UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

BACHELOR FINAL THESIS

Design of a swarm of Unmanned Aerial Vehicles for the exploration of Mars

Document:

Appendices

Author:

Núria Escursell i Serra

Director:

Miquel Sureda Anfres

Degree:

Bachelor in Aerospace Vehicles Engineering

Examination session

Autumn, 2022-23

Table of contents

Appendix A

Codes

A.1 Matlab code for the estimation of Mars temperatures in terms of altitude

```
clc; clear all; close all;
%% T-h Mars graphic:

% Variable definition

h=sym('h','real');      % altitude (km)
%T=sym('T','real');      % Temperature (K)


% Equation (based on the Mars Global Reference Atmospheric Model
% (Mars-GRAM)

T(h) = 238.74-34.488*h+35.133*h^2-15.96*h^3+3.7315*h^4-0.47352*h^5+0.030962*h^6-0.000817*h^7;

% Graphic:
fplot(T(h),h,[0,10], 'red',LineWidth=1);
grid on;
title ('T-h on Mars')
xlabel('Temperature (K)')
ylabel('Altitude (km)')


%% T(h) CALCULATIONS:

% North polar caps: 2.5-3.0 km
```

```
26      Tn1=simplify(subs(T(h),h,-2))
27      Tn2=simplify(subs(T(h),h,-5))
28
29  % South polar caps: 1.0-3.5 km
30      Ts1=simplify(subs(T(h),h,1.0))
31      Ts2=simplify(subs(T(h),h,3.5))
32
33  % Equator:
34      Te=simplify(subs(T(h),h,0))
35
36  % Cydonia Mensae:
37      Tc1=simplify(subs(T(h),h,0.65))
38      Tc2=simplify(subs(T(h),h,0.1))
```

A.2 Arduino codes for the IMU

```
/*IMU - Code 1: orientation of the accelerometer - This program reads the ...
    accelerometer data and translates it to orientation.
The sensor used is the Bosch Sensortech BMI-160 IMU, a commercial off-the-shelf ...
    (COTS) miniature sensor used for the Ingenuity helicopter */

#include "CurieIMU.h"

int lastOrientation = -1; /* previous orientation (for comparison) */

void setup() {
  Serial.begin(9600); /* initialize Serial communication. The default speed for ...
    the serial monitor can also be 115200 */
  while (!Serial);    /* wait for the serial port to open */

  /* initialize device */
  Serial.println("Initializing IMU device...");
  CurieIMU.begin();

  /* Set the accelerometer range to 2G */
  CurieIMU.setAccelerometerRange(2);
}

void loop() {
int orientation = -1;   /* the board's orientation */
  String orientationString; /* string for printing description of orientation */
  /*
    The orientations of the board:
    0: flat, processor facing up
    1: flat, processor facing down
    2: landscape, analog pins down
    3: landscape, analog pins up
    4: portrait, USB connector up
    5: portrait, USB connector down
  */
  // read accelerometer:
  int x = CurieIMU.readAccelerometer(X_AXIS);
  int y = CurieIMU.readAccelerometer(Y_AXIS);
  int z = CurieIMU.readAccelerometer(Z_AXIS);

  /* calculate the absolute values to determine the largest one */
  int absX = abs(x);
  int absY = abs(y);
  int absZ = abs(z);

  if ( (absZ > absX) && (absZ > absY)) {
    /* base orientation on Z */
```

```
44      if (z > 0) {
45        orientationString = "up";
46        orientation = 0;
47      } else {
48        orientationString = "down";
49        orientation = 1;
50      }
51    } else if ( (absY > absX) && (absY > absZ)) {
52      /* base orientation on Y */
53      if (y > 0) {
54        orientationString = "digital pins up";
55        orientation = 2;
56      } else {
57        orientationString = "analog pins up";
58        orientation = 3;
59      }
60    } else {
61      /* base orientation on X */
62      if (x < 0) {
63        orientationString = "connector up";
64        orientation = 4;
65      } else {
66        orientationString = "connector down";
67        orientation = 5;
68      }
69    }
70
71    /* if the orientation has changed, print out a description: */
72    if (orientation != lastOrientation) {
73      Serial.println(orientationString);
74      lastOrientation = orientation;
75    }
76  }
```

```
/*IMU - Code 2: reading data of the accelerometer - This program reads data of the accelerometer.
The sensor used is the Bosch Sensortech BMI-160 IMU, a commercial off-the-shelf (COTS) miniature sensor used for the Ingenuity helicopter */
#include "CurieIMU.h"

void setup() {
  Serial.begin(9600); /* initialize Serial communication. The default speed for the serial monitor can also be 115200 */
  while (!Serial);    /* wait for the serial port to open */

  /* initialize device */
  Serial.println("Initializing IMU device...");
  CurieIMU.begin();

  /* Set the accelerometer range to 2G */
  CurieIMU.setAccelerometerRange(2);
}

void loop() {
  float ax, ay, az;   /*scaled accelerometer values */

  /* reading of the accelerometer measurements from device, scaled to the configured range */
  CurieIMU.readAccelerometerScaled(ax, ay, az);

  /* display tab-separated accelerometer x/y/z values */
  Serial.print("a:\t");
  Serial.print(ax);
  Serial.print("\t");
  Serial.print(ay);
  Serial.print("\t");
  Serial.print(az);
  Serial.println();
}
```

