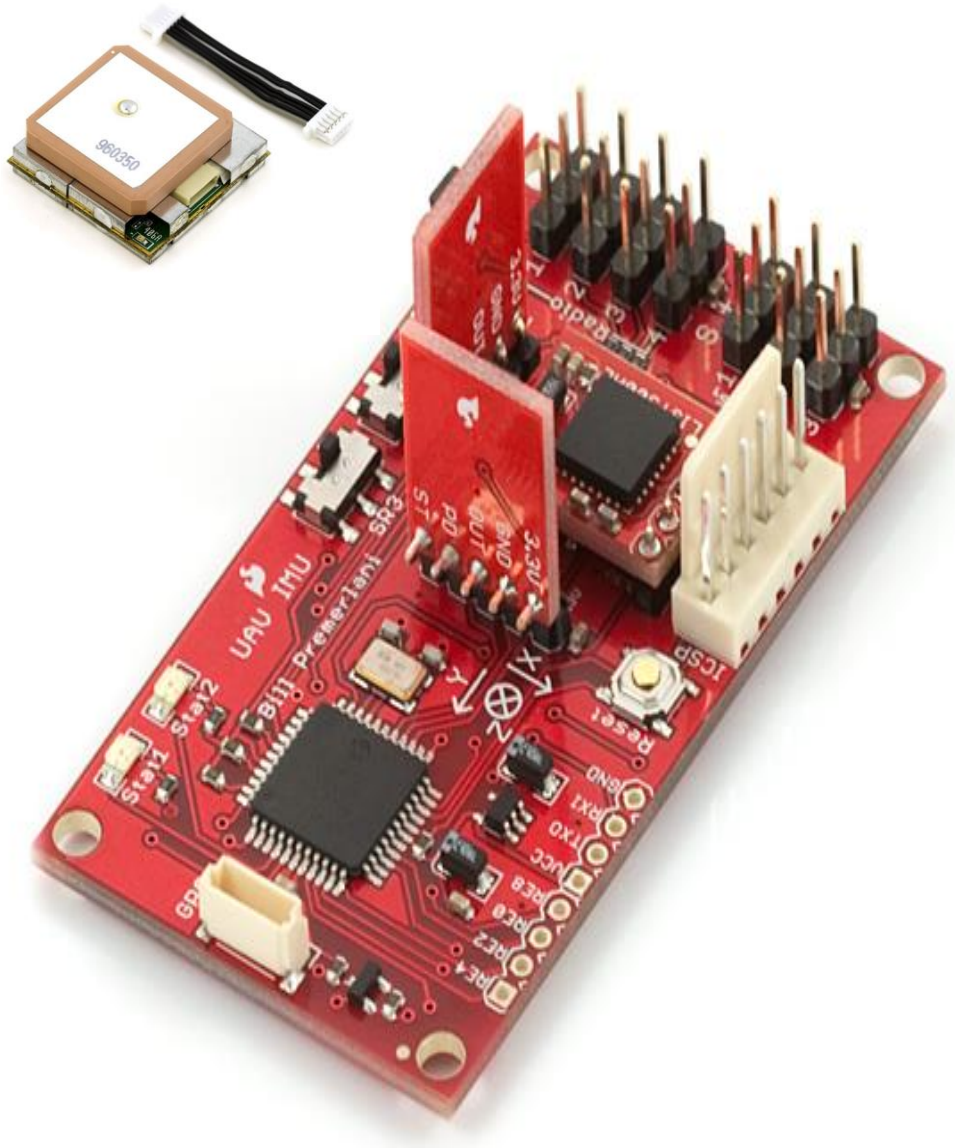


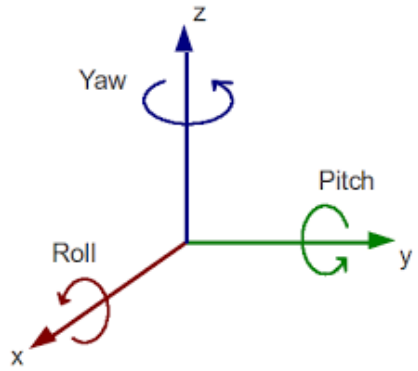
Carlos Caravaca Gallego
Almudena Toronjo Ruiz

GPS AND INERTIAL DATA ACQUISITION, WRANGLING AND FUSION
BRIEF STUDY OF AIRBORNE SOFTWARE SYSTEMS CERTIFICATION

University of Seville

Navigation Aids Systems Course

Contents



OBJETIVES

1. Procurement, handling and integration of both inertial and GPS data from electronic components

2. Study of the considerations in the algorithm certification.

WHY?

