Article

Small UAS-Based Wind Feature Identification System Part 1: Integration and Validation

Leopoldo Rodriguez Salazar *, Jose A. Cobano and Anibal Ollero

Robotics, Vision and Control Group, Universidad de Sevilla, 41004 Sevilla, Spain; jcobano@us.es (J.A.C.); aollero@us.es (A.O.)

* Correspondence: lrodriguez15@us.es; Tel.: +34-663-225-662

Abstract: This paper presents a novel system for identification of wind features such as gust and wind shear which are of particular interest in the context of energy-efficient navigation of Small Unmanned Aerial Vehicles (UAVs). The proposed system generates real-time wind estimations and real-time wind predictions as well as different alerts that are triggered when a certain feature is identified. This system uses information from a standard sensor suite to combine it in a big-data approach with stored information. Estimations are based on the integration of an off-the-shelf navigation system and airspeed readings in a so-called direct approach, whereas the predictions are the result of a statistical approach using atmospheric models to characterize and identify the features separately. This information is combined using a Genetic Algorithm in order to find the most probable wind speed at a certain position and fitting into a Weibull probability density function which meets the Prandtl's power law relationship. It also uses data from the ground station to produce an extrapolation in order to generate a full 3D wind map along with the prediction windows for each feature. Since both estimations and predictions are added to a wind database (DB), the information is constantly updated producing more accurate results. The knowledge of the wind features is crucial for computing energy-efficient trajectories, so the system not only provides a solution that does not require any additional sensor but produces enough information for any trajectory generation module. The integration of the system is described and a detailed system architecture is presented. Moreover, preliminary results for Software-In-The-Loop testing of the different modules as well as results utilizing collected data from different real flights that were performed in the Seville Metropolitan Area in Andalusia (Spain) are shown. Results show that wind estimation and predictions can be calculated at 1 Hz and a wind map can be updated at 0.4 Hz. Predictions show a convergence time with a 95 % confidence interval of approximately 30 s.

Keywords: wind prediction; wind estimation; UAVs; wind shear; gust; multi-platform integration

1. Introduction

Current UAVs technology has advanced in such a way that any unexperienced user is able to plan a route with relatively good accuracy. As the reliability has increased, more applications are plausible with the use of small UAVs and nonlinear natural effects such as winds can be compensated even with Commercial-Off-The-Shelf (COTS) components. Nevertheless, in order to compensate wind effects efficiently, the use of a sensor that can provide wind measurements is sometimes limited by the platform payload and the cost of such sensors. This leads to inefficient attitude compensations which produces drift and sometimes missing way-points which has as a consequence higher energy consumptions [1]. Currently, there are several research efforts to provide wind estimations without a direct measurement of the wind. Langelaan, et al. [2] proposed two ways of estimating the wind field, both using measurements from a standard sensor suite: Inertial Measurement Unit (IMU) and Global Navigation Satellite System (GNSS). The first method consists in a comparison between predictions generated with a dynamic model and actual measurements of the aircraft motion. The second one consists in the estimation on wind acceleration and its derivatives from the GNSS velocity, i.e. using the pseudorange rate of change together with direct measurements of the vehicle acceleration. Johansen,

et. al. [3] have developed a method in which the wind is estimated using an observer which leads to the calculation of sideslip and Angle-of-Attack (AOA). They estimate the wind from the difference between the platform velocity relative to the wind and the velocities in the body frame. A Kalman Filter to estimate the wind is also used in [4]. Other approaches such as the one presented by Larrabee et. al. [5] uses flow angle sensors and a Pitot tube with two Unscented Kalman Filters (UKF). This is innovative in a way that uses information cooperatively from different platforms in order to produce real time estimates. Neumman and Barholmai [6] produced wind estimations with a quadcopter-based UAV without the use of any additional sensors rather than its standard sensor suite even without a dedicated airspeed sensor and/or anemometer, based mainly in the wind triangle and the vector difference between the ground speed and an estimated speed. Condomines, et. al. [7] have published a set of results of a flight campaign with estimations of the wind field considering non-linear wind estimation with an square-root UKF in which the platform was equipped with an standard sensor suite which provides measurements that estimate angle of attack and sideslip.

Previously, as part of this research effort, the authors have introduced an algorithm that can estimate the wind field in such way that the different wind features (gust, shear, etc.) can be identified separately with a method that calculates statistical properties and based on distribution models of the wind, such as the $1 - \cos$ model for gusts and the wind shear model [8,9]. Lawrance and Sukkarieh [4] propose a method that incorporates a Gaussian regression in order to predict within a limited amount of time (up to 10 s) the local wind field despite the feature that is present. The identification of features, such as shear, thermals [10], gusts is of particular importance in the so-called atmospheric energy harvesting [4]. On this field, several authors such as Cutler et. al. [11] and Chakrabati et. al. [12] have published successful results on the generation of static soaring trajectories and others such as Montella and Spletzer [13] and Bird et. al. [14] have developed systems that produce and follow dynamic soaring trajectories. Despite the advances in the generation of the trajectories, methods for identifying wind features and the creation of real-time algorithms for energy harvesting should be addressed and improved. A few authors have described the integration of such methods in a level of detail that can identify areas of opportunities in hardware selection, software architecture, computational time, etc.

This paper presents and describes the detailed integration at a hardware and software level of a system that is capable of estimating the wind field and identifying in real-time wind features with an standard sensor suite. In the previous works of the authors [8], [9], a method to identify wind features is presented. Each feature is identified separately based purely on statistical analysis by generating most probable wind speeds at different locations by fitting into a Weibull distribution. Current method allows the generation of a 3-dimensional wind map with predictions of what the wind vector would be at a certain location, intended purely for trajectory planning or re-planning. In addition a detailed description of the statistical analysis is presented and also how the big-data analysis is utilized in order to store and use the information stored to produce more accurate predictions. Then, On-The-Loop testing is performed, allowing the validation of the system. This is performed by using data of previous flights to produce wind estimations and predictions off-line.

The paper is organized as follows Section 2 presents a brief summary of the methods and statistical analysis utilized. Section 3 describes the hardware and software architecture of the system. Section 4 shows the validation results of the different components and Section 5 presents a discussion on the obtained results. Finally, conclusions are presented in Section 6.

2. Wind Field Estimation and Wind Field Prediction

The generation of the wind field considers both the estimation and prediction processes. Both are equally important, however they not necessary have to occur at the same time and rate. This section provides the insights of the selected methods for these operations.

2.1. Wind Field Estimation

The selected method for estimation process was originally presented in [2]. It estimates the wind without the use of an observer (such as Kalman or Particle filters) by using the velocity vector calculated by the GNSS module together with measurements of the vehicle acceleration and a portion of the state vector of the platform. The goal is to calculate the wind acceleration using the relationship between the GNSS velocity and the body-axis state from the COTS Autopilot Module (APM), which provides filtered airspeed and the Euler Angles. Two relationships are used. The first one indicates the relationship between the vehicle kinematics and the GNSS velocity [9]:

$$\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_{\text{GNSS}} - \left(\mathbf{C}_e^b\right)^{-1} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{1}$$

where (w_x, w_y, w_z) is the wind velocity vector, $(\dot{x}, \dot{y}, \dot{z})$ is the GNSS velocity vector and (u, b, w) is the body axis velocity vector and $\mathbf{C}_e^b$

In order to perform the estimation of the wind velocity using eq. (1), a second equation is necessary. Therefore, the wind acceleration at the step $(k-1)$ can be computed:

$$\begin{bmatrix} \dot{w}_x \\ \dot{w}_y \\ \dot{w}_z \end{bmatrix}_{k-1} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_{b,k-1} - \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}_{k-1} + \begin{bmatrix} -g\sin\theta \\ g\cos\theta\sin\phi \\ g\cos\theta\cos\phi \end{bmatrix}_{k-1} - \begin{bmatrix} qw - rv \\ pw - ru \\ qu - pv \end{bmatrix}_{k-1} - \frac{1}{2\Delta t} \begin{bmatrix} u_k - u_{k-2} \\ v_k - v_{k-2} \\ w_k - w_{k-2} \end{bmatrix} \tag{2}$$

where (a_x, a_y, a_z) is the body axis accelerations vector, (p, q, r) is the rotation rate vector, (b_x, b_y, b_z) are the accelerometer bias, θ is the roll angle and ϕ is the pitch angle.

Note that the navigation (GNSS/IMU) data and the airspeed data are filtered inside the autopilot, therefore, the wind speed vector can be estimated from eqs. (1) and (2) without the use of an optimal filter.

2.2. Wind Field Prediction

The wind field could be estimated at each time step from the data provided by the IMU, GNSS and the vehicle dynamics shown in section 2.1. Previous results [8,9] show that the estimation algorithm produce accurate results. However, these estimations are not sufficient if the information is going to be used for trajectory planning, or even to correct the trajectory in real-time to compensate the wind effect. Therefore, a prediction stage is needed so that the wind field could be inferred within a reasonable time window.

In this context, three models of different wind features have been selected: the wind shear model, the discrete gust model and the continuous (Dryden) wind turbulence model. These are widely used in the aerospace industry and are contained in the Military Specification MIL-F-8785C [15] and Military Handbook MIL-HDBK-1797 [16].

2.2.1. Wind Shear Model

The magnitude of the wind is modeled by the following equation:

$$W_{\text{shear}} = W_{20} \frac{\ln \frac{h}{z_0}}{\ln \frac{6.096}{z_0}}, 1\,\text{m} < h < 300\,\text{m} \tag{3}$$

where W_{shear} is the mean wind speed, W_{20} is the wind speed at 20 ft (6.096 m) and z_0 varies depending on the flight phase. However, a value of 0.0457 m (0.15 ft) is selected due to the characteristics of the platform, i.e. flight below 1000 ft. Finally, h is the actual altitude of the vehicle.

