Bachelor's Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Small UAV flight control system

Wind estimation system

Martin Mondek
Systems and control

June 2015
Supervisor: Martin Hromčík

Acknowledgement / Declaration

I would like to thank Martin Hromčík for his guidance during work on this thesis and Pavel Hospodář for his help during wind tunnel measurements.

This research was supported by the Czech Science Foundation (GACR) under contract No. GA13-06894S.

This work was created within project ”Velká infrastruktura Aerodynamických tunelů”. This project is part of program ”LM2011016/Aerodynamické tunely” with financial support from Ministry of Education, Sport and Youth Czech Republic within ”Projekty velkých infrastruktur pro VaVaI” activities.

I declare on word of honour that I have created my Bachelor’s thesis independently and I have used only sources (literature, projects, applications) mentioned in the appendix of this publication.

In Prague, 22. 5. 2015

. . . .

Abstrakt / Abstract

Cílem této práce je vývoj systému pro odhadování větru. Tento systém je vyvíjen na základě existujícího řídícího systému vyvinutého studenty na Katedře řídící techiky Fakulty elektrotechnické Českého vysokého učení technického v Praze. V práci je diskutována analýza šumu měření a dopadu těchto chyb na funkčnost algoritmu. Dále funkčnost vyvíjeného algoritmu je simulacemi ověřena na realistických fyzikálních modelech letadla. V krátkosti je dále představeno proběhlé vylepšení existujících hardwarových a softwarových částí. Na závěr je prezentován proces a výsledky kalibrace hardwaru obsluhujícího víceotvorovou pitotovu trubici.

Klíčová slova: Algoritmus na odhad větru, bezpilotní prostředky, UAV letoun, řídící systém letadla, víceotvorová pitotova trubice a kalibrace.

Překlad titulu: Návrh a realizace modulárního řídicího systému pro malé UAV (Odhad větru)

This thesis describes development and achieved results of an on–line on–board wind estimation system. The system is planned to augment functionality and improve performance of existing small UAV flight control system units, developed within students activities at the Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague. Measurement error analysis and performance simulations are researched upon realistic flight mechanics models. Improvement of related hardware and software components is also introduced and discussed in brief. Finally the process and results of related multi–hole pitots tube hardware calibration is provided.

Keywords: Wind prediction algorithm, unmanned aerial vehicles, UAV aircraft, flight control board, multi–hole pitots tube.

Contents

Tables / Figures

Tables

Figures

Chapter 1
Introduction

The usage of unmanned aerial vehicles (UAVs) is very broad these days. It varies from simple children's toys controlled by smartphones or more advanced RC sets to professional solutions used by government, armies or medical teams. The different types of UAVs are used for example to monitor and patrol the state borders or guarded buildings, track ground object or perform military missions. The benefit of UAVs can be used during natural disasters too — for example monitoring and searching people in danger during floods or earthquake or very quick distribution of medical supplies or equipment to unreachable areas. In meteorology UAVs can measure the air pressure and humidity or the air pollution at higher altitudes.

Figure 1.1. Modern delivery systems. a) Google delivery system, b) Amazon Prime Air.

These days the popularity of small UAVs is spread between common people by big internet companies such as Amazon with their system Amazon Prime Air (Fig. 1.1b, [1]) or Google with drone delivery system (Fig. 1.1a, [2]). NASA recently introduced special concept of drone with convertible wings.

Figure 1.2. New concept of convertible drone by NASA.

The control systems of these drones are essentially similar to small hobbyist systems such as Ardupilot or Pixhawk. They usually provide standard control and sensor equipment for easy remote control of the UAV. In these systems even some advanced autopilot control structures can be implemented. With different on–board software these UAVs are capable to work in fully autonomous mode.

The motivation to the main theme of this thesis — **the wind estimation system** — came from Jaroslav Halgašík's work [3]. He developed modular flight control system. During implementation of this system in the airship (Fig. 1.3) the control algorithms were not working properly during the wind disturbances.

Figure 1.3. Airship — one of possible implementations of wind estimation system.

Another input came from AgentFly group [4] developing multiagent flight formations at Faculty of Electrical Engineering, CTU. Their algorithms faced the same problem — during external wind disturbances the flying agents were not performing all tasks correctly.

Figure 1.4. Cross wind landing.

Finally if we look at the problem of wind disturbances in larger scale, we can find some dangerous situations. During difficult weather conditions, for example when the aircraft is exposed to big cross or back wind, the passengers, cabin crew or cargo might be in great danger (Fig. 1.4). The aircraft can stall or in an extreme case crash. The information about the speed and direction of the wind can lead to the improvement of the aircraft control systems and it can be very valuable information for pilots too.

Chapter 2
Goals of this project

Main objective of this thesis is to develop wind speed and direction computational algorithm and implement this computation in VZLU/CVUT aircraft using appropriate hardware solutions. This task can be divided into three different parts — the algorithm derivation, the software implementation of this algorithm and related hardware solutions. All parts have several subtask as shown in the following list.

- **Wind prediction algorithm**
  - Develop computational algorithm for wind prediction
  - Perform several simulations using realistic aircraft model
  - Research error measurement sensitivity of developed algorithm
  - Perform field tests for verification of algorithm
  - Analyse measured data and review function of the algoritm
- **Software implementation**
  - Implement developed computational algorithm in C programming language
  - Extend existing on–board and ground station software
  - Implement all functionalities of newly added hardware
- **On-board hardware**
  - Use flight control board developed at DCE FEE CTU [3]
  - Complete related hardware for multi–hole pitots tube measurements [5]
  - Perform multi–hole pitots tube calibration in wind tunnel
  - Add external magnetometer and perform relevant measurement calibration

Several other tasks were performed during work on this thesis. This additional tasks are listed as follows.

- **Additional activities**
  - Review and update implementation of used electronics, sensors and on–board flight control system in the VZLU/CVUT aircraft
  - Implement multi–hole pitots tube and differential pressure board to the VZLU/CVUT aircraft
  - Prepare VZLU/CVUT aircraft for experimental flight tests
  - Provide technical support during flight performance measurement

All listed detailed tasks are in full agreement with the official assignment of this thesis — see Appendix A.

Chapter 3
Theoretical background

3.1 Reference frames

During autopilot or computational algorithm design three reference frames are required. The transformation between these reference systems shall be utilized for expression of one vector in different reference frames. This transformation provides background for mathematical operations with vectors.

3.1.1 Ground reference frame

Due to final implementation of wind prediction algorithm inside small UAVs, ”North–East–Down” (NED) reference system is used to simplify all equations of motions, computations and algorithms. This system uses linear X, Y and Z coordinates to provide position of objects with respect to the origin of the coordinate system.

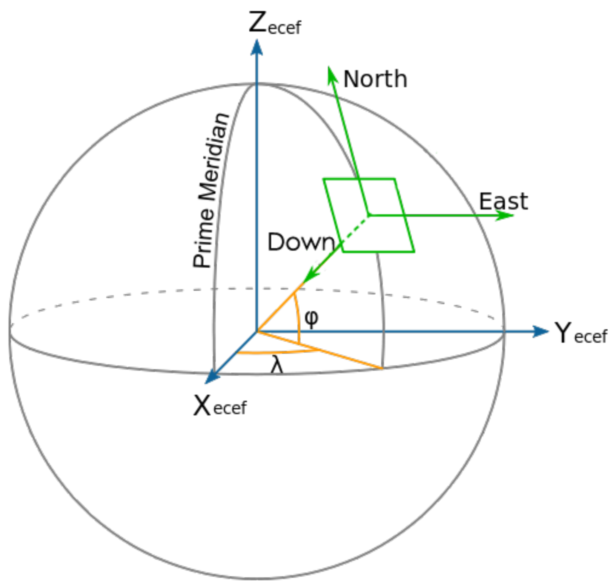


Figure 3.1. Relationship between Earth fixed (blue), Geodetic (orange) and NED (green) systems.

During sensor output implementation big attention must be given to sensor values. For example coordinate system provided by GPS sensor is in geodetic reference system and must be correctly converted to ”flat Earth” (NED) reference system.

3.1.2 Body reference frame

During different flight performances the plane is not usually precisely aligned with NED reference system. Another reference frame is required — the body reference frame. The rotation of the aircraft is defined by three angles called Euler angles ψ, θ and ϕ (heading, elevation and bank angle) using Tait–Bryan convention.

In body coordinate frame X-axis points towards the front of the aircraft (or in the flight direction), the Y-axis points to towards the right wing and Z-axis points down. The X and Z axes lies in symmetry plane of the aircraft.

The heading (yaw) angle is the direction of the aircraft's X-axis in XY plane of NED reference system. Similarly the elevation (pitch) is the angle in XZ plane and bank (roll) is the angle in YZ plane of NED reference system.

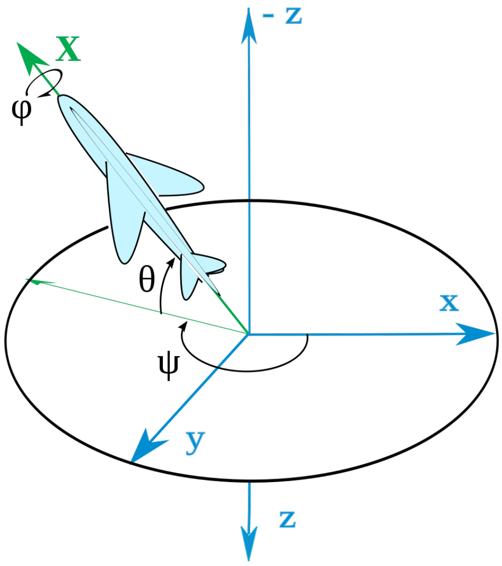


Figure 3.2. Euler angles (yaw, pitch, roll) and body reference frame.