It is the starting point of the development of on board algorithms

TOOLS

Phyton
Matlab
Ada programming Language
UAV v2 Development Platform
EM-406A
NMEA protocol

Methodology
GPS Data Acquisition

- EM406 GPS-UAV Development Platform

EM406 SiRF III GPS receiver

HARDWARE

CPU = Microchip dsPIC30F4011

ICSP header (VPP, VCC, GND, PGD, PGC and N/C)

Basic breakout board for the USB to serial IC

3 axis GPS-UAV with plenty of processing power and features

Spare pins

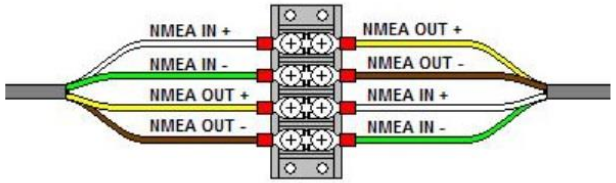
3 STMicroelectronics LISY300AL **gyros** (one for each axis)
1 three axis MMA7260 **accelerometer**

Methodology
GPS Data Acquisition

- NMEA 0183

WHAT IS IT?

Electrical and data specification for communication between marine electronic devices such as gyrocompass, autopilot and GPS receivers



WHAT DOES IT USE?

ASCII= serial communications protocol

defines how **data*** is transmitted in a sentence from one talker to multiple listeners at a given time

ADVANTAGES

→There are standard sentences for each device category

→ There is also the ability to define proprietary sentences for use by the individual company

*DATA: complete PVT solution (position, velocity, time)

Methodology
GPS Data Acquisition

- Python

WHY PYTHON AND NOT OTHERS LIKE CUTECOM?

1. Test the pySerial library

2. Because it provides an easy method to establish a serial interface connection.

3. Because the output is ready to be parsed also in python

1 Hz frequency
and 4800 as baudrate.

```
import serial
import time

class serialInterface():

  tty = None
  baud = None
  serial = None

  def __init__(self, tty, baud = 4800):
    self.tty = tty
    self.baud = baud
    self.serial = serial.Serial(self.tty, self.baud)

  def read(self):
    in_bytes = self.serial.inWaiting()
    time.sleep(1)
    if in_bytes > 0:
      return self.serial.read(in_bytes)
    else:
      return ""

interface = serialInterface('/dev/ttyUSB1',4800)

while True:
  print(interface.read())
```

Inertial Data Acquisition

Coordination with other group of students: *Sistema de Navegación Inercial INS*

DESCRIPTION:
calibration and data acquisition project from an Inertial Measurement Unit, i.e., the 9DOF Razor IMU

9DOF RAZOR IMU
three sensors (triple-axis gyro, triple-axis accelerometer and triple-axis magnetometer) → to give nine degrees of inertial measurement

Timestamp (ns)	accX (m/s^2)	accY (m/s^2)	accZ (m/s^2)
6.81E+13	-0.0104	0.1904	9.9066
6.81E+13	-0.2194	0.5006	9.8176
6.81E+13	0.0448	0.1305	9.9859

Example of Inertial Data

Data Wrangling

HOW?

Python

WHY?

Possibilities within seem endless.

Allows to create CSV output to easily read data in a spreadsheet

WHAT IS WRANGLING FOR?

FROM RAW DATA FROM GPS→ TO DESIRED FORMAT THAT CAN BE INTERPRETED BY MATLAB AND ADA PROGRAMMING LANGUAGE

WHAT HAS BEEN USED?

The **module *csv*** from the Python standard library→ output the data to CSV format.