A.3 Arduino codes for the inclinometer

```
/* Inclinometer - Code 1: self-test - This program puts the device in self-test mode and reads the test result.
The sensor used is the muRata SCA 100T-D02 inclinometer, a commercial off-the-shelf (COTS) miniature sensor used for the Ingenuity helicopter */

#include <SCA100T.h>
int chipSelect = 10; /* ST_1 pin */
long spiBusSpeed = 500000; /* From the datasheet --> The maximum data transfer speed for RDAX (Read X-channel acceleration) or RDAY (Read Y-channel acceleration) is 5300 samples per sec for one channel at
500kHz clock  */
int sensorType = SCA100TD02;

SCA100T mySensor(chipSelect,spiBusSpeed,SCA100TD02);

int registerValue = 0;

void setup() {
  Serial.begin(115200); /*Default speed for the serial monitor*/
  Serial.println(F("Starting code 1: self-test.")); /* F() allows moving constant strings to the program memory instead of the ram, which frees up dynamic ram. */
}

void loop() {
  int i=10;
  // Start self-test on X-axis
  mySensor.startSelfTest(SCA_ACCEL_X);
  Serial.print(F("Self-test started on X-axis, reading values: "));
  while(i--)
  {
    Serial.println(mySensor.readRaw(SCA_ACCEL_X));
    delay(100);
  }
  mySensor.stopSelfTest();

  // Start self-test on Y-axis
  mySensor.startSelfTest(SCA_ACCEL_Y);
  Serial.print(F("Self-test started on Y-axis, reading values: "));
  i = 10;
  while(i--)
  {
    Serial.println(mySensor.readRaw(SCA_ACCEL_Y));
    delay(100);
  }
  mySensor.stopSelfTest();
```

```
41
42   delay(1000);
43 }
```

```
1 /* Inclinometer - Code 2: reading X and Y angles - This program reads the angles...
      X and Y (roll and pitch) from the sensor and prints them to a serial port.
2 The sensor used is the muRata SCA 100T-D02 inclinometer, a commercial off-the-...
      shelf (COTS) miniature sensor used for the Ingenuity helicopter */
3
4 #include <SCA100T.h>
5 int chipSelect = 10;
6 long spiBusSpeed = 500000; /* From the datasheet --> The maximum data transfer ...
      speed for RDAX (Read X-channel acceleration) or RDAY (Read Y-channel ...
      acceleration) is 5300 samples per sec for one channel at
7 500kHz clock  */
8 int sensorType = SCA100TD02;
9
10 SCA100T mySensor(chipSelect,spiBusSpeed,SCA100TD02);
11
12 void setup() {
13   Serial.begin(115200);
14   Serial.println(F("Starting example 1 - Read X and Y -angles."));
15 }
16
17 void loop() {
18   float angle_X = mySensor.getAngle(SCA_ACCEL_X);
19   float angle_Y = mySensor.getAngle(SCA_ACCEL_Y);
20   Serial.print(F("Angle X: "));
21   Serial.print(angle_X);
22   Serial.print(F("Angle Y: "));
23   Serial.print(angle_Y);
24   delay(1000);
25 }
```