3.1.3 Aerodynamic reference frame

Last used reference system is the aerodynamic reference system (or so–called wind-axis or stability reference frame). Unlike in the body coordinate system, in this reference system X-axis points directly to the direction of relative wind (is parallel to speed vector of air flow around plane). The Z-axis remains unrotated.

This reference system introduces two important angles — angle–of–attack α and sideslip angle β. See Eq. (1), where $\mathbf{V} = [\,u \quad v \quad w\,]^{\mathsf{T}}$ is vector of airspeed.

$$\alpha = \tan^{-1}\left(\frac{w}{u}\right) \qquad \beta = \sin^{-1}\left(\frac{u}{|\mathbf{V}|}\right) \tag{1}$$

The angle–of–attack is the angle between velocity vector and the XZ plane. It is positive when relative wind is blowing from below the plane. The sideslip angle is the angle between velocity vector and the plane of symmetry (XY plane). This angle is positive when relative wind is blowing from right of the plane of symmetry. Both angles are shown in Fig. 3.3.

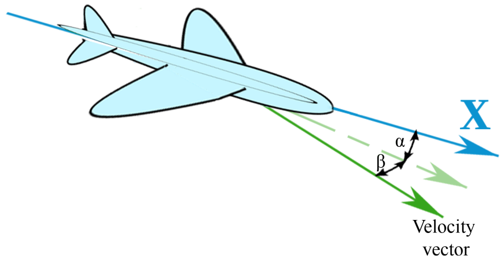


Figure 3.3. Definition of angle–of–attack and sideslip angle.

3.2 Transformation between reference systems

The usage of three different coordination systems brings requirement of transformation between these systems. Therefore rotational matrices are used to fulfil this need.

- **Body–to–Aerodynamic transformation**

Each complex rotation around different axes can be divided into separate rotations around one axis at one time. Thus during rotation between body and aerodynamic reference frame we rotate the system around the Y-axis through the angle–of–attack α first and then around the Z-axis through the sideslip angle β.

(2)

$$\mathbf{R}_\alpha = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \qquad \mathbf{R}_\beta = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Combining rotational matrices $\mathbf{R}_\alpha$ and $\mathbf{R}_\beta$ we obtain rotational matrix for vector transformation from the body reference frame to the aerodynamic reference frame. This transformation is shown in Equation (4).

(3)

$$\mathbf{R}_\mathbf{B}^\mathbf{W} = \mathbf{R}_\beta \mathbf{R}_\alpha = \begin{bmatrix} \cos\alpha\cos\beta & \sin\beta & \sin\alpha\cos\beta \\ -\cos\alpha\sin\beta & \cos\beta & -\sin\alpha\sin\beta \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

(4)

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{wind} = \mathbf{R}_\mathbf{B}^\mathbf{W} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{body}$$

The resulting matrix $\mathbf{R}_\mathbf{B}^\mathbf{W}$ is orthogonal matrix. For the inverse operation (transformation from the wind to the body reference frame) the inverse matrix $\mathbf{R}_\mathbf{W}^\mathbf{B}$ can be used which in fact is only transposition of the original matrix $\left(\mathbf{R}_\mathbf{B}^\mathbf{W}\right)^T$.

- **Ground–to–Body transformation**

Similarly, the Ground–to–Body transformation can be divided into separate rotations as the Body–to–Aerodynamic transformation. We obtain final coordination system from the ground reference system using rotation around the Z-axis by the

(5)

$$\mathbf{R}_\psi = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(6)

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

(7)

$$\mathbf{R}_\psi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}$$

(8)

$$\mathbf{R}_\mathbf{I}^\mathbf{B} = \mathbf{R}_\phi \mathbf{R}_\theta \mathbf{R}_\psi = \begin{bmatrix} \cos(\psi)\cos(\theta) & \sin(\psi)\cos(\theta) & -\sin(\theta) \\ \cos(\psi)\sin(\theta)\sin(\phi) - \sin(\psi)\cos(\phi) & \sin(\psi)\sin(\theta)\sin(\phi) + \cos(\psi)\cos(\phi) & \cos(\theta)\sin(\phi) \\ \cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi) & \sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix}$$

heading angle (yaw, ψ), then applying rotation around the Y-axis of newly created reference system through the elevation angle (pitch, θ) and finally rotating newly created system around the X-axis by the bank angle (roll, ϕ). These rotations can be expressed by rotational matrices $\mathbf{R}_\psi$, $\mathbf{R}_\theta$ and $\mathbf{R}_\phi$ (Eq. (5), (6), (7)) and complex Ground–to–Body transformation by matrix $\mathbf{R}_\mathbf{I}^\mathbf{B}$ (Eq. (8)).

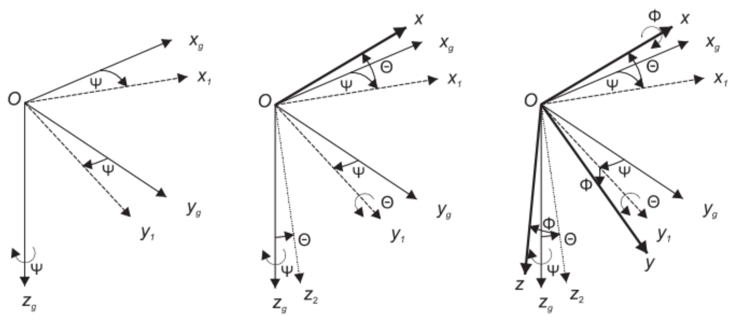


Figure 3.4. Individual steps of Ground–to–Body transformation.

The inverse transformation can be performed using matrix $\mathbf{R}_\mathbf{B}^\mathbf{I} = \left(\mathbf{R}_\mathbf{I}^\mathbf{B}\right)^T$.

3.3 Physics of the plane

The complex equations of motion include variance in plane specifications (e.g. weight of the aircraft), dynamics of movable parts of plane or elasticity of the aircraft. For our purposes these very complex equations are simplified.

We assume that our plane is rigid body with stable weight (no fuel consumption), the gravitational force is constant and the NED reference frame is used as the main coordination system.

Derivation of all equations of motion can be found in various literature, for example in [6]. For our purposes, only collection of results is presented.

3.3.1 Force equations

Basic motion of the aircraft can be expressed by Newton's second law:

(9)
$$\mathbf{F} = m\mathbf{a_c} = m\frac{\mathrm{d}}{\mathrm{dt}}(\mathbf{V})$$

where $\mathbf{F} = [F_x \quad F_y \quad F_z]^T$ is force, a_c is the acceleration of the centre of gravity of aircraft, $\mathbf{V} = [u \quad v \quad w]^T$ is vector of the flight speed, $\mathbf{M} = [M_x \quad M_y \quad M_z]^T$ is moment acting on the aircraft, $\boldsymbol{\omega} = [p \quad q \quad r]^T$ is the vector of angular velocities and $\mathbf{I}$ is the inertia matrix of the plane.

This simple form of Newton's second law is valid only if it is applied to the plane's center of gravity. Additionally these equations must be transformed into body reference system.

(10)
$$\mathbf{F_B} = T_{BI}\mathbf{F} = m\frac{\mathrm{d}}{\mathrm{dt}}(\mathbf{V_B}) + m(\Omega_B)\mathbf{v_B}$$

After force and motion variables substitution we obtain three final force equations in the body reference system. These equations are defined as follows

(11)
$$\dot{u} = \frac{1}{m}(X + T\cos(e_T)) - g\sin(\theta) + rv - qw$$
$$\dot{v} = \frac{1}{m}(Y) + g\sin(\phi)\cos(\theta) + pw - ru$$
$$\dot{w} = \frac{1}{m}(Z + T\sin(e_T)) + g\cos(\phi)\cos(\theta) + qu - pv$$

where $\mathbf{v_B} = [u \quad v \quad w]^T$ is the aircraft speed and $\dot{\mathbf{v}}_\mathbf{B} = [\dot{u} \quad \dot{v} \quad \dot{w}]^T$ is the aircraft acceleration in body reference system, $(\mathbf{F_A})_\mathbf{B} = [X \quad Y \quad Z]^T$ is the vector of aerodynamic forces, $\omega_\mathbf{B} = [p \quad q \quad r]^T$ are angular velocity components, $\mathbf{T_B} = [T\cos(e_T) \quad 0 \quad T\sin(e_T)]^T$ is vector of the thrust force and e_T is the thrust angle. This angle is usually zero due to simplification of equation of motion and design of the aircraft.

3.3.2 Moment equations

(12)
$$\mathbf{M} = \frac{\mathrm{d}}{\mathrm{dt}}\mathbf{h}$$

(13)

$$\mathbf{M_B} = I_B \dot{\omega}_\mathbf{B} + \Omega_B I_B \omega_\mathbf{B}$$

(14)

$$I_B = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} \quad I'_B = \begin{bmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{xz} & 0 & I_{zz} \end{bmatrix}$$

(15)

$$\dot{p} = \frac{I_{zz}}{I_D}[L + I_{xz}pq - (I_{zz} - I_{yy})qr] + \frac{I_{xz}}{I_D}[N - I_{xz}qr - (I_{yy} - I_{xx})pq]$$

$$\dot{q} = \frac{1}{I_{yy}}[M + M_T - (I_{xx} - I_{zz})pr - I_{xz}(p^2 - r^2)]$$

$$\dot{r} = \frac{I_{xz}}{I_D}[L + I_{xz}pq - (I_{zz} - I_{yy})qr] + \frac{I_{xx}}{I_D}[N - I_{xz}qr - (I_{yy} - I_{xx})pq]$$

From equation (12) — after transformation into the body reference system using transformational matrix T_{BI} derived previously and some additional manipulation — final moment equations are given by

Next we assume that our plane is symmetric so the cross-products involving y–axis become zero in the inertia matrix I_B as given in Eq. (14)b.