✚

A **function** → transform latitude and longitude strings into relative x and y coordinates*

***UTM library** was imported and used to apply the Universal Transverse Mercator

```
import utm
import csv
import numpy as np

#Define function to convert latitude and longitude to local XYZ
def latlon_to_xyz(latitude,longitude):
    coordinates
    #Latitude in format DDMM.MMMM
    #Longitude in format DDDMM.MMMM
    lat_sign = 1 if latitude[1] == "N" else -1
    lon_sign = 1 if longitude[1] == "E" else -1
    lat_deg_min     = [int(latitude[0][:2]),float(latitude[0][2:])]
    lon_deg_min     = [int(longitude[0][:3]),float(longitude[0][3:])]
    lat  = ((float(lat_deg_min[0])
            + float(lat_deg_min[1])/60.0)*lat_sign)
    lon  = ((float(lon_deg_min[0])
            + float(lon_deg_min[1])/60.0)*lon_sign)
    return (utm.from_latlon(lat, lon)[0], utm.from_latlon(lat, lon)[1])
```

Sensor Data Fusion. Ada Programming Language

- Features

WHERE IS IT USED?

applications where safety and security are of the utmost importance

It is designed for developing very large software systems.

ADVANTAGES

1. Its feature set and programming paradigms allow software developers to develop applications more effectively and efficiently
2. The syntax of Ada minimizes choices of ways to perform basic operations

Readable, reliable, and maintainable software.

It is an ideal tool to approach the design with compliance to the primary document by which the certification authorities approve all commercial software-based aerospace system → **DO-178C**

Sensor Data Fusion. Ada Programming Language. Features.

- DO-178C

WHAT IS IT?

Software Considerations in Airborne Systems and Equipment Certification published by RTCA

WHAT DOES IT DO?

The document attributes to every system a software level = Design Assurance Level (**DAL**)

It is determined from the safety assessment process and hazard analysis by examining the effects of a failure condition in the system

Depending on the DAL, the document establishes a number of objectives to be satisfied by the systems.

Certifying that a system complies with this standard is a challenging task but appropriate usage of qualified tools and specialized runtime libraries can significantly simplify the effort. For this purpose, the utilisation of the technologies offered by AdaCore is convenient → it helps to cover not only the core DO-178C document but also the technology supplements

DO-178 requires a high **traceabiliy** = connection (called a trace) between the certification artifacts.
EXAMPLE: a Low Level Requirement (LLR) traces up to a High Level Requirement (HLR) → A traceability analysis is then used to ensure that each requirement is fulfilled by the source code, that each requirement is tested, that each line of source code has a purpose (is connected to a requirement), and so forth.
Traceability ensures the system is complete.

Theoretical Analysis and Problem Formulation.

- KALMAN FILTER

WHAT IS IT?

Algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies

↓

Produces estimates of unknown variables that tend to be more accurate than those based on a single measurement

WHAT DOES IT GET?

The mathematical equations provide an **efficient recursive solution** by the least-squares method ➔ A LINEAR, UNBIASED, AND OPTIMAL ESTIMATOR OF THE STATE OF THE SYSTEM at the moment of time t based on the information available at time t-1 and update with the additional information available at time t

It is **recursive** ➔ can run in real-time, using only the present input measurements and the previously calculated state and its uncertainty matrix

predict the system state in the past, present, and future

HOW?

Prediction and correction

- Prediction: Kalman filter produces estimates of the current state variables, along with their uncertainties.
- Correction: the estimates are updated when the outcome of the next one is observed using a **weighted average** ⇔ more weight being given toestimates with higher certainty

*All equations are available in memory

Theoretical Analysis and Problem Formulation

- Ada Programming Language

Basic structure of the general Kalman filter implementation in Ada

Ada.Numerics.Real_Arrays package
Generics
Package-oriented implementation
Strongly-typed syntax

Design focused towards DO-178C compliance

```
with Ada.Numerics.Real_Arrays;
use Ada.Numerics.Real_Arrays;
generic
  type Real is digits <>;  -- used to define any floating-point type
  N : Positive; -- N represents the dimension of the state
  M : positive; -- M represents the number of measurements
package generalKalmanFilter is
  type State is Real_vector (1..N);
  type stateMatrix is Real_Matrix (1..N, 1..M);
  type Measurements is Real_Matrix (1..N, 1..M);
  procedure defineStateMatrix (F : in stateMatrix);
  procedure defineNoise (Q : in stateMatrix);
  procedure defineMeasurements (h : in Measurements);
  procedure defineMeasurementsNoise (r : in Real);
  procedure predictionStep (X : in out State;
                            P : in out stateMatrix);
  procedure updateStep (z : in Real;
                        X : in out State;
                        P : in out stateMatrix;
                        R : out Real);
end generalKalmanFilter;
```