The wind shear is illustrated in fig. 1:

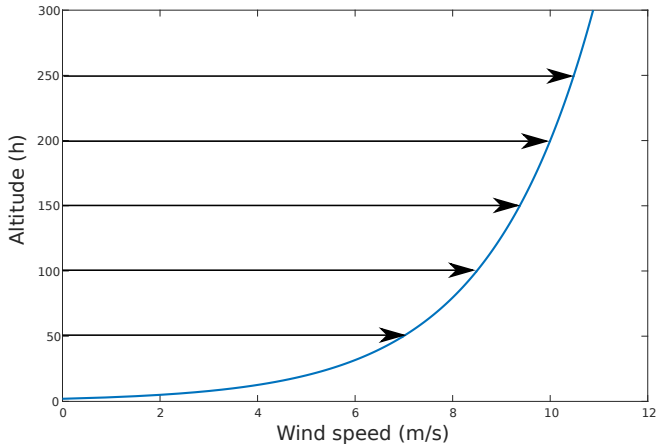


Figure 1. Typical shear profile that shows the increase of wind speed over as the altitude increases, note the relationship is exponential between the two variables.

In order to characterize the wind shear, it is necessary to assume that the wind varies with the altitude following the Prandlt Ratio based on an Empirical Power Law (EPL) [8]:

$$\frac{W_1}{W_2} = \left(\frac{h_1}{h_2}\right)^{\xi} \tag{4}$$

where ξ is the Prandtl coefficient that shapes the EPL function. (W_1, W_2) are two wind speeds and (h_1, h_2) are the corresponding altitudes.

2.2.2. Discrete Gust Model

This model uses the implementation of the $1 - \cos$ shape and its mathematical representation is as follows:

$$W_{\text{gust}} = \begin{cases} 0 & x < 0 \\ \frac{W_m}{2}(1 - \cos\frac{\pi x}{d_m}) & 0 \leq x \leq d_m \\ W_m & x > d_m \end{cases} \tag{5}$$

where W_m is the magnitude of the gust and d_m is the gust length and x is the distance traveled. The discrete gust is illustrated in fig. 2:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2016 doi:10.20944/preprints201611.0002.v1

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

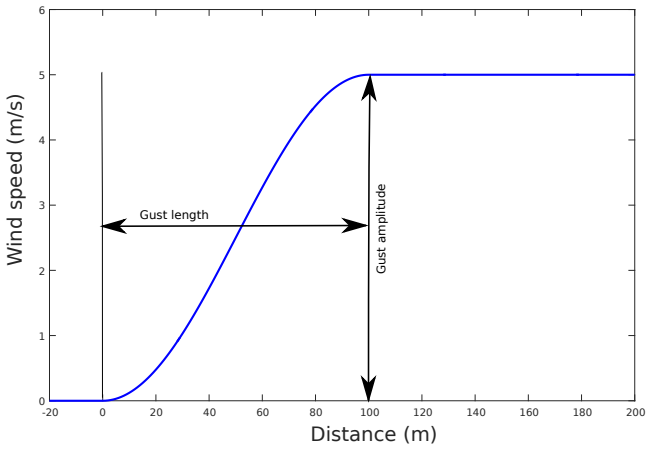


Figure 2. Typical discrete gust profile that shows a growth over the wind on a short period of time from the initial wind speed of the gust magnitude, and a permanent increase at the end of the gust length.

2.2.3. Continuous Gust Model

The selected model for continuous gust utilizes the Dryden spectral representation in which the turbulence is considered a stochastic process defined by velocity spectra. In [15–17], the power spectral densities are defined. Note that for simulation purposes the Low-Altitude scale lengths have been used.

The number of variables in the continuous gust model is vast. Therefore, inferring these values from actual wind measurements trough a regression is very complex. Thus, this model is used only as a simulation input. Furthermore, two methods are proposed for continuous gust identification in both short and long term. The first one incorporates a Standard Gaussian Process (GP) Regression [18]. Considering a set of vertical wind observations of size $\bar{M}$, $\hat{\mathbf{W}}_{\mathbf{z}} = \hat{W}_{z,i}|_{i=1}^{\bar{M}}$. The wind speed prediction $\bar{W}_p(x)$ at any location x can be expressed as:

$$\bar{W}_p(x) = \sum_{i=1}^{\bar{M}} k_i \hat{W}_{z,i} \tag{6}$$

in which k_i is the i-th coefficient of the linear combination of wind measurements $\hat{\mathbf{W}}_{\mathbf{z}}$. Based on the work presented by Park et. al. [19], an optimal coefficient is determined by minimizing the prediction error.

$$\min_{\mathbf{k}} E\left[(\bar{W}_p(x) - W_p(x))^2\right] = \min_{\mathbf{k}}(k^T\left[\mathbf{Q}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}\right]k - 2k^T\mathbf{q}(\mathbf{X},x) + q(x,x)) \tag{7}$$

which can be determined by calculating the covariance matrix $\mathbf{Q}(\mathbf{X},\mathbf{X})$ and the covariance vector $\mathbf{q}(\mathbf{X},x)$ between every two observations at locations $\mathbf{X}$ and x; finally $q(x,x)$ represents the covariance value . This leads to express standard GP regression of the linear predictor as:

$$\bar{p}(X) = \mathbf{k}^T\hat{\mathbf{W}}_{\mathbf{z}} = \mathbf{q}(x,\mathbf{X})\left[\mathbf{Q}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}\right]^{-1}\hat{\mathbf{W}}_{\mathbf{z}} \tag{8}$$

and the covariance value $cov(\bar{p}(x)))$ can be expressed as:

$$cov(\bar{p}(x)) = q(x,x) - \mathbf{q}(x,\mathbf{X})\left[\mathbf{Q}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}\right]^{-1}\mathbf{q}(x,\mathbf{X}) \tag{9}$$

where σ_n^2 is the measurement noise covariance.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2016 doi:10.20944/preprints201611.0002.v1

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

An alternative approach can be used to perform long-term predictions by employing a non homogeneous regression prediction model. Lerch and Thoraninsdottir [20] have performed a comparison between three non-homogeneous regression model which allows to produce predictions within the amount of days. Since the intention of the different flight campaigns is to store the data into a single database, a big amount of data can be utilized to perform these predictions.

At this stage, the truncated normal model was selected as a form of wind estimation. Being W the wind speed and $X_1, \ldots, X_j$ the ensemble member forecasts, the predicted distribution of W can be approximated by a truncated normal distribution:

$$W|X_1, \ldots, X_j \sim \mathbf{N}_{[0,\infty)}(\mu, \sigma^2) \tag{10}$$

where the mean μ is an affine function of the ensemble forecast and the variance σ^2 is an affine function of the ensemble variance. If these exchange members are exchangeable [20], the distribution function of the Truncated Normal (TN) distribution $F(z)$ is given by:

$$F(z) = \Phi\left(\frac{\mu}{\sigma}\right)^{-1} \Phi\left(\frac{z-\mu}{\sigma}\right) \tag{11}$$

for $z > 0$, where Φ is the cumulative standard normal distribution.

This is indeed a simple non-homogeneous method. However, results indicate that the training period to produce accurate predictions in one-day ahead forecasts is of the equivalent 30 days of continuous measurements [20].

2.2.4. Weibull Distribution

The Weibull distribution is a key part of the research performed as many datasets, including wind speed have been proved to fit in. The Weibull distribution has three main parameters, the shaping factor κ, the scaling factor ν and the threshold. Given a dataset $\mathbf{W}= (W_1 ... W_n)$, the Weibull probability density function can be expressed as a function of a wind magnitude W [8]:

$$f(W) = \frac{\kappa}{\nu}\frac{W}{\nu}^{\kappa-1} e^{\frac{W}{\nu}\kappa} \tag{12}$$

From this function the most probable wind speed W_{mp} at a particular location can be expressed in terms of the Weibull parameters:

$$W_{\mathrm{mp}} = \nu(1 - \frac{1}{\kappa})^{\frac{1}{\kappa}} \tag{13}$$

A typical Weibull distribution is shown in fig. 3:

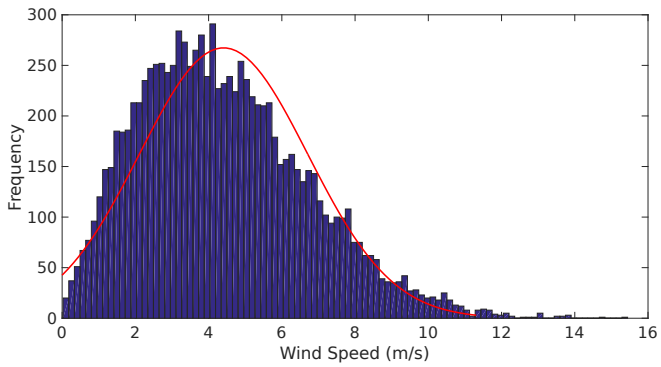


Figure 3. Typical weibull distribution from a National Oceanic and Atmospheric Administration NOAA measurement station at an altitude of 30 m.

Table 1. ODROID-C2 Relevant Characteristics from [22]

CPU	Amlogic S905 SoC, 4 x ARM Cortex-A53 2GHz, 64bit ARMv8 Architecture @28nm
RAM	2GB 32bit DDR3 912MHz
Flash Storage	Micro-SD UHS-1 @83Mhz/SDR50, eMMC5.0 storage option
ADC	10bit SAR 2 channels
Size	85 x 56 mm (3.35 x 2.2 inch)
Weight	40g (1.41oz)

2.3. Wind Mapping

In order to generate a full 3D wind map, a combination of methods is required. Initially, the work presented in [8] suggests the use of a Newton polynomial extrapolation in order to generate local values of the Prandtl coefficient, ξ, from which the shaping and scaling parameters can be calculated in order to obtain a local most probable wind speed W_{mp}. This is true in case of the presence of a wind shear. However, if a gusts is detected, the extrapolation will accumulate error, producing an inaccurate wind map.

