November 11, 2020

Entry Project 2020

Matlab Sensor Model

REV. 0

SKYWARD EXPERIMENTAL ROCKETRY

Software & Control Systems Team

Skyward Experimental Rocketry
Politecnico di Milano

Author: Jan Hämmelmann
Editor:

Abstract
Scrivi qui il tuo abstract

Website:
http://www.skywarder.eu

E-mail:
jan.hammelmann@skywarder.eu

Restricted use policy

Contents

1 Introduction **2**

2 Implementation **3**

2.1 Classes . . . 3

2.1.1 Sensor . . . 3

2.1.1.1 saturation . . . 4

2.1.1.2 quantization . . . 4

2.1.1.3 whiteNoise . . . 4

2.1.1.4 add2DOffset . . . 4

2.1.1.5 addOffset . . . 4

2.1.1.6 addTempOffset . . . 4

2.1.2 Sensor3D . . . 4

2.1.2.1 addOffset3D . . . 5

2.1.2.2 randomWalk . . . 5

2.1.2.3 tranformAxis . . . 5

2.1.3 GPS . . . 5

2.2 Sensor implementation . . . 5

2.2.1 LSM9DS1 . . . 5

2.2.1.1 Accelerometer . . . 5

2.2.1.2 Magnetometer . . . 6

2.2.1.3 Gyroscope . . . 6

2.2.2 MS580301BA01 . . . 6

2.2.2.1 Barometer . . . 6

2.2.3 IIS2MDC . . . 7

2.2.3.1 Magnetometer . . . 7

2.2.4 NEO-M9N . . . 8

2.2.4.1 GPS . . . 8

List of Figures

2.1 Deterministic error of the barometer depending on pressure and temperature from [2] 7

List of Tables

Notes

Change Log

Date	Revision	Editor	Changes
Data	0	Jan Hämmelmann	Prima versione.

Chapter 1

Introduction

In this entry project of 2020 classes should be programmed, which that simulates the acquisition process of real sensors with all the non idealities it leads to. The programming style should be object oriented . After creating the object of the classes should be initialized, which represent the real used sensors of the rocket project.

In the following chapters the implemented classes, the used sensor and the defined properties are described.

Chapter 2

Implementation

The project is separated in small parts. With the function `createMagneticData(magneticDensity, q0, q1, q2, q3)` magnetic data can be created from the quaternion states of the simulator. Afterwards the sensor data can be manipulated the `sens` method of the implemented sensors. For the implementation, depending on the type of sensor the classes Sensor, Sensor3D or GPS can be used. In this project some sensors where already initialized in the `initSensor.m` script.

2.1 Classes

2.1.1 Sensor

Sensor is inheriting from handle to be able to change the properties inside methods. The properties of the class are:

```
minMeasurementRange; % Max limit of sensor
maxMeasurementRange; % Min limit of sensor

resolution; % resolution of the sensor

noiseVariance; % Varianze for the gaussian white noise

offset; % Offset in all directions

tempOffset; % Coefficent for temperature depending offset

dt; % Sampling time

error2dOffset; % first column: inputArg, second column: relativeArg, third column:
```

They are needed to use the methods of the class. With the method `sens(inputArg,temp)` all following methods are called where the necessary properties are defined. The sensor class is created for 1D sensors. Now the methods called in the sens method are shortly described.

2.1.1.1 saturation

This method limits inputArg to the properties minMeasurementRange at the lower end and maxMeasurementRange at the upper end.

2.1.1.2 quantization

This method gets the input inputArg and quantized this with the property resolution.

2.1.1.3 whiteNoise

This method adds to the input inputArg a gaussian white noise with the variance of the property noiseVariance.

2.1.1.4 add2DOffset

The method takes the inputs inputArg and temp and uses the property error2dOffset of the object, which has to be defined before the use of the method. Add2DOffset interpolates between the data points of the 2D dataset error2dOffset [x1,x2,y] and finds a value yq that fits to the inputs inputArg and temp which refer to x1 and x2. Value yq is then added to inputArg and results in outputArg.

2.1.1.5 addOffset

Adds the properties offset to the input signal.

2.1.1.6 addTempOffset

Adds the property tempOffset multiplied by the temperature input temp to the input signal.

2.1.2 Sensor3D

Sensor is inheriting form Sensor.
The properties of the class are:

```
offsetX; % offset in x direction
offsetY; % offset in y direction
offsetZ; % offset in z direction

walkDiffusionCoef; % diffusion coefficient for the random walk

transMatrix % transformation matrix
```

With the method `sens(inputArgX,inputArgY,inputArgZ,temp)` all methods from class sensor are called for all three sensor axis plus the methods defined in class Sensor3D. The methods are addOffset3D, randomWalk and tranformAxis.

2.1.2.1 addOffset3D

Like the method addOffset in class Sensor this method is adding a offset independent to all three axis with the value of the properties offsetX, offsetY and offsetZ.

2.1.2.2 randomWalk

This method is adding a Brown motion to the sensor data with the diffusion coefficient of the property walkDiffusionCoef.