Data Fusion

KEY POINTS

GPS and INS navigation systems are **complimentary**

The frequencies of each device are different → the faster one, the IMU, is used to interpolate the positions and speeds between fifty GPS measurement inputs.

Sensors provide the same type of information about movement, it is necessary to merge both signals in order to enter the filter as a **single source**

Data provided by both sensors have been expressed in the same coordinate system and the same units (rad, m and s)

...ABOUT THE IMPLEMENTATION OF KALMAN FILTER

Its has been used a function available in Matlab to apply a Kalman filtering on the data fusion

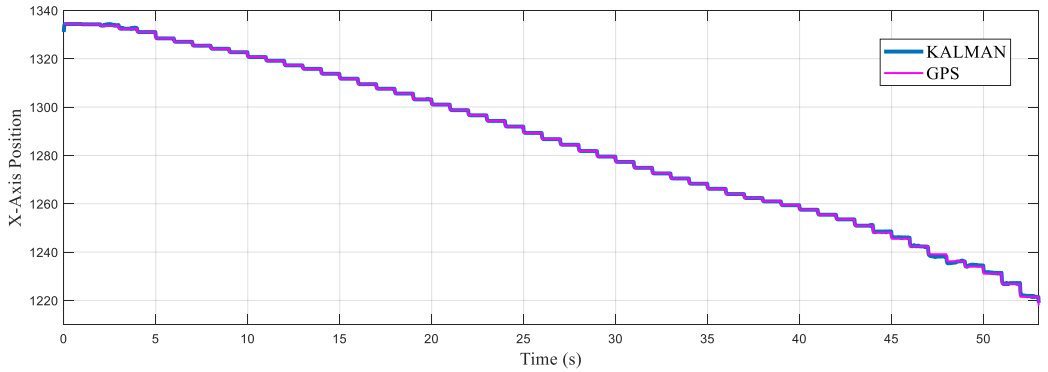
It is **off-line***

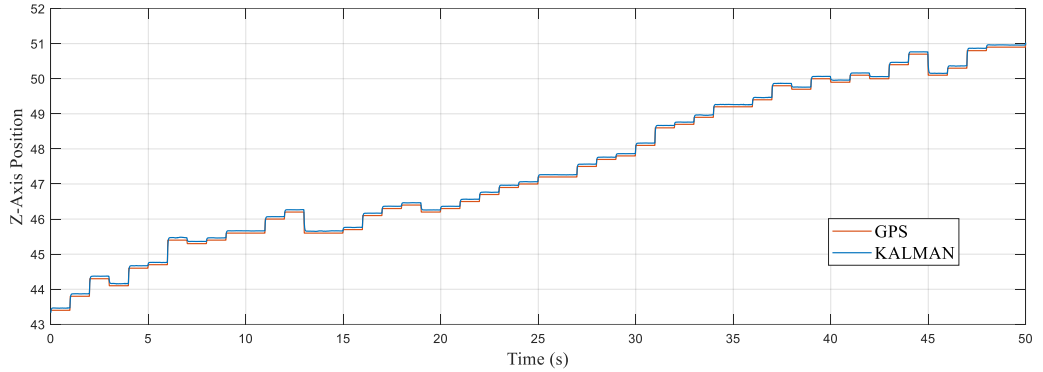
OFF-LINE: that it has been done after data collection.