Therefore, a 3D map can be generated based on the feature that is detected. The GP regression shown in section 2.2.3 allows the generation of a wind map with predictions based on the estimates found at position X. These estimates carry information of the covariance which is continuously updated with the different feature detection algorithms.

3. System Architecture

This section describes the hardware, software and communication architecture of the wind identification system.

The selected hardware takes mainly two COTS components in order to perform the estimation and the prediction of the wind field. The selected autopilot is the 3DR-Pixhawk which is based on the PX4 open-hardware project. The characteristics of this module can be found in [21]. Given the processor characteristics, the wind estimation and wind prediction algorithms have to reside in a dedicated computer. In this paper, the selected computer is the ODROID-C2 [22] that contains a quad-core processor at 2 Ghz at 64 bit. The main characteristics are enumerated at table 1:

The required algorithms need an additional platform that shall do the data analysis of the stored variables. All the wind estimates and predictions are kept in a database. As more flights are to be performed as part of the validation, verification and other applications, the wind database will grow. Due to its size and for reliability, a ground station contains the wind prediction and estimation database. In this work, a PC with an ©Intel Core(TM) i755000U CPU at 2.4 GHz with 16GB of RAM is considered.

The software design has followed a deep evolution since its conception. Initially this system was created in a multi-platform way with different computing language interacting at a very high level. The intention of the proposed architecture is to minimize these interfaces at component level in order to enhance the maintainability and upgradeability. In the architecture shown in fig. 4, the autopilot sends information from a request made by the communication module, this information is sent to the wind estimation algorithm that generates wind estimates that will go to the prediction block which uses information from the wind database and also calls the storage module once a prediction is performed.

As it was mentioned before, the designed architecture considered a diversity of programming languages and even various operating systems. The modules communication of this system was done

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

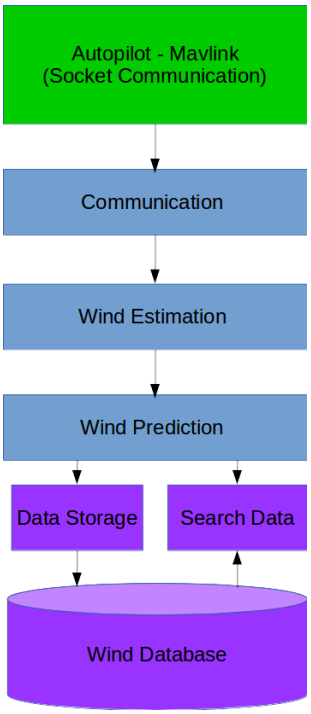


Figure 4. High level software architecture that shows the flow of information of the wind identification system.

in Linux with pymavlink [1]. The wind estimations and the simulation test-bed with the models shown in section 2 were done using MATLAB® and the Simulink® environment with S-Functions. Finally, in order to generate a database, initially the idea was to create comma separated (*.csv) files, however, Structured Query Langate (SQL) was selected to be utilized for Database Accessing and Management, which required Java and C++ connector of SQL.

After observing problems in the synchronization of the various system, the solution was to migrate everything to C++ leaving only the database management in JAVA with the MySQL® C++ connector. The concept was to build a modular architecture that runs under a handler that manages the communication between the various modules that interact to identify the wind (see fig. 5).

The modules are the same shown in fig. 4 plus the alerts generation. Below the block name, each module receives inputs and outputs from other modules: Autopilot (AP) communiction request, Wind Estimation (We), Wind Prediction (Wp), Database Management (Dm), Database Search (Ds) and Alerts Generation (Ag).

An advantage of the modular implementation is that the system can be easily expanded to provide additional functionalities besides the wind identification system. This was thought in order to be able to integrate trajectory optimization functions and controlling modules to follow the desired trajectory.

The details on the implementation of the modules are described in sections 3.1 to 3.5.

3.1. Communication Block and Handler

The explanation of the communications is divided in two parts. The first one, described in section 3.1.1 analyzes the details of the communication between the three main hardware components:

[1] MAVLINK (Micro Air Vehicle Communication Protocol) is a communication library for UAV that can pack C-structures over a serial channel and send this packets with other modules. It was originally release in 2009 by Lorenz Meier with a GNU Lesser General Public Licence (LPGL). Pymavlink is a Python implementation of MAVLINK [23].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2016 doi:10.20944/preprints201611.0002.v1

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

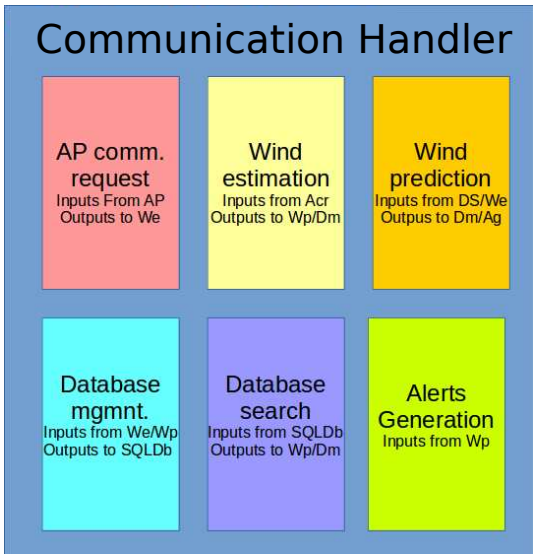


Figure 5. Architecture design with communications handler.

the ODROID, the Pixhawk and the PC with SQL. The second part, section 3.1.2, explains the details of the communication between the functional software blocks.

3.1.1. Hardware communication

The hardware communication is performed by a C++ Software derived from the MAVCONN software created by Lorenz Meier as a complementary MAVLINK toolset [24]. The main characteristic is the low latency that allow the communication between processes approximately at 100 microseconds. The system is implemented asynchronously, which allows that the data is sent immediately after it is available. The asynchronous communication is an alternative solution to the widely use polling which is proven to require extra CPU resources because of the context switch. On the other hand, asynchronous design requires minimum CPU resources. Nevertheless, it needs a multi-threaded implementation which is computationally more complex. The ODROID computer allows this type of implementation. Further details of this implementation can be found in [25].

3.1.2. Module Communication

The communication between modules is managed by a handler (see fig. 5. Each module publishes its information at a certain order based on a request and the importance of the information. Therefore, if a module requires priority information the framework will designate this request over others. Table 2 shows the selected requirements in terms of communication rate and an assigned priority based on the importance of its information to other subsystems.

Table 2. Communication Scheme of Handler. The publishing rate which was determined based on the results found in [9] and the priority based on the system requirements.

Function	Publishing rate	Priority
APM Comm Request	2 Hz	1
Wind Estimation	1 Hz	1
Wind Prediction	0.25 Hz*	2
Database Management	0.2 Hz	3
Database Search	0.25 Hz	2
Alert Generation	0.1 Hz	4

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

* This rate was selected due to the current time required to scan the wind database. Future work will optimize this rate allowing the generation of predictions at a lower rate.

The main advantage of this system is the modularity, since the intention is to have total independence between systems. If there is any communication problem, or the data is proved to be corrupted, this is handled directly by the communication handler which will continue to serve the other functions to preserve the overall integrity.

The processes with highest priority of publishing are the communication request between the ODROID and the PIXHAWK and the wind estimation processes (see Table 2). The first one was based on the publishing rate of the information available from the User Datagram Protocol (UDP) connection with MAVLINK. The second one was based on the computational time that requires the prediction which was subject of previous study in [8,9].

3.2. Wind Estimation Module

The wind estimation module uses the information from the autopilot in order to generate real-time 3D estimations of the wind field. The implemented algorithm is shown in algorithm 1:

Algorithm 1 Wind Estimation Algorithm

1: **procedure** WINDVEL(V_{GNSS}, C_b^e,$\dot{W}$)
2: **while** $k > 2$, $\dot{W} > 0$ **do** ▷ Start calculation at least with two measurements.
3: **if** CalcWindVel = False **then**
4: W=$V_{GNSS} - (C_b^e) * V_b$; ▷ From eq. (1);
5: CalcWindVel = True;
6: break;
7: **else**
8: break;
9: **end if**
10: **end while**
11: **end procedure**
12: **procedure** WINDACCEL($\hat{A}$, $\hat{B}$,$\hat{V}_b$,R,Φ) ▷ Calculates Wind Acceleration at previous timestep.
13: **while** $k > 2$ **do**
14: **if** CalcWindAcc = False **thenç**
15: getPreviousV();
16: **end if**
17: **return** WindAccel; ▷ From eq. (2)
18: **end while**
19: **end procedure**

where $\hat{A}$ is the acceleration vector $\hat{B}$, is the accelerometer bias vector $\hat{V}_b$ is the body axis velocity,R is the rotation rate vector andΦ is the attitude vector.

All the variables are requested to the communication handler based on the priorities and frequencies illustrated in table 2.

3.3. Wind Prediction Module

The implemented prediction algorithm (see algorithm 2) is able to identify different wind features. First, it performs an statistical analysis on the current samples after a number n of wind estimations. Based on this statistical analysis, the system determines whether a wind feature is detected and if possible identifies it. It could be a gust, shear or an unidentified feature (generally gust). Then, it generates the corresponding flags that will trigger the alert system.

