Write Servo Commands to the SD Card

Due Jul 6 by 11:59pm **Points** 0 **Submitting** a file upload **File Types** pdf

The objective of this assignment is to program the Raspberry Pi Pico to calculate autonomous flight commands and write them to an SD card. The format of the data on the SD card should match the format of the data collected using the Multi Record feature on the app "Physics Toolbox Sensor Suite", or **for I-Phone or I-Pad users, the phyphox or MATLAB Mobile app,** with three additional columns. The additional columns are for the servo commands delta_e, delta_a, and delta_r. The commands should not exceed the limits of [-1,1]. The headers for the SD card should be as follows:

time	gFx	gFy	gFz	wx	wy	wz	p	Bx	By	Bz	Azimuth	Pitch	Roll	Latitude	Longitude	Speed (m/s)	delta_e	delta_a	delta_r

Autonomous Flight Arduino Library

The chapter ***Autonomous Flight*** in the textbook AutonomousFlight.pdf describes waypoint following strategies and closed loop control strategies. Write an Arduino library and implement a control strategy that can guide the glider to a designated GPS waypoint. Choose a local GPS waypoint such as in the fields above the STC building or at the stadium where you will be able to flight-test your glider in the future. Program the autopilot onto your Raspberry Pi Pico and write the servo commands delta_e, delta_a, and delta_r to the SD card.

MATLAB Code to Check Your Signals

```
close all
clear all
clc

%% import the phone data
ImportDataFromPhone

gyro = [xG,yG,zG]; %(rad/s) 3-axis gyrometer data
Accel = [xA,yA,zA]; %(m/s2) 3-axis accelerometer data
Magnet = [xM,yM,zM]; %(uT) 3-axis magnetometer data
qTrue = [e0,e1,e2,e3]; %(0-1) simulated quaternion orientation

%Get the length of the time vector
N = length(t);
Alt0 = Altitude(1);
rEarth = 6371000; %(m) Earth's radius
GPStarget = [43.815, -111.78600];
Lat0 = GPStarget(1);
Lon0 = GPStarget(2);
[GPSx,GPSy] = getXYpath(GPStarget(1),GPStarget(2), Lat0, Lon0);

%Convert the GPS path to xI and yI displacements
[pathX,pathY] = getXYpath(GPS_lat,GPS_lon, Lat0, Lon0);

figure('WindowState','maximized')
hold on
for ii = 2:N
 %get the time-constant and filter coefficient
 dt = t(ii) - t(ii-1);
 if ~mod(ii,100) % change this value to run faster or slower
   clf('reset')
   subplot(121)
   % hold on
   DrawAirplane(0,0,0,qTrue(ii,:))
   [xI,yI] = getXYpath(GPS_lat(ii),GPS_lon(ii), Lat0, Lon0);
   subplot(122)
   plot(pathX,-pathY, xI, -yI,'o',GPSx,-GPSy,'x')
   axis('equal')
   xlabel('North')
   ylabel('West')
   drawnow
 end
end

function [pathX,pathY] = getXYpath(GPS_lat,GPS_lon, Lat0, Lon0)
```

```
% [pathX,pathY] = getXYpath(GPS_lat,GPS_lon, Lat0, Lon0);
N = length(GPS_lat);
%Allocate memory to store the GPS path of the plane
pathX = zeros(1,N);
pathY = pathX;
for ii = 1:N
 % calculate displacement from GPS
 dLambda = (GPS_lon(ii) - Lon0) * pi/180;
 if abs(dLambda) <= pi
  dLongitude = dLambda;
 elseif abs(dLambda) > pi
  if GPS_lon(ii) > Lon0
   dLongitude = (GPS_lon(ii) - (Lon0+360))*pi/180;
  else
   dLongitude = ((360+GPS_lon(ii)) - Lon0)*pi/180;
  end
 end

 rEarth = 6371000; %(m) Earth's radius
 %Get North xI displacement (m)
 xI = rEarth * (GPS_lat(ii) - Lat0)*pi/180;
 %Get East yI displacement (m)
 yI = sign(dLongitude)*rEarth*...
  acos((cos(dLongitude)-1)*(cosd(GPS_lat(ii)))^2+1);
 %Store the path
 pathX(ii) = xI;
 pathY(ii) = yI;
end
end
```

What to Submit

To get credit for this assignment, demonstrate your working code to the instructor. Also, create a document and do the following:

1. Copy and paste your Raspberry Pi Pico code into the document.
2. Copy and paste your Arduino library control strategy into the document.
3. Save the document as a .pdf file and submit it.

The instructor should be able to run your code to verify that it calculates servo commands in the range [-1,1] and writes them to the SD card.