A.4 Arduino codes for the LiDAR

```
/* LiDAR - Code 1: initialization, configuration and reading the distance from a LiDAR connected over the I2C interface (a multi-controller).
The sensor used is the Garmin LiDAR-Lite v3, a commercial off-the-shelf (COTS) miniature sensor used for the Ingenuity helicopter*/

/* Connections:
  LIDAR-Lite 5 Vdc (red) to Arduino 5v
  LIDAR-Lite I2C SCL (green) to Arduino SCL
  LIDAR-Lite I2C SDA (blue) to Arduino SDA
  LIDAR-Lite Ground (black) to Arduino GND

  (It is recommended a capacitor to mitigate inrush current when device is enabled)
  680uF capacitor (+) to Arduino 5v
  680uF capacitor (-) to Arduino GND
*/

#include <Wire.h>
#include <LIDARLite.h>

LIDARLite myLidarLite;

void setup()
{
  Serial.begin(115200); /* Default speed for the serial monitor. Initialize serial connection to display distance readings. */

  /*
    begin(int configuration, bool fasti2c, char lidarliteAddress)
    Starts the sensor and I2C.
    Parameters
    ----------------------------------------------------------------------
    configuration: Default 0. Selects one of several preset configurations.
    fasti2c: Default 100 kHz. I2C base frequency.
      If true I2C frequency is set to 400kHz.
    lidarliteAddress: Default 0x62. Fill in new address here if changed. See
      operating manual for instructions.
  */
  myLidarLite.begin(0, true); /* Set configuration to default and I2C to 400 kHz. */

  /*
    configure(int configuration, char lidarliteAddress)
    Selects one of several preset configurations.
    Parameters
    ----------------------------------------------------------------------
```

```
42      configuration: Default 0.
43        0: Default mode, balanced performance.
44        1: Short range, high speed. Uses 0x1d maximum acquisition count.
45        2: Default range, higher speed short range. Turns on quick termination
46            detection for faster measurements at short range (with decreased
47            accuracy)
48        3: Maximum range. Uses 0xff maximum acquisition count.
49        4: High sensitivity detection. Overrides default valid measurement ...
      detection
50            algorithm, and uses a threshold value for high sensitivity and noise.
51        5: Low sensitivity detection. Overrides default valid measurement ...
      detection
52            algorithm, and uses a threshold value for low sensitivity and noise.
53      lidarliteAddress: Default 0x62. Fill in new address here if changed. See
54        operating manual for instructions.
55    */
56    myLidarLite.configure(0); /* This number can be changed to try alternate ...
        configurations*/
57  }
58
59  void loop()
60  {
61    /*
62      distance(bool biasCorrection, char lidarliteAddress)
63      Take a distance measurement and read the result.
64      Parameters
65      --------------------------------------------------------------------------
66      biasCorrection: Default true. Take aquisition with receiver bias
67        correction. If set to false measurements will be faster. Receiver bias
68        correction must be performed periodically. (e.g. 1 out of every 100
69        readings).
70      lidarliteAddress: Default 0x62. Fill in new address here if changed. See
71        operating manual for instructions.
72    */
73
74    /* Take a measurement with receiver bias correction and print to serial ...
        terminal */
75    Serial.println(myLidarLite.distance());
76
77    /*Take 99 measurements without receiver bias correction and print to serial ...
        terminal */
78    for(int i = 0; i < 99; i++)
79    {
80      Serial.println(myLidarLite.distance(false));
81    }
82  }
```