The wind prediction system initially assumes that the detected feature is either a discrete gust or a shear. The system looks for anomalies in the behavior of the standard deviation (see line 5). The estimations are grouped based on the nearest altitudes of a reference value (see line 18). Then, it detects the shaping and scaling parameter of the Weibull distribution using Genetic Algorithms (GA) (see line

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

25) in order to find the parameter that results into the minimum residual error between the mean and the standard deviation of the measurements [9]. After the shaping parameter κ is detected, the scaling parameter v is calculated. Then the most probable wind speed at the reference altitude is calculated (see line 32). With a group of most probable wind speeds, the Prandtl coefficient is calculated and if the difference between the estimates and the prediction curve is bounded it alerts the presence of a shear. If the estimates present a punctual jump and then the error is bounded it triggers the alerting of a discrete gust. However, if the error is unbounded the system considers that a continuous gust is present and starts the Gaussian regression in order to produce short-term predictions (see line 43).

Algorithm 2 Wind Prediction Algorithm

```
1:  procedure DetectFeature(WDb)          ▷ Requires Wind Database with at least 30 elements.
2:      while Size(WDb ≥ 30) do
3:          CalcMean(WDb);
4:          CalcStd(WDb);
5:          if Std(Wdb ≥Threshold) then
6:              FeatureDetected=True;
7:              break;
8:          end if
9:      end while
10:     if FeatureDetected = True then
11:         RaiseFlag(FeatureDetected);
12:     end if
13: end procedure
14: procedure IdentifyFeature(WDb)
15:     while FeatureDetected = True do
16:         WDb=Order(WDb);
17:         for j ← min h to max(alt) do
18:             NearAlts= FindNearAltitudes(j, WDb(h)); ▷ Group altitudes close to a reference value j
19:         end for
20:         DetectJumps(NearAlts);
21:         if JumpDetected = True then
22:             DiscGustDetected = True;
23:             ξ=CalcPrandlt;
24:         else
25:             κ = GeneticAlgorithm(NearAlts(WDb));   ▷ Detect Shaping Parameter with GA.
26:             ShearDetected=True;
27:     end if
28:     if Exist(κ) then
29:         v =CalcUpsilon;                ▷ Calculate scaling parameter based on [8].
30:     end if
31:     if Exist(κ and v) then
32:         CalcWMostProbable;             ▷ Calculate Most Probable Wind Speed
33:         ϵ=CalcError(WDb);
34:     end if
35:     if ϵ,Std≥threshold then            ▷ If the standard deviation does not stabilize
36:         ContGustDetected = True;
37:     end if
38: end while
39:         if ContGustDetected or ShearDetected = True then
40:             RaiseFlag(ContGust);
41:         end if
42:         if (ContGustDetected=True) then
43:             PerformGaussianRegression;                ▷ See note *
44:         end if
45:     end procedure
```

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

* The system may perform a long-term and a short term prediction. For this research activity only the short-term which is a Standard GP regression. The non-homogeneous GP regression requires a vast amount of information which is part of future activities.

3.4. Data Storage and Wind Database

An important part of the designed system is the storage and management of the information that is generated by the estimation and the prediction modules, together with the information that is generated by the autopilot (vehicle state: position, velocity, acceleration).

SQL is selected as a means of the generation, storage and management of the database. This was because SQL is a standardized language for database management. SQL is a language by itself, therefore, it requires an interface with the wind identification system. The SQL system that is selected for this research is MySQL® and the interfacing between the database and the wind identification comm-handler is done trough the MySQL® C++ connector. This allows the generation of C++ commands that will read and write information from any SQL database.

The communication scheme is shown in fig. 6:

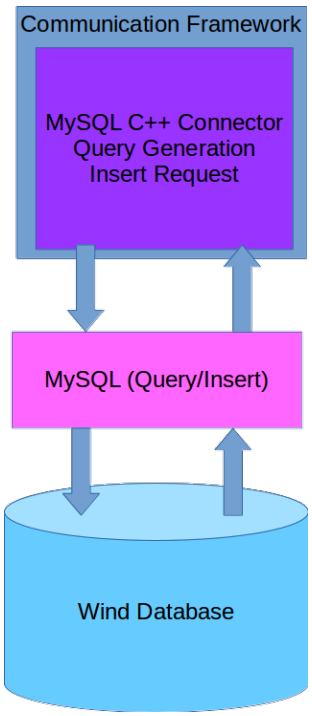


Figure 6. Database interfacing with communication handler.

Figure 6 illustrates the two modules that are required to interact with wind database. One is the MySQL C++ connector and the other the MySQL system, which have to be compatible with the used operating system.

The algorithm for accessing the database, perform a query of the useful data and write the generated data for the modules consists in a series of calls to the SQL connector which needs to open a Connection to the SQL server and then to execute an update or a query to the database based on the parameters that are needed. This is described in algorithm 3.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2016 doi:10.20944/preprints201611.0002.v1

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

Algorithm 3 Database Access, Query and Writing Algorithm

```
1: procedure WaitForRequest(DBReq)
2:     if DBReq=Write then
3:         WriteDB(DB);
4:     else if DBReq=Query then
5:         Query(DB,Cond);
6:     else
7:         TriggerException;
8:     end if
9: end procedure
10: procedure WriteDB(DB,WindVector)
11:     Con→ CreateDriver();                          ▷ Create Database Driver
12:     Con→GetDriverInstance();   ▷ Used to get the Driver Instance and Load the DB.
13:     Con→setSchema(DB)                             ▷ Set the DB to write to
14:     Stmt→WindVector
15: end procedure
16: procedure QueryDB(DB,Cond) State Con→ CreateDriver();
17:     Con→GetDriverInstance();
18:     Con→setSchema(DB);
19:     execute;
20:     stmt→executeQuery(Condition);                 ▷ See note below.
21: end procedure
```

Note that the *CONDITION* variable in the query procedure is a string in the form *SELECT CONDITION AS MESSAGE* which is a SQL command with the conditions necessary to perform a query. In this case, the Conditions are determined in a different algorithm that determines which of them are going to be used in order to optimize the query. The main condition is the altitude, therefore it looks for values closer to the corresponding height +/- 5 m. Then it performs a query based on the location the time of the year. Since the location of a typical mission may not vary the data should be valid, however a future step is to include the METAR information to the query so that it only looks for wind predictions and estimations performed in similar meteorological conditions.

3.5. Alert Generation Module

A complementary part of the estimation module is the alert generation algorithm. This part illustrates what kind of alerts need to be triggered internally to the system and to the user so that it can take a decision. These alerts are related to the detection of a feature and the uncertainty both in time and space of the predicted feature. In addition there are different alerts that are generated inside each module related to the information that is generated, these includes the generation of exception.

Table 3 shows the alerts that are generated once a feature is detected, the variable type of the alert and the priority.

Table 3. Alert Types. The following table contains the possible alerts generated, its range (if applicable) and its default value.

Alert Type	Type	Value
Feature Is Present	Boolean	1
Wind Shear Detected	Boolean	1
Discrete Gust Detected	Boolean	2
Continous Gust Detected	Boolean	3
Prediction Time Window	Integer	2
Uncertainty Level	Double	4

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

The priorities of the alert help on determining how often these should be informed and the intention of the system is to display this alerts in the ground station.

The prediction time window and uncertainty level require additional computational resources. In the first case, if a discrete gust o a shear is detected, the time window is long enough so that an alert is not required. However, if a continuous gust is detected the time window of the prediction goes critical depending to the behavior of the error between the prediction and the estimation. If the error is bounded, then the prediction window goes up 5 more seconds based on a typical prediction window of discrete gust [26].

4. Experimental Results

This section presents the preliminary results of the system obtained with Autopilot/Framework software-in-the-loop experiments and simulations. The results presented here provide validation towards the system requirements.

Initially a simulation test bed will be described, following by the simulation results with its respective analysis. Several scenarios have been simulated, however only one is shown to illustrate how the system works. Finally a Autopilot/Framework in-the-loop test is presented with real telemetry data obtained by four flights which took place in Seville Metropolitan Area in Andalusia (Spain).

4.1. Simulation Test Bed Description

Table 4 show the characteristics of the computer used in order to perform the simulations.

Table 4. Simulation Computer Relevant Characteristics

CPU	Intel Core i7-5500U CPU 2.40GHz x 4
RAM	15.6 GiB
Graphics	Intel HD Graphics 5500 (Broadwell GT2)
OS Type	64-Bit
OS	Ubuntu 16.04 lts

MATLAB® and Simulink® has been utilized in order to perform the simulation as its Aerospace toolbox contains wind-model blocks of shear, discrete and continuous gusts. In addition the AeroSim® blockset has been utilized in order to generate a 6DOF model of a small UAV with the trimming parameters indicated in Table 5.

The Simulink model utilized together with the wind dynamic model blocks is shown in fig. 7.