After some manipulation, expansion and substitution with variables we can present final moment equations of the plane as follows

where $I_D = I_{xx}I_{zz} - I_{xz}^2$ and $\mathbf{M_B} = [L \quad M + M_T \quad N]^T$ is the moment vector.

(16)

$$I''_B = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

In some cases the problem can be expressed in principal axes (all cross–products in the inertia matrix are zero, see Eq. (16)). Then the moment equations simplifies into

(17)

$$\dot{p} = \frac{L - (I_{zp} - I_{yp})qr}{I_{xp}}$$

$$\dot{q} = \frac{M + M_T - (I_{xp} - I_{zp})pr}{I_{yp}}$$

$$\dot{r} = \frac{N - (I_{yp} - I_{xp})pq}{I_{zp}}$$

3.3.3 Kinematic equations

In this section kinematic equations (or body–axis orientation equations) are given. All three scalar equations are

(18)

$$\dot{\phi} = p + (q\sin(\phi) + r\cos(\phi))\tan(\theta)$$
$$\dot{\theta} = q\cos(\phi) - r\sin(\phi)$$
$$\dot{\psi} = (q\sin(\phi) + r\cos(\phi))\sin((sec)\theta)$$

3.3.4 Navigation equations

Finally navigation equations in body reference system are given in this section. Simple vector equation Eq. (19) is given after simple expansion in Eq. (20).

(19)

$$\begin{bmatrix} \dot{x}_E \\ \dot{y}_E \\ \dot{z}_E \end{bmatrix} = T_{IB} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

(20)

$$\dot{x}_E = u(\cos(\theta)\cos(\psi)) + v(\sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi))$$
$$+w(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi))$$
$$\dot{y}_E = u(\cos(\theta)\sin(\psi)) + v(\sin(\phi)\sin(\theta)\sin(\psi) + \cos(\phi)\cos(\psi))$$
$$+w(\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi))$$
$$\dot{h} = -\dot{z}_E = u\sin(\theta) - v\sin(\phi)\cos(\theta) - w\cos(\phi)\cos(\theta)$$

3.4 Simulink model of the aircraft

For the first test implementation of the algorithm and later simulations of the measurement sensitivity the realistic mathematical model was required. In Josef Novák's diploma thesis [7] mathematical model of small Cessna plane (see section 8.1) was introduced. This model was derived from example mathematical models delivered within Simulink plugin called Aerosim Blockset [8]. Unfortunately this plugin is out–of–date and it is no longer supported by authors. Yet this plugin can still be found in several web discussion forums or in various web archives.

All previously listed differential equations of motion are implemented in this model. Because the external wind affects the physics of the plane, external input was required to be implemented according to specification of our algorithm (see Fig. 3.5, golden box). The implementation of the wind effect was composed only from the simple equations listed in section 6.1 and it is hidden inside light blue box representing the model of the UAV. Additional request was the implementation of additional parts of the model and simple control feedback loops such as pitch or roll hold autopilot (light green boxes), pitch and yaw dampers (pink boxes) or some servo models (orange boxes). With this additional features we were able to create several different flight paths to create different environments and scenarios for testing our wind prediction algorithm.

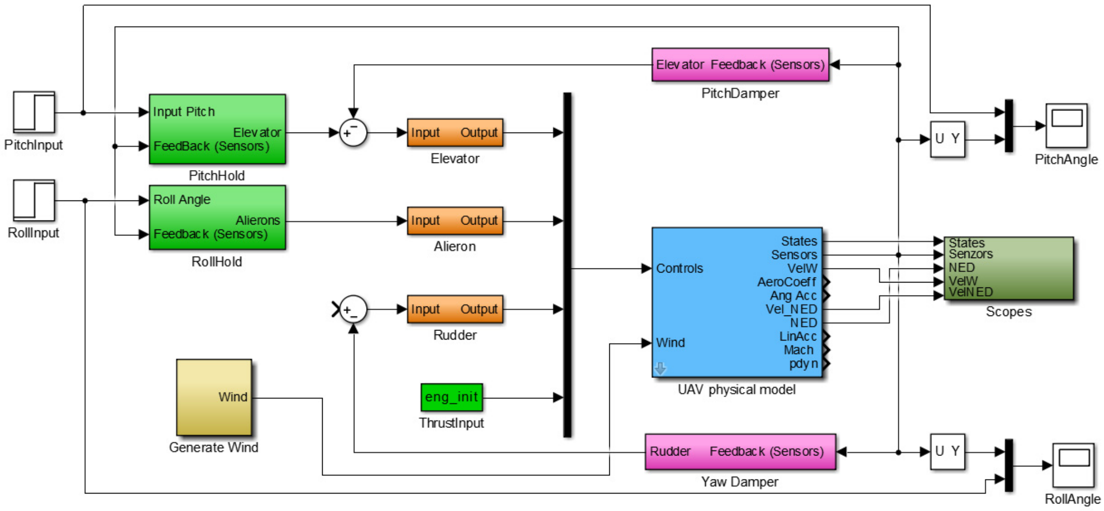


Figure 3.5. Used realistic Simulink model with implemented wind effects (golden block).

Chapter 4
Hardware components

Several hardware components were developed within previous student's activities [3]. Some of these components were additionally adjusted to fit all purposes of this thesis. The complete flight control system consist of two separate parts: the ground station and the on–board flight control unit. Additional computer system or Android mobile phone can be connected to the ground station by Bluetooth to display measurement data or control on–board systems in real time.

4.1 On–board flight control unit

Main on–board control system must be able to perform several tasks. The main task of these systems is to control aircraft propulsion and all control surfaces alongside with managing communication channel with the ground station and respond to the pilot's input. Some additional algorithms and autopilots can be implemented too. More complex tasks and algorithms in civilian area can be applied — for example meteorology measurement system, aerial photography, ground mapping or some research systems. Depending on the final usage, different selection of on–board control system should be considered.

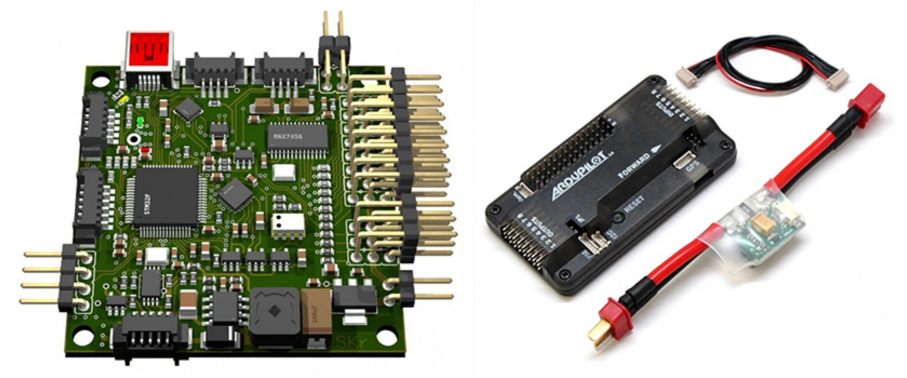


Figure 4.1. Example of aircraft control systems. a) Openpilot, b) Ardupilot.

These days simple electronic control units are very popular between RC aircraft makers. Interested enthusiast are speeding up the development of these control boards for different types of UAVs and causing the price of whole system to drop. Most of these control systems are open-source, such as ArduPilot [9] or Openpilot [10] (See Fig. 4.1).

For educational purposes Jaroslav Halgašík in his thesis [3] developed own control system to fulfil all his needs. Main advantage of his solution (Fig. 4.2) is very small size and reduced weight. Final implementation of the whole control system in VZLU/CVUT was set as part of this thesis (See section 8.2). In this section the main parts of on–board system are described. Improved microcontroller programs and implementation of wind prediction algorithms can be found in Chapter 5.

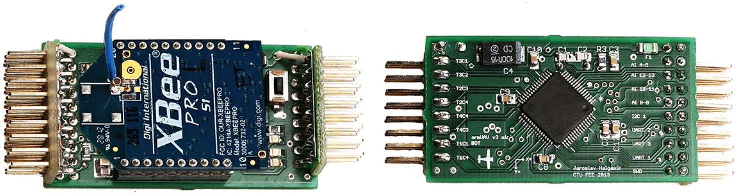


Figure 4.2. Main on–board flight control unit.

4.1.1 Main microcontroller unit

The ”heart” of the on–board flight control system is small and powerful ARM microcontroller (STM32F100RB) from ST Microelectronics companz. This 32-bit RISC core microcontroller operates at maximal 24 MHz frequency and incorporates with embedded memories (Flash 128 Kbytes, SRAM 8 Kbytes). It includes one 12-bit ADC converter and provides support for typical peripherals such as timers for controlling small servos using PWM signal, SPI, I^2C and USART communication bus. More informations about this microcontroller can be found in datasheets [11].

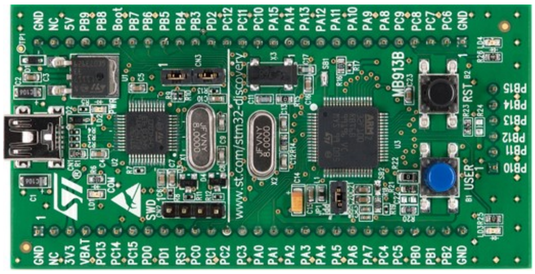


Figure 4.3. STM32VLDiscovery kit with an in–circuit programmer/debugger.

Big advantage of this microcontroller is uploading of new software by the SWD interface — only 4 connections are required (+3.3V, SWCLK, SWDIO and GND). Special and commercially available programmers can be used too. Low-cost STM32VLDiscovery development (Fig. 4.3) board was used to provide interface between CoCoox IDE and on–board microcontroller unit using an in–circuit ST–Link debugger/programmer. Using this board programmers can easily upload new programs into the flight control unit or debug and instruction–by–instruction check their code.