A.5 Matlab code for the Triangular Formation Algorithm (TFA)

```
%% Code 1 TFA: Initial triangle
clear; close all; clc;

%% Data entry and parameters:
% x,y coordinates of Pi
xi=0;
yi=30;
Pi=[xi, yi]; % position of the leader UAV

% x,y coordinates of Pj
xj=10;
yj=0;
Pj=[xj, yj]; % position of the follower UAV

% Heading angle
psi_i=0;

% Coordinates of Pk (symmetrical to Pj)
Pk= [xi-sqrt((xi-xj)^2+(yi-yj)^2)*sin(psi_i), yi+sqrt((xi-xj)^2+(yi-yj)^2)*cos(psi_i)]; % ...
    position of the follower UAV (symmetrical to Pj)
% the square for calculating the distance: |Pi - Pj|
xk=Pk(1);
yk=Pk(2);

% Distance between vehicles:
d_vinculum=50;

%% Implementation of the code
% Pi(t)-Pj(t)|    |Pi(t)-Pk(t)| and |Pi(t)-Pk(t)|    2d_vinculum
if (d_vinculum≤(sqrt((xj-xk)^2+(yj-yk)^2))/2) && ...
     (d_vinculum≤(sqrt((xi-xk)^2+(yi-yk)^2))/2) && (d_vinculum≤(sqrt((xi-xj)^2+(yi-yj)^2))/2)
    Pi_circumflex=(Pi+Pj+Pk)/3;

else
    % Middle point
    Pm=(Pj+Pk)/2;
    xm=Pm(1);
    ym=Pm(2);

    % Angles
    theta_jk= mod(atan2(yi-yk,xj-xk),2*pi);
    theta_im= mod(atan2(ym-yi,xm-xi),2*pi);
    Δ_theta=theta_jk-theta_im;

    % Height h
    h=sqrt(d_vinculum^2-(sqrt((xm-xj)^2+(ym-yj)^2))^2);
```

```
45
46      if (Δ_theta≥0) && (Δ_theta<pi) % To obtain the desired position
47          Pi_circumflex=[xm-h*sin(theta_jk), ym+h*cos(theta_jk)];
48          xi_circumflex=Pi_circumflex(1);
49          yi_circumflex=Pi_circumflex(2);
50      else
51          Pi_circumflex=[xm+h*sin(theta_jk), ym-h*cos(theta_jk)];
52          xi_circumflex=Pi_circumflex(1);
53          yi_circumflex=Pi_circumflex(2);
54      end
55  end
56
57  % Implementation of the TFA for the initial case
58  figure(1)
59  plot([Pj(1) Pk(1)],[Pj(2) Pk(2)],'r');
60  hold on
61  plot([Pi_circumflex(1) Pk(1)],[Pi_circumflex(2) Pk(2)],'g');
62  hold on
63  plot([Pj(1) Pi_circumflex(1)],[Pj(2) Pi_circumflex(2)],'c');
64  hold on
65  grid
66  plot([Pi_circumflex(1) Pm(1)],[Pi_circumflex(2) Pm(2)],'y');
67  xlabel('x (m)')
68  ylabel('y (m)')
69  title('Implementation of the TFA for the initial case')
70
71
72  % Demonstration that Pj and Pk are symmetric
73  distance_Pj_Pm=sqrt((xm-xj)^2+(ym-yj)^2);
74  distance_Pk_Pm=sqrt((xm-xk)^2+(ym-yk)^2);
75
76  if (distance_Pj_Pm==distance_Pk_Pm)
77      figure(2)
78      plot([Pj(1) Pm(1)],[Pj(2) Pm(2)],'r');
79      hold on
80      plot([Pm(1) Pk(1)],[Pm(2) Pk(2)],'b');
81      xlabel('x (m)')
82      ylabel('y (m)')
83      title('Demonstration that Pj and Pk are symmetric')
84  end
```

```
%% Code 2 TFA: Triangle according to time
clear; close all; clc;