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

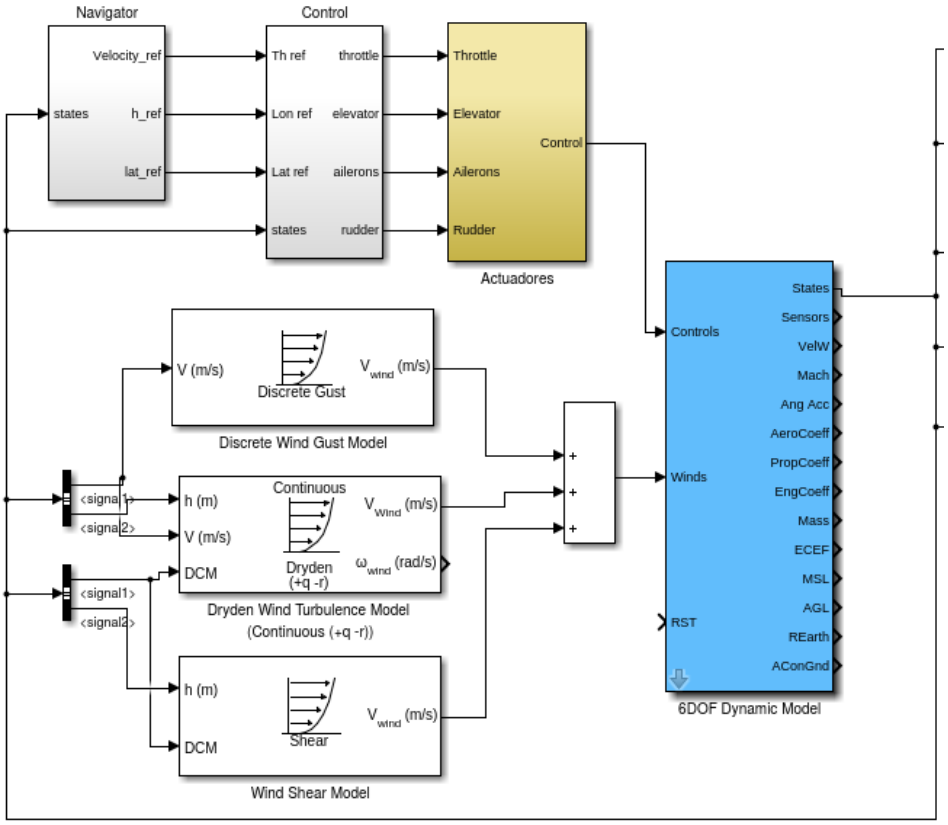


Figure 7. Simulink dynamic model of the simulation environment for the wind identification system. The blue block shows the 6DOF dynamic model and the white blocks show the wind dynamic model. In addition, there is an actuator block that represents the dynamic model of the actuators. There are two additional blocks that show the navigation and the control modules which are a series of nested Proportional-Integral-Derivative (PID) controllers

Table 5. Selected Trimming Parameters

Trim airspeed	25 m/s
Trim altitude	150 m
Trim bank angle	0 degree
Fuel mass	2 kg
Flap setting	0

The simulation scenario consists of a Helix trajectory with ascending altitude. This trajectory is illustrated in fig. 8.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2016 doi:10.20944/preprints201611.0002.v1

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

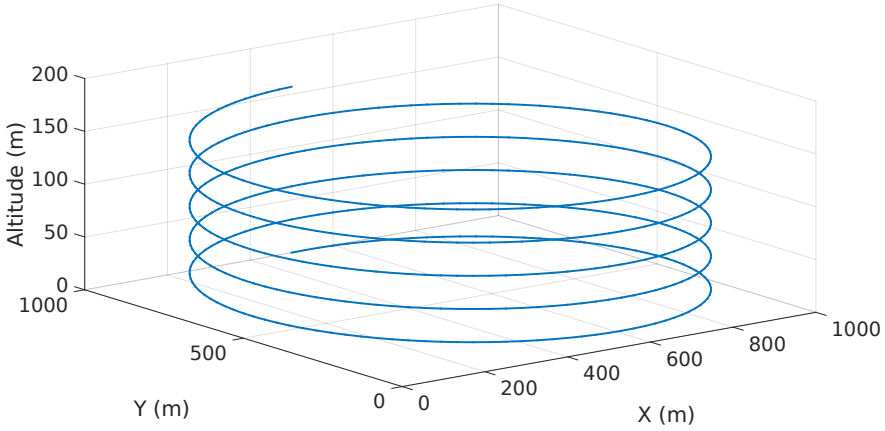


Figure 8. Simulated Helix Trayectory

Once the UAV starts the flight, its trajectory is under the influence of different wind types. Two types of influence are considered. The first one considers each feature separately (shear, discrete gust, continuous gust) and the second one considers all features at the same time. The purpose of this simulation is to prove the ability of the system in controlled conditions of detecting the features separately.

Figure 9 depicts the wind effects on the trajectory.

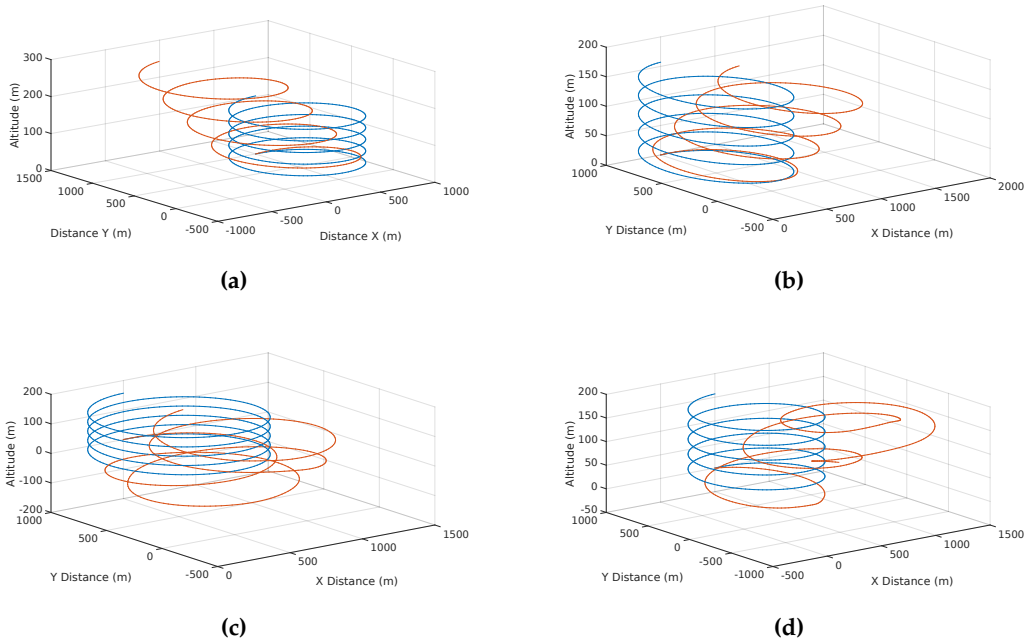


Figure 9. Effects of different simulated features on UAV trajectory: (a) the effect of a shear wind with increasing deviation as altitude rises; (b) the effects of a discrete gust with a constant deviation on the trajectory in a single direction; (c) a chaotic deviation due to the effects of a continuous gust and finally (d) the total effects of the wind present at the same time.

4.2. Simulation Results

The detection capabilities of the system are illustrated fig. 10. It shows the information that feeds up the system and how it will detect and identify the different features.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2016 doi:10.20944/preprints201611.0002.v1

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

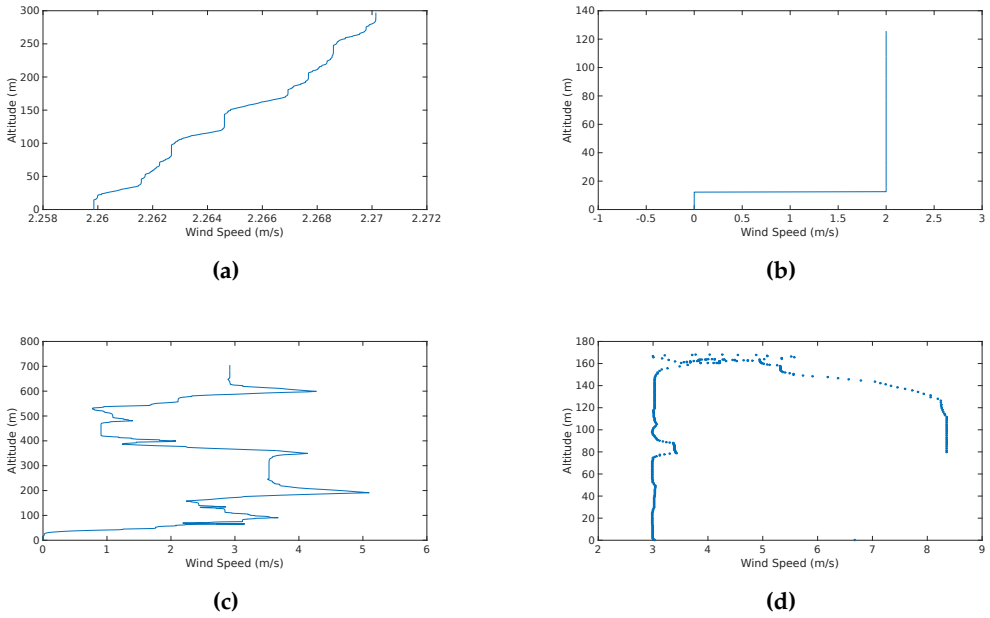


Figure 10. Wind Speed/Altitude maps of the different simulation scenarios: (a) the wind shear as an increase of wind speed with altitude; (b) a sudden increase in wind speed at a certain altitude (discrete gust); (c) a continuous gust with a chaotic effect and rapid increases and decreases of wind speed; and (d) the sum of the three effects.

The presence of parallel lines indicate a discrete gust as shown in fig. 10d [9]. Since the shear effect is very little, it appears that the wind tends to be of approximately 3 m/s.