4.1.2 On–board sensors

Figure 4.4. Used on–board sensors. a) GPS sensor, b) 3D accelerometer and gyroscope sensor.

External sensors are used to measure position and orientation of the aircraft. Localization and ground speed of the aircraft is measured by Maestro Wirelles Solutions GPS receiver unit A2035–H (Fig. 4.4a) based on SiRFstarIV chip sets. Different communication interfaces are available within this chip (UART, SPI and I^2C) and two data protocols can be used (NMEA and SiRF). For our purposes UART interface and NMEA protocol was used. Data update rate can be set to 1 or 5 Hz and position accuracy of this module is lower than 2.5 meters in open space. This should be enough for our wind estimation algorithms.

Small interface board (Fig. 4.5) was developed for easier manipulation with selected GPS sensor. This board contains several parts required to proper functionality of this sensor. However, the main task of this board is to implement simple interface for easy manipulation and quick cable connection to the main control board via UART bus.

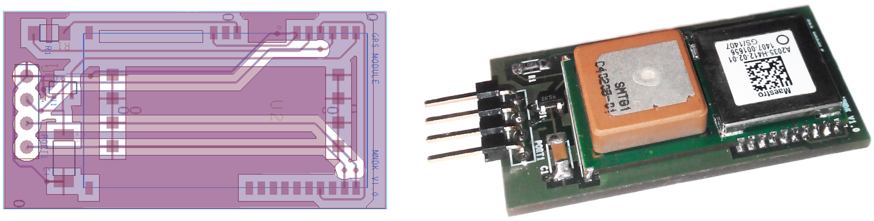


Figure 4.5. GPS module board a) OrCad PCB design, b) finished board.

Measuring inertial position is not enough. To supply information about orientation and rotation of the aircraft (e.g. yaw, pitch and roll angle and appropriate angular rates) small 3D accelerometer and gyroscope SMD sensor (IMU sensor LSM330DL, Fig. 4.4b) is used. Communication is provided by SPI interface and interface update rate is set to 10 Hz and can be adjusted up to 50 Hz. This sensor is soldered directly to the on–board module.

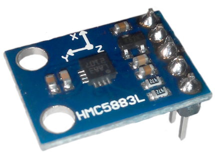


Figure 4.6. External magnetometer HMC5883L connected via I^2C bus.

Simple heading angle computation algorithm (using LSM330DL sensor) was implemented in the original on–board autopilot software [3]. This algorithm used integration of the heading angle rate, but it led to quite slow and improper function of this computation (the integration error). Then external digital magnetometer module (HMC5883L, Fig. 4.6) was added and the integration problems were resolved. It also led to precise alignment of measured north to actual magnetic north of the Earth. The usage and software programming of this sensor is similar to usage of IMU sensor described above. An example of I^2C bus initialization and heading angle data reading using this module can be found in section 5.1.2.

More informations about GPS, IMU and magnetometer sensor can be found in relevant datasheets [12], [13], [14].

4.1.3 Additional on–board components

All measured data are stored and send to ground station, where can be logged too. For on-board data logging 64 MB flash memory N25Q064A by Micron with SPI interface is used. Last and very important part of the on–board system is wireless communication channel with ground station. Commercially available X–Bee modules (Fig. 4.7) were chosen to fulfil this need. These modules simply provide wireless UART communication channel and reliability of this solution is quite good for reasonable distances between the ground station and the aircraft.

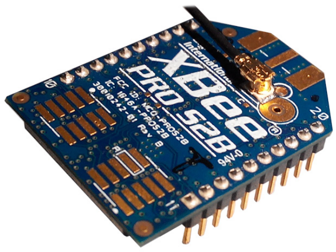


Figure 4.7. Wireless XBee module.

4.2 Differential pressure sensor board

For proper function of wind estimation algorithm, information about wind speed, angle–of–attack and sideslip angle must be available. For this reason measurement board (Fig. 4.8) with three differential pressure sensors is connected to three microcontroller ADC channels. This module was developed within diploma thesis [5] by CTU student Petr Pahorecký. Unfortunately, this module was not finished and ready for implementation inside plane. This module was then improved to fulfil all our needs — measuring all three differential pressures.

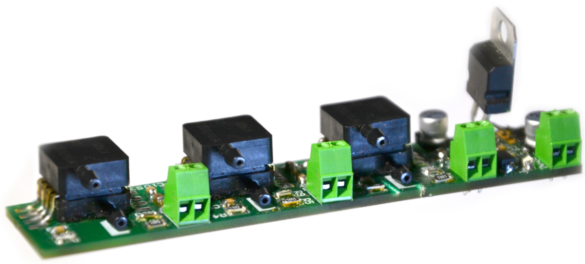


Figure 4.8. Differential pressure measurement board.

Three FreeScale MPXV7002DP differential pressure sensors are mounted on this measurement board. Used sensors works under 5 V power supply, voltage divider to 3.3 V is present on board for correct voltage connection to microcontroller. Maximum range of used pressure sensors is from -2 to +2 kPa, so this measurement board provides resolution about 1 Pa per bit change.

The purpose of using three sensors is to connect six outputs from advanced 5-hole pitot tube (see Fig. 7.1b). The pitot tube for our UAV with proper mounting was developed by Vojtěch Rubáš within his Diploma Thesis [15]. From pressure difference between front hole and side holes the air speed of the aircraft can be computed. From pressure difference between upper and lower hole angle-of-attack can be estimated and similarly from difference between left and right hole the sideslip angle can be computed. Due to well–known rounded head shape differential pressure drop caused by big aerodynamic angles is reduced unlike in conical or pyramid shaped probes.

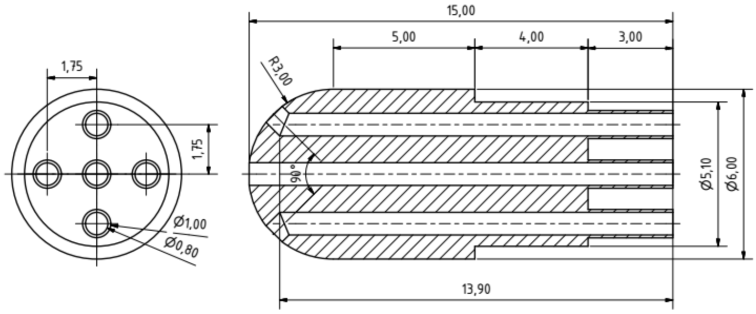


Figure 4.9. Drawing of multi–hole pitot tube.

Calibration of all pressure sensors was done and the measurement process and the results are presented in section 7.1.

4.3 Ground control station

One of the main tasks of the ground control station is to replace original transmitter of RC system. The complex ground station is chosen instead of only xBee-USB adaptor used in similar projects to provide robust and reliable communication with the on–board control unit.

The basic hardware of the ground control station is based on the flight control unit board. The same microcontroller is used, but different accessories are equipped to fulfil needs for interaction between pilot and ground station.

Figure 4.10. Ground control station with RC aircraft transmitter.

Unlike at the flight control board, some hardware parts for input commands must be included in the ground control station design. Four mechanical switches and four trimmers are added to provide control of aircraft control surfaces. Several menu buttons and small graphical LCD display can provide selected informations about aircraft during flight mission to the pilot of the aircraft. Two UART ports are included, one for wireless serial XBee communication and the other for serial communication channel with PC using USB–TTL cable converter or with Android phone using UART–Bluetooth module.

Chapter 5
Software

Several different pieces of code in C language are given in the following section. This code should help new students using our flight control board to quick orientation in flight control board programming. Several others parts of the code (e.g. the initialization of internal timers, external interrupts, communication buses and others) can be found in the Jaroslav Halgašík's thesis [3]. All source codes can be found in attached CD.

5.1 Flight control board software

The on–board software and the ground control station software are very similar. Essential difference is in the implementation of code controlling various connected sensors (e.g. the on–board GPS and IMU — the ground station display and multiplexed PWM)

5.1.1 PWM signal for servo control

The on–board implementation of servo control is easy. The microcontroller uses several internal timers to generate PWM signals for thrust and ailerons, rudder and elevator servos. The timer in the ground station used to read multiplexed PWM signal from connected RC transmitter (See section 5.2.2) must be initialized to the same values as internal timers for servo control for easy manipulation with PWM data. The ground station sends an integer number, which is then directly written into timers registers and PWM signal is generated. Thus no computation is required.

```
...
//MAVLink servo control handle
int rc_raw_read_handle(const mavlink_message_t *msgR){

    //Data from RC transmitter
    if(mavlink_msg_rc_channels_raw_get_port(msgR) == 0){
        mavlink_msg_rc_channels_raw_decode(msgR, &rc_raw);
    }

    //Data from ground station potentiometers and switches
    if(mavlink_msg_rc_channels_raw_get_port(msgR) == 1){
        mavlink_msg_rc_channels_raw_decode(msgR, &station_raw);
    }
}
...
```

In the code above the handling of the servo commands is implemented. There are two different messages (one from RC transmitter, second one from potentiometers and switches integrated in the ground station). We can resolve them by MAVLink port number — as seen in the `if` structure of the code above. The integer data (timer value in microseconds) for all servo channels are then saved into variables with structure explained in table 5.1.