%% Data entry and parameters:
N=2;
M=10; % number of rows

psi_i=0; % Heading angle

% Coordinates of Pi and Pj
for i=1:M
    Pi(i,1)=20*i;
    Pj(i,1)=20*i+10;
    for j=1:M
       Pi(j,2)=20*j+30;
       Pj(j,2)=20*j;
    end
end

% Coordinates of Pk (i,j). The square for calculating the distance: |Pi - Pj|
for i=1:M
    Pk(i,1)= Pi(i,1)-sqrt((Pi(i,1)-Pi(j,2))^2+(Pi(j,2)-Pj(j,2))^2)*sin(psi_i);
    for j=1:M
       Pk(j,2)=Pi(j,2)+sqrt((Pi(i,1)-Pi(j,2))^2+(Pi(j,2)-Pj(j,2))^2)*cos(psi_i);
    end
end

% Distance between vehicles:
d_vinculum=50;

%% Implementation of the code
% Pi(t)-Pj(t)|    |Pi(t)-Pk(t)| and |Pi(t)-Pk(t)|    2d_vinculum
xi=Pi(:,1);
yi=Pi(:,2);
xj=Pj(:,1);
yj=Pj(:,2);
xk=Pk(:,1);
yk=Pk(:,2);

for i=1:M
    for j=1:M
       if (d_vinculum≤(sqrt((Pj(i,1)-Pk(i,1))^2+(Pj(j,2)-Pk(j,2))^2))/2) && ...
            (d_vinculum≤(sqrt((Pi(i,1)-Pk(i,1))^2+(Pi(j,2)-Pk(j,2))^2))/2) && ...
            (d_vinculum≤(sqrt((Pi(i,1)-Pj(i,1))^2+(Pi(j,2)-Pi(j,2))^2))/2)
           Pi_circumflex=(Pi+Pj+Pk)/3;
       else
           % Middle point
           xm=(Pi(:,1)+Pj(:,1))/2;
```

```
            ym=(Pi(:,2)+Pj(:,2))/2;
            Pm=[xm, ym];

            % Angles
            theta_jk= mod(atan2(yi-yk,xj-xk),2*pi);
            theta_im= mod(atan2(ym-yi,xm-xi),2*pi);
            Δ_theta=theta_jk-theta_im;

            % Height h
            h=sqrt(d_vinculum^2-(sqrt((Pm(i,1)-Pj(i,1))^2+(Pm(j,2)-Pi(j,2))^2))^2);

            if (Δ_theta(j)≥0) && (Δ_theta(j)<pi) % To obtain the desired position
                Pi_circumflex=[xm-h*sin(theta_jk), ym+h*cos(theta_jk)];
                xi_circumflex=Pi_circumflex(1);
                yi_circumflex=Pi_circumflex(2);
            else
                xi_circumflex= xm+h*sin(theta_jk);
                yi_circumflex=ym-h*cos(theta_jk);
                Pi_circumflex=[xi_circumflex, yi_circumflex];
            end
        end
    end
    % TFA depending on time
    figure(1)
        plot([Pj(i,1) Pk(i,1)],[Pj(j,2) Pk(j,2)],'r');
        hold on
        plot([Pi_circumflex(i,1) Pk(i,1)],[Pi_circumflex(j,2) Pk(j,2)],'g');
        hold on
        plot([Pj(i,1) Pi_circumflex(i,1)],[Pj(j,2) Pi_circumflex(j,2)],'c');
        hold on
        grid
        plot([Pi_circumflex(i,1) Pm(i,1)],[Pi_circumflex(j,2) Pm(j,2)],'y');
        xlabel('x (m)')
        ylabel('y (m)')
        title('Implementation of the TFA depending on time')
end

% Plot to prove that d_vinvulum does not depend on time
figure(2)
x = linspace(0,10);
y = 0*x+d_vinculum;
plot(x,y, 'b')
grid
xlabel('t (s)')
ylabel('d (m)')
title('d vs time')
```