The results of the estimation and prediction of the wind are show in fig. 11 and the error in the prediction and the estimation is shown fig. 12

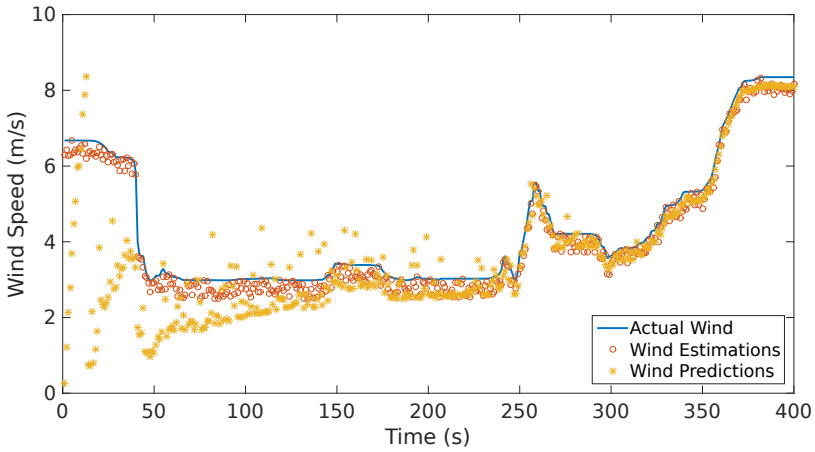


Figure 11. Actual, estimated an predicted wind speed in the considered scenario. The predicted wind starts with high dispersion, however, it converges to the actual value within 100 seconds.

.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2016 doi:10.20944/preprints201611.0002.v1

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

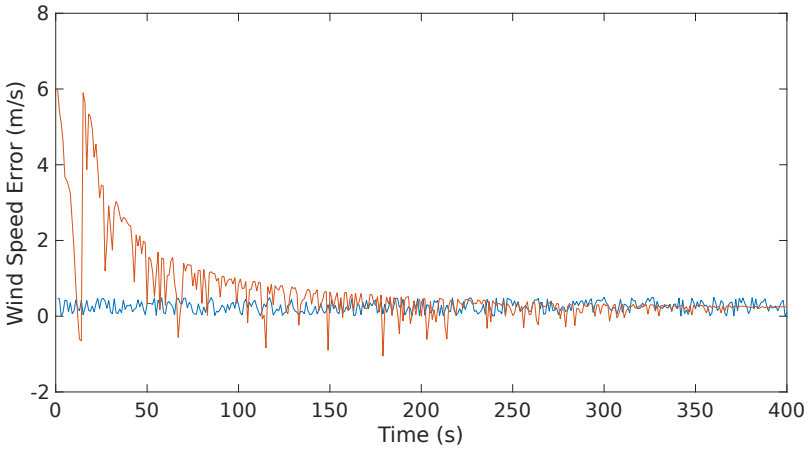


Figure 12. Actual, estimated an predicted wind. Error is higher in the first predictions whereas it starts decreasing with time as previous estimates are generated.

In this flight, an alert of a continuous gust detected was triggered almost immediately (at approximately 20 s). In addition, there was an alarm of two detected gusts: one occurred at approximately 40 s and the other occurred at 250 s. This coincides with the jumps, abrupt changes of the wind speed, that can be seen in fig. 11.

4.3. Software-In-The-Loop Experiments

The wind identification system has been functionally tested with real telemetry data. The data were fed into the system using Mavlink interfacing with a Ground Control Station as in [27]. The sensor information was transmitted to the wind identification system at a rate of 0.5 Hz. Nevertheless the communication framework demands varied in frequency due to the asynchronous scheme. The transmission to the system does not match the actual duration of the telemetry log, it was truncated once UAV had landed.

The architecture of the designed test required the connection of the telemetry data and the transmission using ROS Kinetic that transmits the data to MATLAB® that calls the wind estimation system towards a series of S-Functions as illustrated in fig. 13.

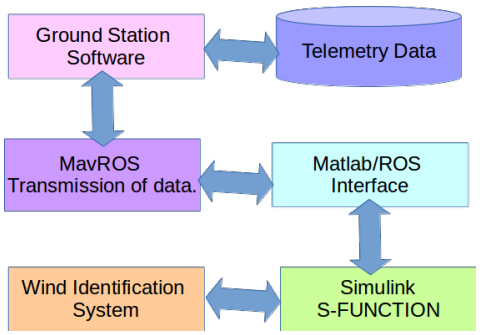


Figure 13. Information flow for on-the-loop experiments of the wind identification system. The blocks show the multi-platform interfaces that allowed the validation tests.

The platform and the airspeed sensor in which the experiments were performed is shown in fig. 14a.

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

(a) (b)

Figure 14. Sensor and UAV platform utilized in experiments. (a) UAV SkyWalker X8 with carbon fiber frame equipped with a 12x6 prop with 2x20g servomotor; (b) Digital airspeed sensor utilized in experiments which contains a 4525DO sensor which enables a resolution of 0.84Pa [28].

Table 6 presents the characteristics of the platform shown in fig. 14a.

Table 6. Selected Trimming Parameters

Wing Span	2122 mm
Wing Area	80 dm^2
Max Payload	2 kg
Center of Gravity	435 mm away from nose

The UAV was equipped with an APM2.6 autopilot with the airspeed sensor illustrated in fig. 14b.

4.4. Software-In-The-Loop Experiments Results

The information of the flights performed in the Seville Metropolitan Area (Brenes)is shown in table 7.

Table 7. Experiments information.

	Flight 1	Flight 2	Flight 3	Flight 4
Duration	521 s	315 s	631 s	749 s
Distance Traveled	5.1 km	3.7 km	6.3 km	7.4 km
Maximum Altitude	179 m	125 m	134 m	146 m

Note that these simulations intend to prove that the difference between the estimated wind speed and the predicted wind speed is bounded or tends to zero within the specific prediction. A full verification stage with a ground-truth wind sensor will be performed and described in the next part of this research.

Figure 15 depicts the trajectories of the flights utilized in the scenarios described in table 7. The first flight has not a particular pattern and performs 8 maneuvers at different altitudes. The other three flights consisted on takeoff, several spirals at a certain altitude and then the descent and landing.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2016 doi:10.20944/preprints201611.0002.v1

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

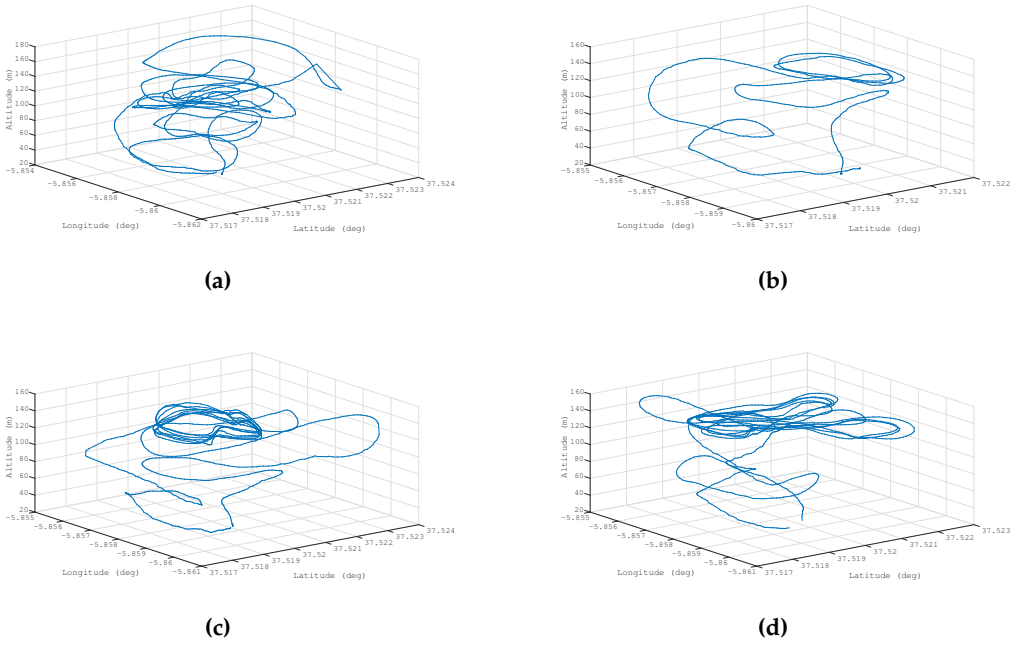


Figure 15. UAV Trajectory for validation of the wind information system. (a) shows a medium altitude with few spirals, (b) shows the shortest flight, (c) shows a flight with spirals performed at an altitude of 120 m and (d) shows a flight with wide spirals at an altitude of 120 m.

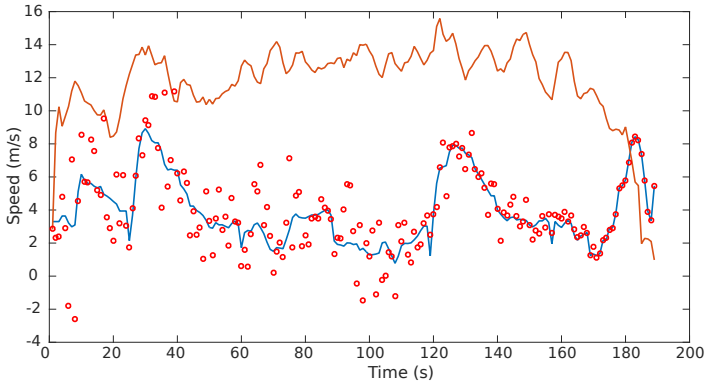


Figure 16. Wind speed estimation (blue line), wind speed prediction (red dots) and airspeed (orange line) generated throughout the flight shown in fig. 15a.

In fig. 16 the red dots indicate the predicted wind speed prediction within an time window of 5 seconds average. The blue line indicate the estimations which were obtained with the direct computation method presented in [8,9]. During the considered flight, due to the continuous changes in wind speed over time, a continuous gust alarm was generated almost at the beginning of the flight. This is confirmed with the distribution of the wind speeds which is similar to the one shown in fig. 10c.