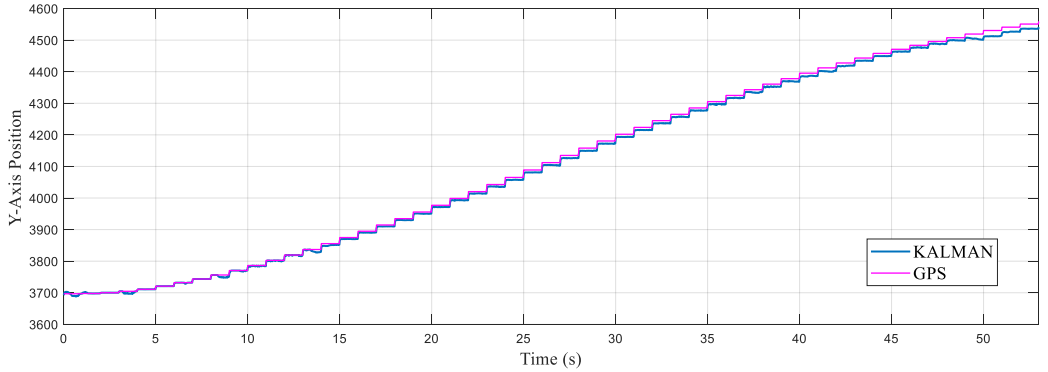
CRUCIAL CONCUSIONS AND CONSIDERATIONS ABAUT KALMAN FILTER

Kalman filter is not able to eliminate errors in the calibration of the sensors or in the modeling of the dynamic system that imply large variations on the real results. This is what happened in this case, the measurements taken by the IMU were not entirely reliable, so the measurements with the greatest weight were those of the GPS.

It is important to highlight the idea that a Kalman filter is not capable of deciding whether the measure or the model is correct or wrong. Its only task is to **estimate,** within the available data, an optimal solution free of random noise.







Some of the results obtained

Conclusions

The importance of certification in on board algorithm design has been emphasized throughout the whole project, as it was our main goal.

The airborne software systems certifications requirements have been briefly studied.

The basis for an Ada based certification-oriented approach has been laid out.

Differents trade-offs were evaluated and compromises were made in order to arrive at the best possible solutions.

Differents technologies have been researched to perform the different stages of our project.

Our knowledge about topics studied in the Navigation Aid Systems Course have been expanded and consolidated.

Futures lines of research

Design of any number of the procedures defined within the basic structure of the Ada implementation.

Development of a Python application to visualize the coordinates obtained from the Filter fusion algorithm.

Study of the considerations needed to analyze the certification process of the algorithm design.

Application of Kalman Filter fusion technique in real time

Improve the calibration and capacity of the devices and the quality of the data collection process to get more reliable the data

References

[1] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45. ISSN: 0021-9223. DOI: 10.1115/1.3662552. eprint: https://asmedigitalcollection.asme.org/fluidsengineering/article-pdf/82/1/35/5518977/35_1.pdf. URL: https://doi.org/10.1115/1.3662552.

[2] *UAV v2 Development Platform by SparkFun.* URL: https://www.sparkfun.com/products/retired/9038.

[3] *UAV v2 Development Platform User Manual.* URL: https://www.sparkfun.com/datasheets/GPS/GPSUAV2Manual_v29.pdf.

[4] *FTDI Basic Breakout.* URL: https://www.sparkfun.com/products/9873.

[5] *Hookup Guide.* URL: https://learn.sparkfun.com/tutorials/sparkfun-usb-to-serial-uart-boards-hookup-guide.

[6] *NMEA 0183.* URL: https://www.nmea.org/content/STANDARDS/NMEA_0183_Standard.

[7] *NMEA FAQ.* URL: https://web.archive.org/web/20140215150802/http://www.kh-gps.de/nmea.faq.

[8] *NMEA Senteces Table.* URL: https://www.trimble.com/OEM_ReceiverHelp/V4.44/en/NMEA-0183messages_MessageOverview.html.

[9] *CuteCom.* URL: http://cutecom.sourceforge.net/.

[10] *pySerial Library.* URL: https://pypi.org/project/pyserial/.

[11] *9 Degrees of Freedom - Razor IMU.* URL: https://www.sparkfun.com/products/retired/10736.

[12] *PYPL PopularitY of Programming Language.* URL: https://pypl.github.io/PYPL.html.

[13] *CSV File Reading and Writing.* URL: https://docs.python.org/3/library/csv.html.

[14] *CSV format.* URL: https://tools.ietf.org/html/rfc2048.

[15] *UTM library.* URL: https://pypi.org/project/utm/0.4.2/.