2.1.2.3 tranformAxis

This method is transforming the input data vector with the transformation matrix in property transMatrix.

2.1.3 GPS

GPS is inheriting from Sensor3D. The `sens(lat,lon,alt,temp)` is before calling the method `sens(inputArgX,inputArgY,` of the superclass Sesnor3D transforming the latitude, longitude and altitude in x, y and z coordinates. After using the method of Sensor3D the data is transformed back into latitude, longitude and altitude.

2.2 Sensor implementation

The Sensor are initialized in `initSensor.m`. The values can be fined in the data sheets. In the following subsection the initializations is described.

2.2.1 LSM9DS1

The LSM9DS1 is a IMU including an accelerometer, a magnetometer and a gyroscope. The sensor data can be found in [1]. The accelerometer is sensing the x, y and z accelerations in mg. The gyroscope is sensing the angular velocities in mdps. The magnetometer is sensing the x, y and z magnetic field in mgauss.

2.2.1.1 Accelerometer

Here the accelerometer of the LSM9DS1 is described. The parameter are defined with:

```
ACCEL_LSM9DS1=Sensor3D(); % acceleration in mg
ACCEL_LSM9DS1.maxMeasurementRange=16000; % 2000, 4000, 8000, 16000 in mg
ACCEL_LSM9DS1.minMeasurementRange=-16000; % -2000, -4000, -8000, -16000 in mg
ACCEL_LSM9DS1.resolution=0.732; % 0.061, 0.122, 0.244, 0.732 in mg
ACCEL_LSM9DS1.noiseVariance=4; % guess in mg
ACCEL_LSM9DS1.offsetX=0; % +-90 in mg
ACCEL_LSM9DS1.offsetY=0; % +-90 in mg
ACCEL_LSM9DS1.offsetZ=0; % +-90 in mg
ACCEL_LSM9DS1.walkDiffusionCoef=1; % guess
ACCEL_LSM9DS1.dt=0.01; % sampling time
```

2.2.1.2 Magnetometer

Here the magnetometer of the LSM9DS1 is described. The parameter are defined with:

```
MAGN_LSM9DS1=Sensor3D(); % magnetic field in mgauss
MAGN_LSM9DS1.maxMeasurementRange=16000; % 4000, 8000, 12000, 16000 in mgauss
MAGN_LSM9DS1.minMeasurementRange=-16000; % -4000, -8000, -12000, -16000 in mgauss
MAGN_LSM9DS1.resolution=0.58; % 0.14, 0.29, 0.43, 0.58 in mgauss
MAGN_LSM9DS1.noiseVariance=2; % guess in mgauss
MAGN_LSM9DS1.offsetX=0; % +-1000 in mgauss
MAGN_LSM9DS1.offsetY=0; % +-1000 in mgauss
MAGN_LSM9DS1.offsetZ=0; % +-1000 in mgauss
MAGN_LSM9DS1.walkDiffusionCoef=1; % guess
MAGN_LSM9DS1.dt=0.01; % sampling time
MAGN_LSM9DS1.transMatrix=diag([1 1 1]); % axis transformation
```

2.2.1.3 Gyroscope

Here the gyroscope of the LSM9DS1 is described. The parameter are defined with:

```
GYRO_LSM9DS1=Sensor3D(); % angular rate in mdps
GYRO_LSM9DS1.maxMeasurementRange=2000e3; % 245e3, 500e3, 2000e3 in mdps
GYRO_LSM9DS1.minMeasurementRange=-2000e3; % -245e3, -500e3, -2000e3 in mdps
GYRO_LSM9DS1.resolution=70; % 8.75, 17.5, 70 in mdps
GYRO_LSM9DS1.noiseVariance=100; % guess in mdps
GYRO_LSM9DS1.offsetX=0; % +-30e3 in mdps
GYRO_LSM9DS1.offsetY=0; % +-30e3 in mdps
GYRO_LSM9DS1.offsetZ=0; % +-30e3 in mdps
GYRO_LSM9DS1.walkDiffusionCoef=1; % guess
GYRO_LSM9DS1.dt=0.01; % sampling time
GYRO_LSM9DS1.transMatrix=diag([1 1 1]); % axis transformation
```

2.2.2 MS580301BA01

The MS580301BA01 is a barometer. The sensor data can be found in [2]. The measuring unit is mbar.

For the sensor saturation **??**, quantization **??** and white noise **??** is implemented. Furthermore there is a deterministic error for the sensor dependency on the pressure and temperature. This is implemented with the add2DOffset method 2.1.1.4. The data sheet of the sensor provides us with the information in figure 2.1. The information is extracted in `.\data\char\MS580301BA01` in the CSV files `ep\_p\_0.csv`, `ep\_p\_25.csv`, `ep\_p\_85.csv`, `ep\_p\_85.csv` and `ep\_p\_-40.csv`. They are combined in a matrix with columns [pressure p, temperature T, error pressure ep]. This matrix must be used as property `error2dOffset` of the MS580301BA01 object.