In the second experiment (see fig. 17) a wind speed/altitude plot can be observed. there is a jump in the estimation as UAV reaches its maximum altitude and in lower altitudes th airspeed tends to increase. This was detected as a negative gust that sudden reduces the wind speed which reached up to $9\,\mathrm{m\,s^{-1}}$.

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

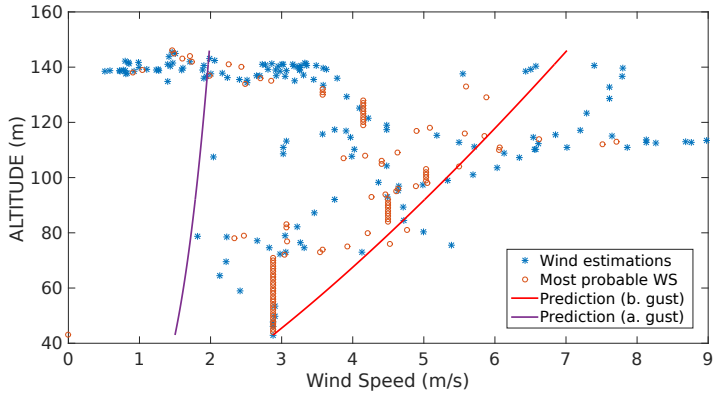


Figure 17. Estimation, most probable wind speeds and last wind prediction generated throughout the flight shown in fig. 15b.

Note that the red and purple line indicates two prediction lines that were generated due to the presence of a discrete gust characterized with a sudden reduction in wind speed at an altitude of 140m.

A similar behavior is observed in both fig. 18 and in fig. 19, in which the wind speed tends to increase as the altitude grows. The high concentration of estimations of the wind speed between 120m and 140m altitude is due to the spirals and the white noise generated due to the sensors limits.

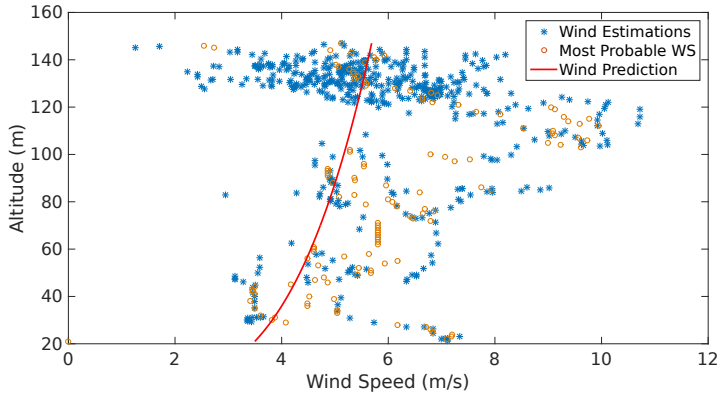


Figure 18. Estimation, most probable wind speeds and last wind prediction generated throughout the flight illustrated in fig. 15c.

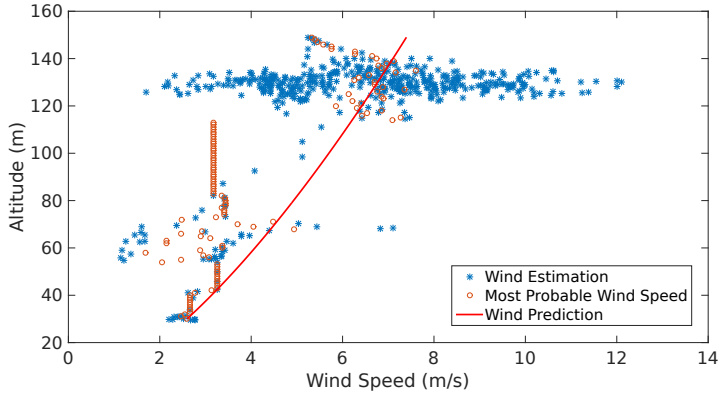


Figure 19. Estimation, most probable wind speeds and last wind prediction generated throughout the flight depicted in fig. 15d.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2016 doi:10.20944/preprints201611.0002.v1

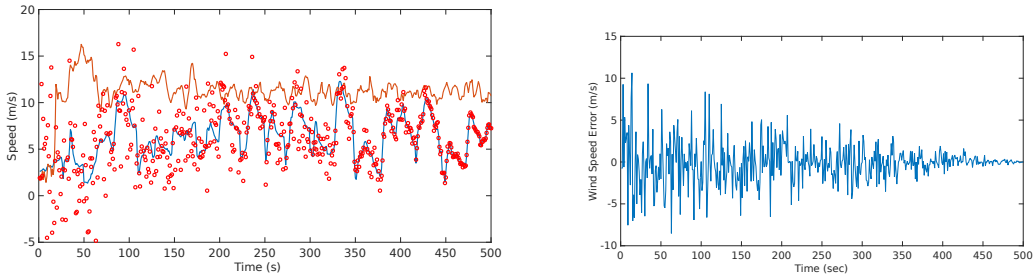
Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

The average Weibull shaping parameter, κ, Weibull scaling parameter, ν, and the calculated Prandtl coefficient, ξ, are shown in table 8.

Table 8. Scenarios outputs (single run)

Scenario	$\mu(\kappa)$	$\mu(\nu)$	$\mu(\xi)$
1	$4.24m/s$	1.9825	0.6628
2	$1.1579, 3.1425m/s$	0.4531, 1.6671	0.5314, 0.6628
4	$3.1425m/s$	0.8349	0.3124
4	$5.7425m/s$	0.9623	0.7614

Figure 20 shows the difference comparison between the estimations and the predictions of the wind speed magnitude. It is important to consider that it is the difference, therefore, it cannot be considered as an absolute error. However it aims to prove that this difference is bounded since it considers local measurements.



(a) **(b)**

Figure 20. Predicted (red circles) vs. Estimated wind speed (blue line): (a) shows the difference between the estimation and the prediction of the wind speeds with the airspeed (orange) as a reference; (b) shows the actual difference between these two quantities which appears to be bounded and shows a normal behavior.

The difference analysis between all the flights together showing the mean $\mu(W_e)$ and the standard deviation $\sigma(W_e)$ are shown in table 9:

Table 9. Mean and standard deviation of difference between the wind speed estimations and the wind speed predictions

Flight	$\mu(W_e)$	$\sigma(W_e)$
1	1.985m/s	2.54m/s
2	1.197m/s	1.21m/s
3	0.932m/s	0.74m/s
4	0.854m/s	0.27m/s

5. Results Discussion

In the simulation results, the effects of the wind features (continuous and discrete gusts and shear) are shown in fig. 9. It is observed that a single or multiple features can affect the trajectory without any sort of drifting compensation. In addition, fig. 10 shows how the different features in a wind speed/altitude plot. The system is able to identify this feature from the summed effects plot showed at fig. 10d. However at this stage noise is not considered and it can have a significant impact to the plot, so the estimation process has to be accurate enough since it is the only input to the wind identification

Peer-reviewed version available at Sensors 2017, 17, 8; doi:10.3390/s17010008

system. The case shown in fig. 10d is analyzed in detail since the effects of each feature separately were analyzed in [8,9].

The results plotted in fig. 11 show the behavior of the estimation and prediction process. The estimation process considered a slight Gaussian noise which is typical for airspeed sensors [2]. The error of the estimation process is bounded as observed in fig. 12 which is indeed the expected behavior and is consistent with te result presented in [2,8,9]. On the other hand, the prediction shows a different behavior. At the beginning and up to 70 seconds there is a considerable dispersion of the prediction. This is due to the assumption that only a shear feature is present since the beginning. Then, the system starts identifying other features and at 40 seconds a rapid change is observed which triggers a discrete gust alarm. Up to that point the system starts converging and the predictions with a variable window start happening. Since there is a continuous gust during the entire scenario, the system utilizes Gaussian regression to start predicting the behavior of the plot. Even though there are rapid changes at some parts, the identification of the discrete gust minimizes the effect in the Gaussian regression. The prediction error shows a gradual decrease up to the point that it follows a similar behavior than the estimation which leads to the conclusion that the prediction error tends to be bounded as more data is fed to the system.

The software-in-the-loop testing shows four scenarios with actual airspeed measurements. Figure 15 shows different paths that depict different maneuvers at variable altitudes. These scenarios are very helpful to comprehend how the noise of the airspeed measurements affect the estimation. However these results have to be treated carefully since there is no ground-truth and validate the functionality of the system only.

In the scenario shown in fig. 15a the results of the prediction and estimation process show a very chaotic behavior with estimations that vary in short altitudes (see fig. 16). The system triggers a continuous gust alarm and the estimation process employs Gaussian Regression, nevertheless the system generates shear predictions based on the identification of the most probable wind speeds which show less dispersed tendency due to the distribution of the measurements among the different altitudes that follow the Weibull distribution.

The scenario shown in fig. 15b shows two shear features that are identified due to a sudden change of wind speed that occurs at 40 m. Since a discrete gust was detected the system tries to identify this two features. It is observed that there is are three points that the most probable wind speeds looks constant. This is due to the lack of measurements possibly by a communication error between the system and the ground station.

The remaining two flights (see figs. 15c and 15d) show a very similar behavior at high altitude. Nevertheless in the fourth flight there is a lack of airspeed measurements and the most probable wind speed is assumed to be constant. However the actual prediction (red line) shows what in reality the airspeed has to behave. Since a lot of spiral maneuvers were performed, the system has a vast amount of estimations over an altitude of 120m, however, there is a substantial dispersion that follows the Weibull distribution allowing the generation of coherent most probable wind speeds and to treat this measurements as part of a wind shear feature.

