex2",stateHistory) # states using adatpive controller
elif type(customController).__name__ == 'LQRController':
    np.save("x_lqr_hist_ex2",stateHistory) # states using lqr controller

# simulation finished, draw plots
showPlots()
```