[16] John P. Snyder. *Map projections: A working manual.* U.S. Government Printing Office, 1987.

[17] *ISO/IEC 8652:2012 Information technology — Programming languages — Ada.* Standard. International Organization for Standardization/ International Electrotechnical Commission, Dec. 2012.

[18] J. Fuegi and J. Francis. "Lovelace Babbage and the creation of the 1843 'notes'". In: *IEEE Annals of the History of Computing* 25.4 (2003), pp. 16–26. DOI: 10.1109/MAHC.2003.1253887.

[19] *The Ada Programming Language.* URL: https://web.archive.org/web/20160522063844/http://groups.engin.umd.umich.edu/CIS/course.des/cis400/ada/ada.html.

[20] *The DoD High Order Language Working Group.* URL: http://archive.adaic.com/pol-hist/history/holwg-93/holwg-93.htm.

[21] *ANSI/MIL-STD 1815A.* URL: https://web.archive.org/web/20040625113309/http://archive.adaic.com/standards/83lrm/html/Welcome.html.

[22] *ISO 8652:1987 Programming languages — Ada.* Standard. International Organization for Standardization, June 1987.

[23] *ISO/IEC 8652:1995 Information technology — Programming languages — Ada.* Standard. International Organization for Standardization/ International Electrotechnical Commission, Feb. 1995.

[24] John W. McCormick and Peter C. Chapin. *Building High Integrity Applications with SPARK.* Sept. 2015. ISBN: 9781107656840.

[25] *Learn AdaCore.* URL: https://learn.adacore.com/about.html.

[26] *Who's Using Ada?* URL: https://www2.seas.gwu.edu/~mfeldman/ada-project-summary.html#Banking_and_Financial_Systems.

[27] *Buy DO-178C.* URL: https://my.rtca.org/nc__store?search=do-178c.

[28] *DO-178B and DO-178C Differences.* URL: https://www.patmos-eng.com/do-254-training-do-178c-training/178b-178c-differences/.

[29] *AC 20-115C.* URL: https://web.archive.org/web/20140903075843/http://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_20-115C.pdf.

[30] Frédéric Pothon and Quentin Ochem. *AdaCore Technologies for DO-178C / ED-12C.* Mar. 2017.

[31] *DO-331 Guidance for Model-Based Development and Verification.* URL: https://www.sae.org/learn/content/c2008/.

[32] *DO-332 Object-Oriented Technology and Related Techniques.* URL: https://standards.globalspec.com/std/1461703/rtca-do-332.

[33] *DO-333 Formal Methods.* URL: https://www.rtca.org/training/do-333-formal-methods-do-178c-supplement/.

[34] Mohsen Kavehrad and Reza Aminikashani. *Visible Light Communication Based Indoor Localization.* 2019.

[35] *El filtro de Kalman.* URL: https://www.bccr.fi.cr/investigaciones-economicas/DocMetodosCuantitativos/Filtro_de_Kalman.pdf.

[36] Carlos Caravaca Gallego. *Design of Extended Kalman Filters for Estimation of UAV Flight Parameters.* Istanbul Technical University, 2020.

[37] Eric Pula Moreno Juan Carlos Fernandez-Portillo Vicho and Miguel Martínez De la Hidalga Martínez. *FILTRO DE KALMAN. INTEGRACIÓN TIGHT INS/GPS.* Universidad de Sevilla, 2012.

[38] *Ada Standard library.* URL: https://docs.adacore.com/gnat_rm-docs/html/gnat_rm/gnat_rm/standard_library_routines.html.

[39] *Generics in Ada.* URL: https://en.wikibooks.org/wiki/Ada_Programming/Generics.

[40] *Ada packages.* URL: https://www.adaic.org/resources/add_content/standards/05aarm/html/AA-7-1.html.

[41] *Ada types.* URL: https://www.radford.edu/~nokie/classes/320/fundamentals/fundTypes.html.

[42] Maria Esther Aranda Romasanta. *Estudio y aplicación del Filtro de Kalman en fusión de sensores en UAVs.* Universidad de Sevilla, 2017.

[43] *AdaCore Tools.* URL: https://www.adacore.com/gnatpro/toolsuite.