For the last scenario a comparison between the estimation and the prediction can be observed in fig. 20. The error between the predictions and estimation is bounded since the very beginning, mainly to the absence of continuous gust features. This results can be confirmed with the study of the deviation of this difference that is shown in table 9.

6. Conclusions and Future Work

This paper presents the integration of a a wind identification system using small UAS. It describes the high and low level architecture and provides a initial validation with simulations and software-in-the-loop testing.

The system architecture integrates different components at various levels. In terms of hardware, the proposed system uses COTS components which help on cost efficiency without the sacrifice of

functionality or reliability mainly due to the characteristics of the system and the current state-of-the-art. On the other hand, the software has a deep integration at the system and component level due to the development of a communication handler that manages the information exchange between the different modules of the system. The main advantages of this communication scheme are the separation between different functional modules, which ensures the upgradeability and the module dependencies. Also the possibility can be easily expanded by adding other functional modules, such as trajectory generation/optimization. Another factor considered in the design was the possibility of asynchronous communication between blocks. This is an important requirement due to the possible variation on the processing time for different modules regardless if the variation is generated at a software or hardware level.

Each software module of the wind identification system is described at high and low level. The implemented methods in these modules have been extensively studied in previous work [8,9].The main goal was to move from a multi-platform system to a single-platform with clear interactions between other systems at software and hardware level.

In terms of the wind speed calculation and wind speed prediction validation, the system was tested with both simulations and software-in-the-loop. In the simulations, results indicate that the wind identification system is capable of identifying and eventually converge to the actual wind within variable time windows. However, the accuracy varies according to the identification of other features. Therefore, as clearer features are identified, the convergence time is reduced together with the error magnitude and dispersion. In the software-in-the-loop testing the system exhibits dispersion on the wind estimates which is mainly attributable to the noise from the airspeed sensor. The system estimates were Weibull-distributed in altitudes on which the aircraft remains for longer periods and presented inaccurate predictions at altitudes in which the aircraft has low or null density of measurements. The presented results lead to the conclusion that the system fulfills the design requirements and provides the identification of separated wind features which could be really useful for trajectory planning and optimization. The novelty of the system relies in two main aspects: first the architecture with a upgradeable system with minimum module dependency and secondly the information that the system generates since the identification of separated wind field features could easily be used for efficient trajectory planning, for instance in dynamic soaring.

Future work includes more validation and verification of the system at various levels as well as on-board/real-time testing of the system. This paper intends to present a detailed description and the initial stages of validation and verification of the system. The full validation and verification, including hardware-in-the-loop and on-board testing activities and the integration of the mapping feature will be subject of a further publication which will provide results at system and component level in terms of accuracy, reliability and a detail analysis of the computational cost of the different methods. In addition, upcoming work includes the integration of a trajectory generation module and the generation of control commands to follow the wind-efficient trajectories which ultimately is derived from the objective of increasing substantially the flight duration in a given mission.

Abbreviations

The following abbreviations are used in this manuscript:

ADC: Analog to Digital converter
AOA: Angle of Attack
APM: Autopilot Module
COTS: Commercial-Off-The-Shelf
CPU: Central Processing Unit
DB: Database
EKF: Extended Kalman Filter

GEV: Generalized Extreme Value
GNSS: Global Navigation Satellite System
GP: Gaussian Proccess
IMU: Inertial Measurement Unit
LPGL: GNU Lesser General Public License
PID: Proportional, Integral, Derivative
RAM: Random Access Memory
SQL: Structured Query Language
TN: Truncated Normal (Distribution)
UAV: Unmanned Aerial Vehicle
UDP: User Datagram Protocol
UKF: Unscented Kalman Filter

Acknowledgments: This work has been supported by the MarineUAS project, funded by the European Comission under the H2020 Programme (MSCA-ITN-2014-642153) and the AEROMAIN Project (DPI2014-5983-C2-1-R), funded by the Science and Innovation Ministery of the Spanish Government.

Author Contributions: Leopoldo Rodriguez have designed the overall system, Jose Antonio Cobano has contributed in the architecture conception and the experiments, Anibal Ollero revised and edit the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Bibliography

1. Sheherao Biradar, A. Wind Estimation and Effects of Wind on Waypoint Navigation of UAVs. Master thesis, Arizona State University, 2014.
2. Langelaan, J.W.; Alley, N.; Neidhoefer, J. Wind Field Estimation for Small Unmanned Aerial Vehicles. *Journal of Guidance Control and Dynamics* **2011**, *Vol. 34*, 109–117.
3. Johansen, T.; Crisofaro, A.; Sorensen, K.; Fossen, T. On estimation of wind velocity, angle-ofattack and sideslip angle of small UAVs using standard sensors. *International Conference on Unmanned Aircraft Systems* **2015**, *Denver, CO*, 510–519.
4. Lawrance, N.; Sukkarieh, S. Simultaneous Exploration and Exploitation of a Wind Field for a Small Gliding UAV. *AIAA Guidance, Navigation and Control Conference* **2010**, *Toronto, Ontario, Canada*.
5. Larrabee, T.; Chao, H.; Gu, Y.; Napolitano, M. Wind field estimation in UAV formation flight. *American Control Conference* **2014**, *Portland, OR*, 5408–5413.
6. Neumann, P.; Bartholmai, M. Real-time wind estimation on a micro unmanned aerial vehicle using its inertial measurement unit. *Sensors and Actuators: A. Physical* **2015**, *235*, 300–310.
7. Condomines, J.F.; Bronz, M.; Erdely, J.F. Experimental Wind Field Estimation and Aircraft Identification. *International Micro Air Vehicles Conference and Flight Competition* **2015**, *Aachen, Germany*.
8. Rodriguez, L.; Cobano, J.; Ollero, A. Wind Characterization and Mapping Using Fixed-Wing Small Unmanned Aerial Systems. *International Conference on Unmanned Aircraft Systems* **June 7-13, 2016**, *Arlington, VA, USA*, 178–184.
9. Rodriguez, L.; Cobano, J.; Ollero, A. Wind Fiel Estimation and Identification Having Shear Wind and Discrete Gust Features with a Small UAS. *IEEE/RSJ International Conference on Intelligent Robots and Systems* **2016**, *Daejon,Korea*.
10. Cobano, J.A.; Alejo, D.; Vera, S.; Heredia, G.; Sukkarieh, S.; Ollero, A., Distributed Thermal Identification and Exploitation for Multiple Soaring UAVs. In *Human Behavior Understanding in Networked Sensing: Theory and Applications of Networks of Sensors*; Spagnolo, P.; Mazzeo, L.P.; Distante, C., Eds.; Springer International Publishing: Cham, 2014; pp. 359–378.
11. Cutler, M.J.; McLain, T.W.; Beard, R.W.; Capozzi, B. Energy Harvesting and Mission Effectiveness for Small Unmanned Aircraft. *AIAA Guidance, Navigation and Control Conference* **2010**, *Toronto, Ontario, Canada*.
12. Chakrabarty, A.; Langelaan, J.W. Flight Path Planning for UAV Atmospheric Energy Harvesting Using Heuristic Search. *AIAA Guidance, Navigation and Control Conference* **2010**, *Toronto, Ontario, Canada*.

13. Montella, C.; Spletzer, J.R. Reinforcement Learning for Autonomous Dynamic Soaring in Shear Winds. *IEEE/RSJ International Conference on Intelligent Robots and Systems* **2014**, *Chicago, IL, USA*.
14. Bird, J.; Langelaan, J.; Montella, C.; SPletzer, J.; Grenestedt, J. Closing the Loop in Dynamic Soaring. *AIAA Guidance, Navigation, and Control Conference* **2014**, *National Harbor, MD, USA*.
15. U.S. Department of Defense. *Military Specification, Flying Qualities of Piloted Airplanes MIL-F-8785C*, 1980.
16. U.S. Department of Defense. *Handbook, Flying Qualities of Piloted Airplanes MIL-F-8785C*, 1997.
17. The MathWorks, Inc. *MATLAB Reference Pages, Dryden Wind Turbulence Model (Continuous)*, 2010.
18. Gao, C. Autonomous soaring and surveillance in wind fields with an unmanned aerial vehicle. PhD thesis, University of Toronto, 2015.
19. Park, C.; Huang, J.; Ding, Y. Domain decomposition for fast Gaussian process regression. *Journal of Machine Learning Research* **2011**, *12*, 1697–1728.
20. Lerch, S.; Thorarinsdottir, T.L. Comparison of nonhomogeneous regression models for probabilistic wind speed forecasting. *Tellus A* **2013**, *65, 21206*.
21. 3DR. *Pixhawk Autopilot, Quick Start Guide*, 2014, [https://goo.gl/lhi0jx].
22. Hardkernel co., LTD. *Odroid Platforms, ODROID-C2*, 2013, [http://goo.gl/8nQVKO].
23. Control, Q. *MAVLink Micro Air Vehicle Communication Protocol*, 2016, [http://qgroundcontrol.org/mavlink/start].
24. Meier, L. *MAVCONN- MICRO AIR VEHICLE MIDDLEWARE*, 2014, [http://goo.gl/P82kbG].
25. Pixhawk, Computer Vision on Autonomous Aerial Robots. *MAVCONN- MICRO AIR VEHICLE MIDDLEWARE*, 2014, [http://goo.gl/P82kbG].
26. Passner, J.; Knapp, I.; Ding, Y. USING WRF-ARW DATA TO FORECAST TURBULENCE AT SMALL SCALES. *13th Conf. on Aviation, Range, and Aerospace Meteorology* **2007**, *13*, 3.1.
27. Team, A.D. *http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html*, 2016, [https://goo.gl/thkBTu].
28. jDrones. *Digital airspeed sensor*, 2016, [https://goo.gl/1T73k2].