Received data are then processed by the on–board microcontroller. The data for servos are directly written into registers of relevant timers to generate PWM signals. This code is shown below.

```
...
//Write control data into registers of relevant timers
void setServos(void) {
TIM1->CCR4 = (int)rc_raw.chan4_raw;    //Thrust
TIM1->CCR1 = (int)rc_raw.chan1_raw;    //Rudder
TIM4->CCR4 = (int)rc_raw.chan2_raw;    //Aileron
TIM4->CCR3 = (int)rc_raw.chan3_raw;    //Elevator
}
...
```

The initialization of the timers generating PWM signal can be found in code included in attached CD. (file init.c — methods TIM1_Inicializace and TIM4_Inicializace)

Data type	Name	Explanation
uint32_t	time_boot_ms	Timestamp (milliseconds since system boot).
uint16_t	chan1_raw	RC channel 1 value, in microseconds.
uint16_t	chan2_raw	RC channel 2 value, in microseconds.
	...	
uint16_t	chan8_raw	RC channel 8 value, in microseconds.
uint8_t	port	Servo output port (set of 8 outputs = 1 port). Port 1 - RC transmitter, Port 2 - Ground station
uint8_t	rssi	Receive signal strength indicator, 0: 0%, 255: 100%

Table 5.1. Servo control data structure.

5.1.2 Reading and correcting magnetometer data

External magnetometer is connected via I_2C bus. The complete 12 bit value is divided into two bytes, so the full 3D measurement magnetometer data are made by reading 6 different sensor registers. The reading from magnetometer registers is shown in the following code example.

```
...
//Read raw data from magnetometer
void hmc5883_raw() {
    uint8_t buf_top; //upper 8 bits of measurement
    uint8_t buf_low; //lower 8 bits of measurement

    //Read magnetometer data X axis
    i2c_read(0x1E,0x03,&buf_top,1);
    i2c_read(0x1E,0x04,&buf_low,1);
    rawMagData.x = (buf_top << 8) | buf_low;

    //Read magnetometer data Z axis
    i2c_read(0x1E,0x05,&buf_top,1);
    i2c_read(0x1E,0x06,&buf_low,1);
    rawMagData.z = (buf_top << 8) | buf_low;

    //Read magnetometer data Y axis
    i2c_read( 0x1E, 0x07, &buf_top, 1);
```

```
        i2c_read( 0x1E, 0x08, &buf_low, 1);
        rawMagData.y = (buf_top << 8) | buf_low;
    }
    ...
```

Raw data from magnetometer should be calibrated using process described in Section 7.2. The following code shows simple computation of correct magnetometer values.

```
...
//Compute calibrated magnetometer data
void hmc5883_update(vec3 * magData) {
    vec3 res;

    hmc5883_raw(); //Read raw data from magnetometer

    //Subtract bias vector
    res = diff(rawMagData,biasVector);

    //Transformate vector
    res = mat_vec_multiply(callibrationMatrix,res);

    //Save data
    magData->x = res.x;
    magData->y = res.y;
    magData->z = res.z;
}
...
```

Reading position data from the on–board inertial measurement unit (accelerometer and gyrometer) is very similar.

5.2 Ground station software

The main program of the ground control software is very similar to the on–board software, but has several different functions — such as to display status and measured values or to provide direct control input from user.

5.2.1 Display

The ground control station includes small graphical display (84 x 48 pixel). This display is mounted on special board to provide communication and other connection pins — the SPI bus and power pins (Fig. 5.1).

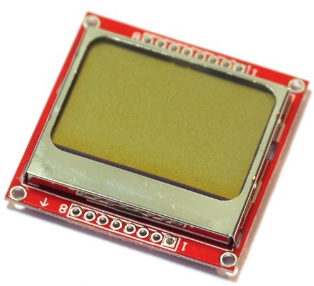


Figure 5.1. LCD display with integrated PCD8544 driver.

This display uses integrated driver PCD8544. This driver is very popular worldwide and many different libraries are available. For our project we implemented C/C++ library for PCD8544 driver developed originally for Arduino system by Henning Karlsen [16]. This library is licensed under a CC BY–NC–SA 3.0 (Creative Commons Attribution–NonCommercial–ShareAlike 3.0 Unported) License. For more information about this licence see [17].

5.2.2 Multiplexed PWM signal resolving

The data from the RC transmitter are coded in multiplexed PWM signal. The explanation of this coding can be found in Figure 5.2 and an example of real signal with seven different multiplexed channels is in Figure 5.3.

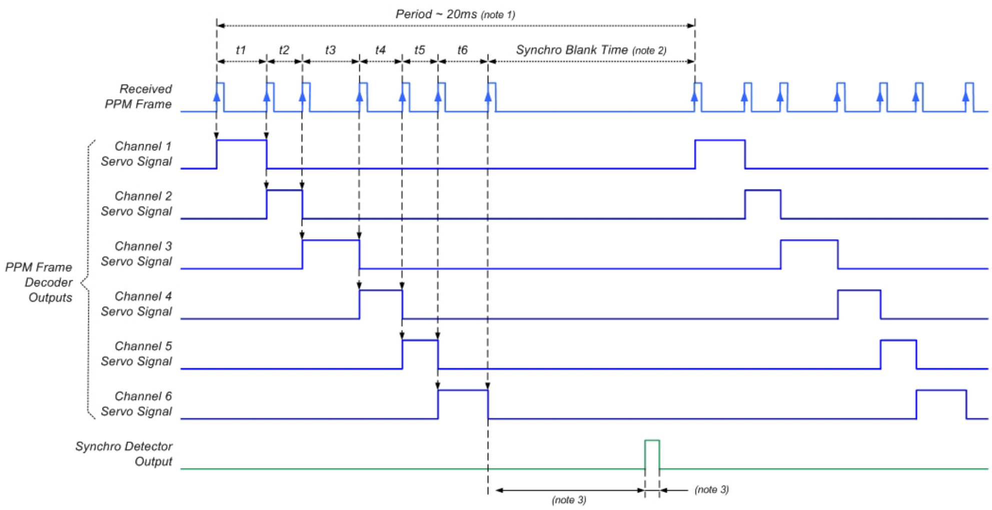


Figure 5.2. Explanation of multiplexed PWM signal.

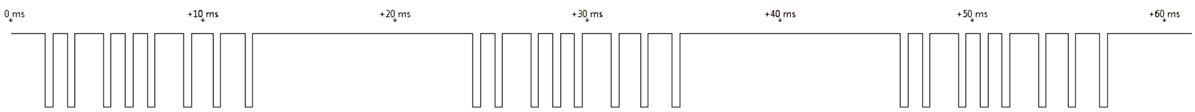


Figure 5.3. Example of multiplexed PWM signal — seven channels.

The channel values of the multiplexed PWM signal can be easily resolved using precise internal microcontroller timers. When external interrupt occurs (falling edge of the voltage), the timer starts to measure time in microseconds. After another interrupt it saves the channel time into variable and starts measuring another time for the second channel. This cycle repeats until all channels are resolved (or the time between two external interrupts is too long — this indicates new value for channel 1). Resolving code (from [3]) is provided in the following example.

```
...
//Handle external interrupt - RC transmitter multiplexed PWM
void EXTI9_5_IRQHandler(void) {

    if (EXTI_GetITStatus(EXTI_Line6) != RESET)
    {

     //Read timer value and start another measurement
        rc_tmp = TIM3->CNT;
```

```
        TIM3->CNT = 0;

        //Start another set of measurement
        if(rc_tmp > 5000) {
            rc_index = 0;
        } else {
            rc_times [rc_index] = rc_tmp;     //Save the time into
            if(rc_index < 10){                //the relevant position
                rc_index++;                   //of the RC variable
            } else {
                rc_index = 0;
            }
        }

        //Clear interruption
        EXTI_ClearITPendingBit(EXTI_Line6);
    }
}
...
```

The following code then generates MAVLink packets containing control information for servos (from the code above) or for the current function of flight control board.

```
...
//Send message containing control data
void send_RC(void){

    //RC transmitter data
    rc_raw.chan1_raw =  rc_times [0];
    rc_raw.chan2_raw =  rc_times [1];
    rc_raw.chan3_raw =  rc_times [2];
    rc_raw.chan4_raw =  rc_times [3];
    rc_raw.chan5_raw =  rc_times [4];

    //Select MAVLink port
    rc_raw.port = 0;

    //Generate MAVLink packet
    mavlink_msg_rc_channels_raw_encode(system_id,
            component_id, &msg, &rc_raw);
    mav_len = mavlink_msg_to_send_buffer(buf, &msg);

    //Send message via XBee
    sendUART1((char*)&buf[0],mav_len);

    //Ground station data
    rc_raw.chan1_raw =  adcValues [0];
    rc_raw.chan2_raw =  adcValues [1];
    rc_raw.chan3_raw =  adcValues [2];
    rc_raw.chan4_raw =  adcValues [3];
    rc_raw.chan5_raw =  switches [0];
    rc_raw.chan6_raw =  switches [1];
    rc_raw.chan7_raw =  switches [2];
    rc_raw.chan8_raw =  switches [3];
```

```
        //Select MAVLink port
        rc_raw.port = 1;

        //Generate MAVLink packet, send message via XBee
        mavlink_msg_rc_channels_raw_encode(system_id,
                component_id, &msg, &rc_raw);
        mav_len = mavlink_msg_to_send_buffer(buf, &msg);
        sendUART1((char*)&buf[0],mav_len);
    }
    ...
```

5.3 MAVLink communication protocol,

The Micro Air Vehicle Link protocol was first released in 2009 and it is currently used in various control systems such as ArduPilot [9] and it is implemented in many additional projects to advance functionality of the control system. Due to projects such as DroidPlanner (Android) [18] or QGroundControl (Windows/Linux) [19] user is able to extend simple display of ground station by complex graphical and status displays, artificial horizonts or even by simple command interface.

In our system MAVLink communication protocol is implemented. The complex libraries written in different programming languages are available such as C or Python. This very efficient protocol is inspired by industrial standards CAN and SAE AS-4 and includes simple data correction mechanism. Decription of MAVLink can be seen in Figure 5.4 and explained in Table 5.2.