2.2.2.1 Barometer

Here the barometer MS580301BA01 is described. The parameter are defined with:

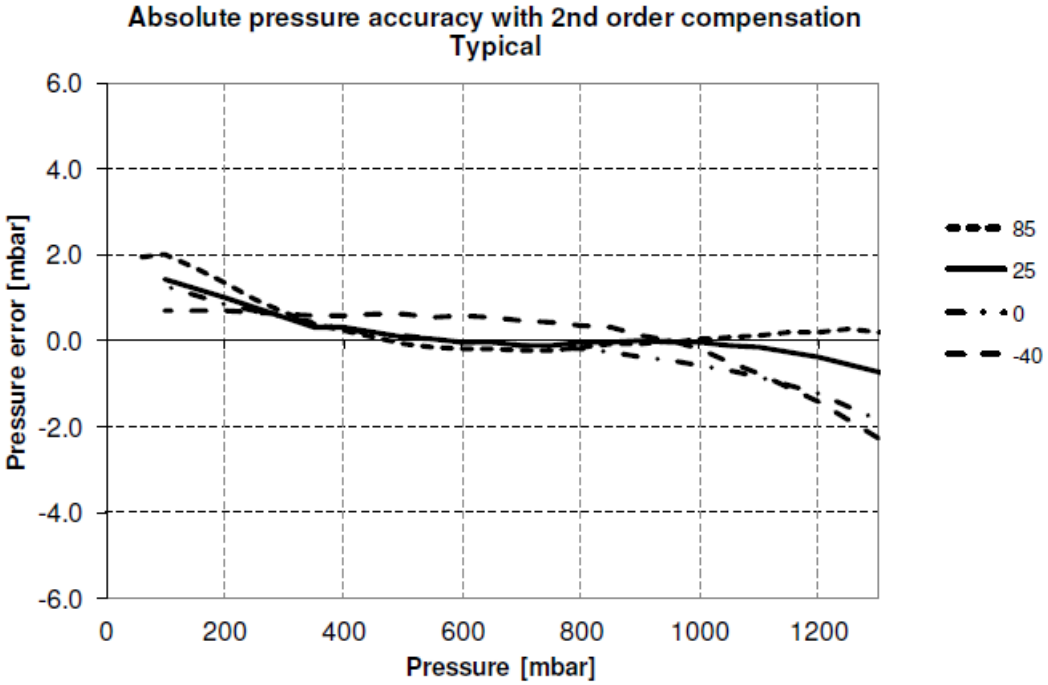

Figure 2.1: Deterministic error of the barometer depending on pressure and temperature from [2]

```
MS580301BA01=Sensor(); % presure in mbar, temp should be in C°
MS580301BA01.maxMeasurementRange=1100; % 1100, 1300 in mbar
MS580301BA01.minMeasurementRange=300; % 300, 10 in mbar
MS580301BA01.resolution=0.065; % 0.012, 0.018, 0.027, 0.042, 0.065 in mbar
MS580301BA01.noiseVariance=4; % guess in mbar
MS580301BA01.error2dOffset=ep_data; % [p in mbar, T in celsius, ep in mbar]
```

2.2.3 IIS2MDC

The IIS2MDC is a magnetometer. The sensor data can be found in [3].

2.2.3.1 Magnetometer

Here the magnetometer IIS2MDC is described. The parameter are defined with:

```
MAGN_IIS2MDC=Sensor3D(); % magnetic field in mgauss, temp should be in C°−25C°
MAGN_LSM9DS1.maxMeasurementRange=49152; % in mgauss
MAGN_LSM9DS1.minMeasurementRange=−49152; % in mgauss
MAGN_LSM9DS1.resolution=1.5; % in mgauss
MAGN_LSM9DS1.tempOffset=0; % +−0.3 in mgauss
MAGN_LSM9DS1.noiseVariance=9; % guess in mgauss
MAGN_LSM9DS1.offsetX=0; % +−60 in mgauss
MAGN_LSM9DS1.offsetY=0; % +−60 in mgauss
```

```
MAGN_LSM9DS1.offsetZ=0; % +-60 in mgauss
MAGN_LSM9DS1.walkDiffusionCoef=1; % guess
MAGN_LSM9DS1.dt=0.01; % sampling time
MAGN_LSM9DS1.transMatrix=diag([1 1 1]); % axis transformation
```

2.2.4 NEO-M9N

The NEO-M9N is a GPS module. The sensor data can be found in [4].

2.2.4.1 GPS

Here the magnetometer NEO-M9N is described. The parameter are defined with:

```
GPS_NEOM9N=GPS(); % lon, in degree lat in deree, alt in m
GPS_NEOM9N.noiseVariance=4; % in m
GPS_NEOM9N.transMatrix=diag([1 1 1]); % axis transformation
```

Bibliography

[1] LSM9DS1 data sheet

[2] MS580301BA01 data sheet

[3] IIS2MDC data sheet

[4] NEO-M9N data sheet