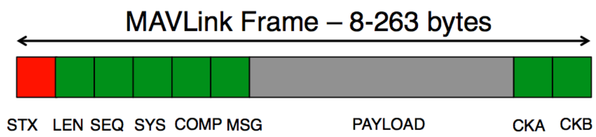


Figure 5.4. MAVLink communication protocol. Packet anatomy.

Byte	Content	Explanation
STX	Packet start sign	Indicates the start of a new packet.
LEN	Payload length	Indicates length of the following payload.
SEQ	Packet sequence	Each component counts up his send sequence. Allows to detect packet loss.
SYS	System ID	ID of the SENDING system. Allows to differentiate different MAVs on the same network.
COMP	Component ID	ID of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot.
MSG	Message ID	ID of the message - the ID defines what the payload "means" and how it should be correctly decoded.
Payload	Data	Data of the message, depends on the message id.
Checksum	Low, high byte	ITU X.25/SAE AS-4 hash

Table 5.2. MAVLink communication protocol. Packet content explanation.

Since used asynchronous communication channel (provided by XBee modules) does not usually have mechanism to control reliability, correctness and the ability to detect errors of received messages, the implementation of these mechanisms must be somehow solved in software. If the RC plane receives control messages from the ground station with some errors, the on–board control systems set control surfaces to wrong position and this situation can cause stall or in an extreme case even crash of the plane.

The simplest way to implement correcting mechanism is the checksum. With advance we use the MAVLink protocol, because it implements the checksum in default. Another advantage of this protocol are so called ”HeartBeat” messages. By checking of these messages sent periodically by the ground control system or the on–board system, the software is able to resolve some hazardous events such as UAV is out of ground control station signal range.

Standardized libraries are easy to use and reduce the time required for full implementation of the message coding and decoding. For every type of communication message several methods for creating, editing and reading MAVLink packets are implemented.

The implementation of method for creating and sending ”HeartBeat” message is shown as follows.

```
...
// Mavlink HeartBeat Message
int heartbeat_telemetry(void){

    //generate HeartBeat message
    mavlink_msg_heartbeat_pack(system_id, component_id, &msg,
    MAV_TYPE_FIXED_WING, MAV_AUTOPILOT_GENERIC,
    (uint8_t)autopilot_state, 0, MAV_STATE_ACTIVE);

    //copy message to buffer
    mav_len = mavlink_msg_to_send_buffer(buf, &msg);

    sendUART1((char*)&buf[0],mav_len); //send message via UART1
    return 0;
}
...
```

Decoded example of ”HeartBeat” message sent via asynchronous communication channel can be found in Figure 5.5.

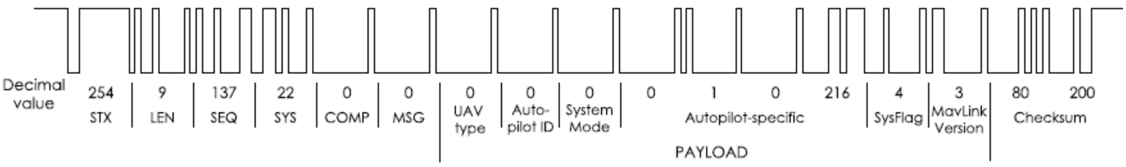


Figure 5.5. MAVLink communication protocol. Example of ”HeartBeat” message.

In following example process of creating message carrying six ADC values is described.

```
...
// Mavlink ADC Message
// int adc[] - voltage values from ADC
mavlink_msg_ap_adc_pack(system_id, MAV_COMP_ID_IMU, &msg, adc[0],
adc[1], adc[2], adc[3], adc[4], adc[5]);
mav_len = mavlink_msg_to_send_buffer(buf, &msg);
```

```
sendUART1((char*)&buf[0],mav_len);
...
```

Decoded example of MAVLink packet sent via asynchronous communication channel carrying some ADC data (airspeed, angle–of–attack and sideslip angle) can be found in Figure 5.6. This type of message can contain up to six 16 bit values.

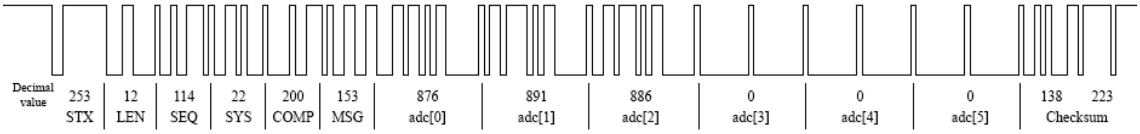


Figure 5.6. MAVLink communication protocol. Decoded ADC data message using asynchronous serial communication.

Chapter 6
Wind estimation algorithm

6.1 Algorithm derivation

If we look into special situation when plane is not moving and surrounding atmosphere is flowing in parallel direction with main X-axis of the plane (like in wind tunnels), we can say, that air speed of the plane is the same as the atmosphere flow speed.

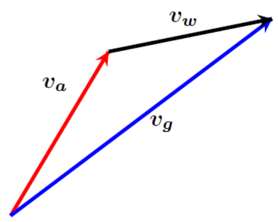


Figure 6.1. Vector wind triangle.

If we consider some wind distortion in the atmosphere example, we simply add wind vector to the atmosphere flow speed vector. The resulting vector is the air speed of the plane. Applying this principle onto the flying plane we discover, that the air speed of the plane minus the ground speed of the wind is actually the ground speed of the plane. This situation is shown in Fig. 6.1 and in Eq. (1), where $\mathbf{v_g}$ is the ground speed of the aircraft, $\mathbf{v_a}$ is the aircraft velocity with respect to the aerodynamic reference frame (the air speed) and $\mathbf{v_w}$ is the wind speed with respect to the ground reference system.

(1)

$$\mathbf{v_g} = \mathbf{v_a} + \mathbf{v_w}$$

If we express components of the wind speed from equation (1) we obtain equations (2), where α is the angle–of–attack, β is sideslip angle, γ is the flight path angle ($\gamma = \theta - \alpha$), ψ is the heading angle of the aircraft, $[\dot{x} \quad \dot{y} \quad \dot{z}]^T$ is the ground speed of the aircraft and V_a is the air speed of the aircraft with respect to the aircraft reference frame. These equations were adopted from [20].

(2)

$$V_{wx} = \dot{x} - V_a \cos\gamma \cos\beta \cos\psi + V_a \cos\gamma \sin\beta \sin\psi$$

$$V_{wy} = \dot{y} - V_a \cos\gamma \cos\beta \sin\psi - V_a \cos\gamma \sin\beta \cos\psi$$

$$V_{wz} = \dot{z} - V_a \sin\gamma$$

In these equations it is assumed that aircraft is performing only wing–level flight (with roll angle $\phi = 0$).

However, for our wind prediction algorithm we need general case of these computational equations. Then we are able to compute wind speed during advanced flight maneuvers such as coordinated turn using controlled bank angle. For this purpose we need to express the air speed of the aircraft in the ground reference frame using transformation matrices $\mathbf{R_I^B}$ and $\mathbf{R_B^W}$ derived in section 3.2. Final extended equations of wind estimation algorithm are derived as follows.

(3)

$$\begin{bmatrix} V_{wx} \\ V_{wy} \\ V_{wz} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_{GPS} - \mathbf{R_I^B R_B^W} \begin{bmatrix} V_a \\ 0 \\ 0 \end{bmatrix}$$

Previously discussed equations (2) are actually in their final form only simplified version of our new equations (3) (with zero roll angle).

Computing final wind speed from wind speed vector is easy. We simply compute magnitude of this vector.

(4)

$$V_w = \sqrt{V_{wx}^2 + V_{wy}^2 + V_{xz}^2}$$

6.2 Measurement sesitivity to errors

In the wind prediction algorithm several non–linear computations are used. The most used computations are goniometric functions. If some measurement is somehow incorrect — which is highly possible due to usual sensor noise — the final computational result can be incorrect too or on the other hand they can be correct even if some sensor provides wrong measurement.

Analysis of the measurement errors is based on this idea. We can show the relation between some different values of flight variables, different measurement error and the error of wind speed computation. The example of this relation is shown in the example of incorrect angle–of–attack measurement.

(5)

$$\alpha' = \alpha_{true} + \alpha_e$$

If the true value of the angle–of–attack in the computational algorithm (Eq. (3)) is replaced by measured value α' (Eq. (5)) with additional measurement error and from resulting equations original equations are subtracted, we obtain analytical expression of relation between the estimated wind speed measurement error and the angle–of–attack measurement error. This relation is shown in Equation (6), but only one V_{wx} component of the wind speed vector is presented due to complexity of these equations.

(6)

$$\begin{aligned} V_{wxErr} = V_a \sin(\alpha + \alpha_e)\cos(\beta)\left(\sin(\phi)\sin(\psi) + \cos(\phi)\cos(\psi)\sin(\theta)\right) \\ -V_a \cos(\beta)\sin(\alpha)\left(\sin(\phi)\sin(\psi) + \cos(\phi)\cos(\psi)\sin(\theta)\right) \\ +V_a \cos(\alpha + \alpha_e)\cos(\beta)\cos(\psi)\cos(\theta) - V_a \cos(\alpha)\cos(\beta)\cos(\psi)\cos(\theta) \end{aligned}$$

Similarly we can express all different relations between all incorrectly measured values and the total error V_{wErr} of the computed value of wind speed V_w.

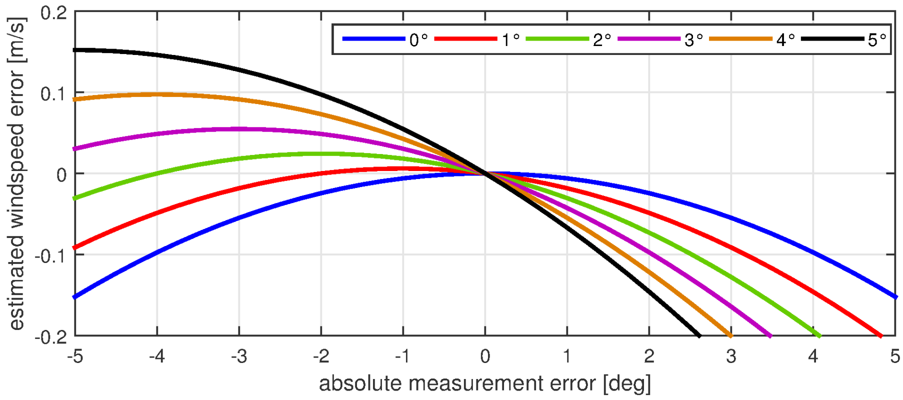


Figure 6.2. Relation between angle–of–attack measurement error and error of computed V_{wx} component of the wind speed vector.

The relation between the angle–of–attack measurement error and computed V_{wx} component of wind speed vector (Eq. (5)) is shown in Figure 6.2. In this figure we can see that if the aircraft is flying with zero angle–of–attack $\alpha = 0°$ (blue line), total error of wind speed component (Y–axis) is smaller than when plane is flying with angle–of–attack $\alpha = 5°$ (black line) for the same absolute measurement error (X–axis) of both measurements.

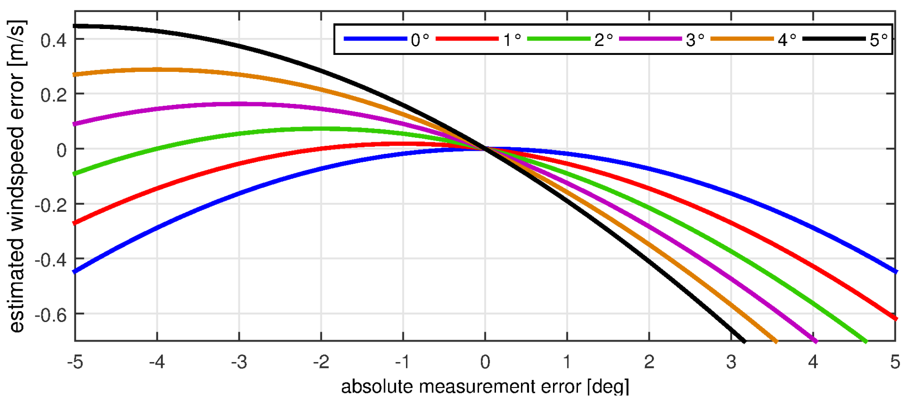


Figure 6.3. Relation between angle–of–attack measurement error and error of computed V_w wind speed vector.

In figure 6.3 the relation between total V_w wind speed computational error and angle–of–attack measurement is shown. It is clear that for this particular flight measurement does not matter whether angle–of–attack α is measured as positive or negative angle. This effect can be seen for example in measurement for $\alpha = 2°$ (green line). If absolute angle–of–attack measurement error is zero ($\alpha_e = 0°$), total error of wind speed vector is null. Then if absolute angle–of–attack measurement error is minus 4 ($\alpha_e = -4°$), which gives final measurement value $\alpha' = -2°$, total error of wind speed vector is null too.

Using similar procedure we can even express relation between computed wind speed error and angle–of–attack measurement error α' considering different heading or different flight variables. This process simulates different flight states during flight performance.

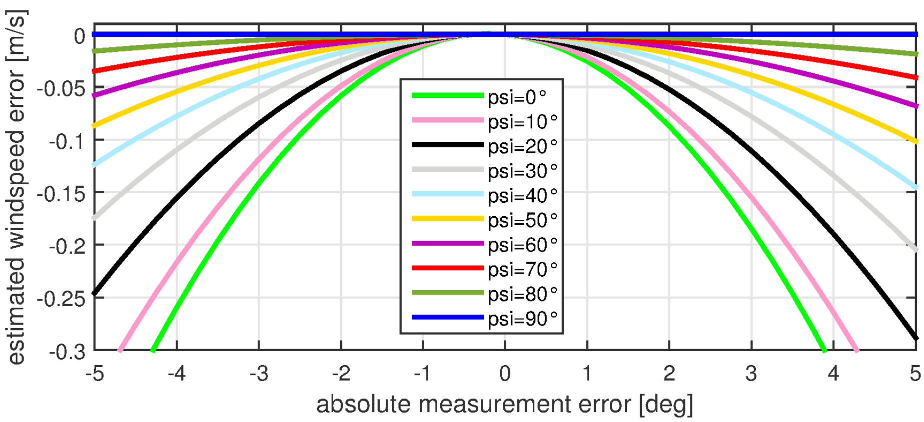


Figure 6.4. Relation between angle–of–attack measurement error α_e and error of computed V_w wind speed vector for different heading angle ψ.

In figure 6.4 the relation between computed wind speed error and angle–of–attack measurement error considering different heading is shown. It is clear that if plane is flying in the same direction as the wind blows, the total computed wind speed error is bigger than when the plane is flying at an angle of 90 degrees from the wind direction. Then the angle–of–attack measurement error does not affect the total computed wind speed error.

The same procedure, as mentioned above, can be applied on every measurement (airspeed, angle–of–attack, sideslip angle, euler angles and GPS position and speeds). Some of these figures representing different relations are shown and commented below and in appendix D.

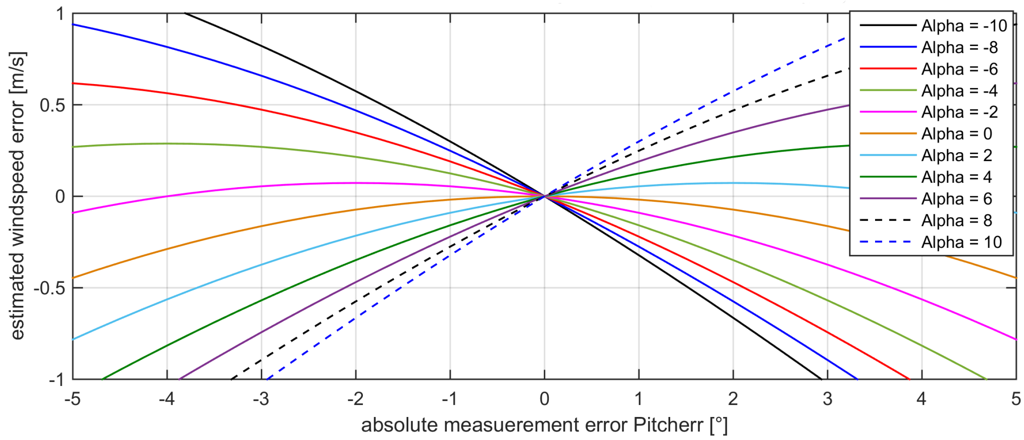


Figure 6.5. Relation between pitch angle measurement error θ_e and error of computed V_w wind speed vector for different angle–of–attack α.

In figure 6.5 the relation between pitch angle measurement error and total computation error according to the different angle–of–attack is shown. It is clear that the error in Pitch measurement affects the final computation result less when angle–of–attack is small then when this angle is bigger.

On the other hand some computational errors are independent on some measured errors. For example in Figure 6.6 relation between airspeed measurement error and total wind speed computational error according to change of angle–of–attack is shown. The relation is almost linear and does not depend on actual angle–of–attack value.

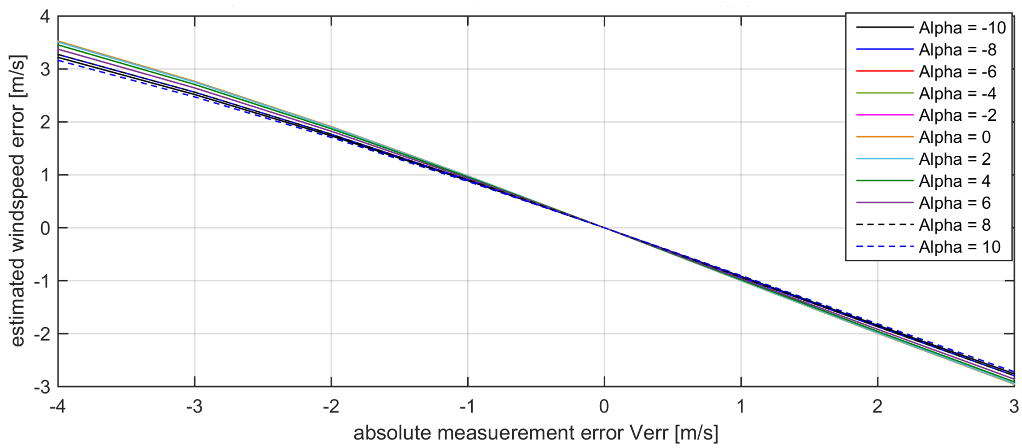


Figure 6.6. Relation between airspeed measurement error V_{ae} and error of computed V_w wind speed vector for different angle–of–attack α.

Another independent relation is shown in figure 6.7. This relation between sideslip angle measurement error and total wind speed computational error according to change of roll angle has almost quadratic behaviour.

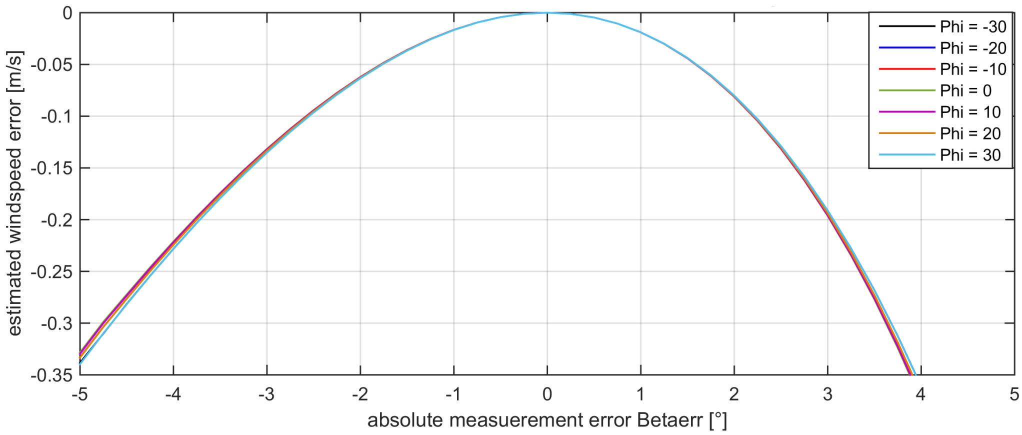


Figure 6.7. Relation between angle–of–attack measurement error β_e and error of computed V_w wind speed vector for different roll angle ϕ.

6.3 Simulation of wind prediction

The correctness of the wind prediction algorithm was tested using simplified realistic Simulink models of plane. The basic of this model is inspired by example flight models from AeroSim Blockset plugin [8] for Simulink and was described in section 3.4.

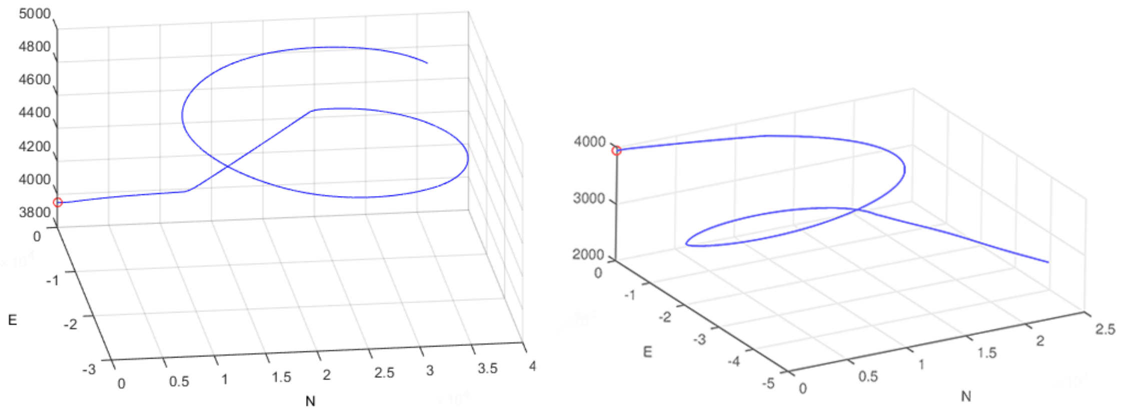


Figure 6.8. Simple flight paths for algorithm verification.

The simplified realistic model was extended by the implementation of external wind and some basic autopilots. Several different flight paths were simulated, two examples of them are shown in Figure 6.8.

During the flight performance along these flight paths the aircraft is exposed to various wind speeds. An example of these reference wind speeds in one direction can be seen in Figure 6.9. Calculated wind speeds from algorithms are present too.

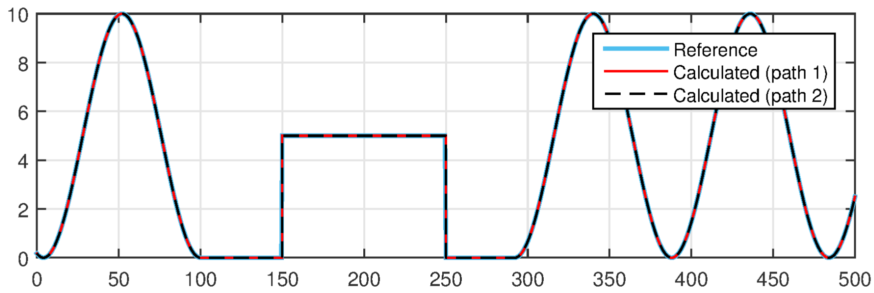


Figure 6.9. Comparison of estimated and true value of wind speed.

In these results it can be easily seen that developed algorithm works very nicely for simulation of random flight path with exact and correct measurement values.

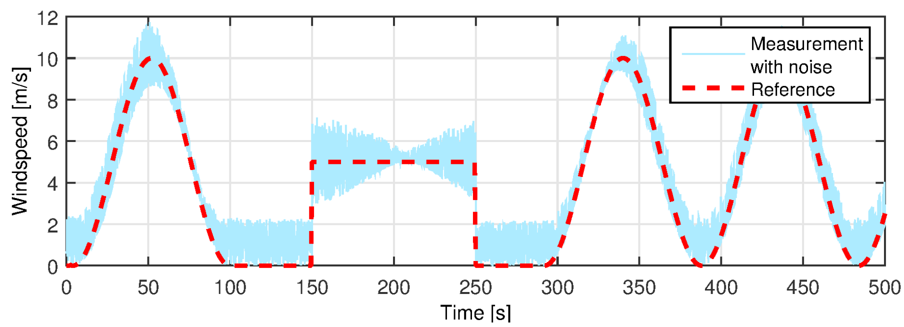


Figure 6.10. Comparison of estimated and true value of wind speed for noisy measurements.

However, in the real world achieving zero measurement error without external influence noise or the sensor noise is nearly impossible. Thus some additional white noise was added to the measured values. The result is shown in Figure 6.10. We can immediately see, that when the UAV is flying at an angle of 90 degrees from the wind direction, the wind speed estimation error is nearly zero (Fig. 6.10, at time $t = 75$ s, $t = 205$ s and $t = 325$ s). The same results were obtained in the measurement sensitivity Section 6.2.

In the simulation graphs and analysis presented above it is assumed only white sensor noise of the measurement values. Another hazardous error is the overall bias or drift of the sensor values. This type of error can resolve into unexpected behaviour of the system.

In Figure 6.11 several bias error in angle–of–attack measurement are shown.

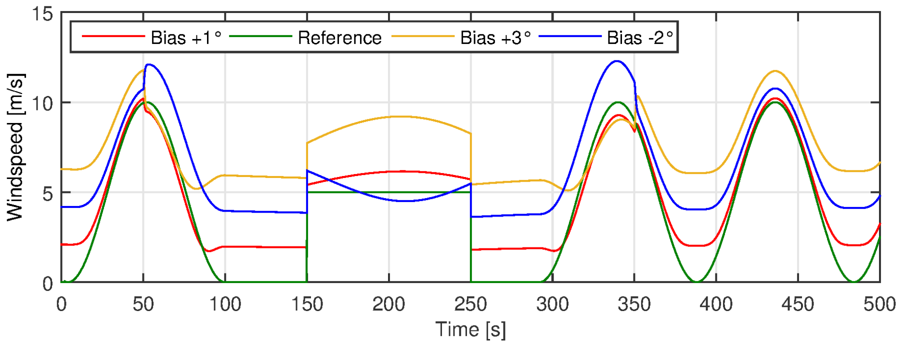


Figure 6.11. Comparison between actual value of wind speed and several computed values with different sensor bias error.

It is clearly shown, that different biases have different impacts on final computed value of the wind speed. For example if all measured values of angle–of–attack are by one degree bigger then the real value, the final computed value of the wind speed is almost as the real value. On the other hand if overall bias is by three degrees bigger, the computed value of wind speed is in the most cases very different from the real value from the actual wind speed.

Based on several eror analysis and simulation methods mentioned and presented above the development and implementation of some advanced filtering methods (e.g. Kalman filtering methods or sensor fusion) is highly recommended. These filters can provide robustness and measurement error resistance to our wind prediction algorithm.

6.4 Software implementation

Lot of basic mathematical functions, such as matrix multiplication or cross product of two vectors, were implemented in source code of Jaroslav Halgašík's thesis [3]. Therefore the C language implementation of the wind prediction algorithm was quite easy. This problem was divided into two parts (and two methods actually): the creation of the transformational matrix from aerodynamic to ground fixed reference frame and the algorithm equation itself.

First method takes all telemetric data (airspeed, angle–of–attack, sideslip angle, pitch, roll and yaw angle) and computes the aerodynamic–to–ground transformation matrix. The sub–resulting transformational matrices (aerodynamic–to–body and body–to–ground) are stored in memory too.

```
...
void createTransformationMatrix(telemetryData data) {
    //initialization of transformation matrix
    mat33 transIB;                //ground-to-body
    mat33 transBA;                //body-to-wind
```

```
    //computations SIN and COS values of flight angles
    float sinAlpha = sin(data.alpha);
    float cosAlpha = cos(data.alpha);
    float sinBeta = sin(data.beta);
    float cosBeta = cos(data.beta);
    float sinPhi = sin(data.roll);
    float cosPhi = cos(data.roll);
    float sinTheta = sin(data.pitch);
    float cosTheta = cos(data.pitch);
    float sinPsi = sin(data.yaw);
    float cosPsi = cos(data.yaw);

    //body-to-ground matrix initialization
    transIB.d11 = cosPsi*cosTheta;
    transIB.d21 = sinPsi*cosTheta;
    transIB.d31 = -sinTheta;
    transIB.d12 = cosPsi*sinTheta*sinPhi - sinPsi*cosPhi;
    transIB.d22 = sinPsi*sinTheta*sinPhi + cosPsi*cosPhi;
    transIB.d32 = cosTheta*sinPhi;
    transIB.d13 = cosPsi*sinTheta*cosPhi + sinPsi*sinPhi;
    transIB.d23 = sinPsi*sinTheta*cosPhi - cosPsi*sinPhi;
    transIB.d33 = cosTheta*cosPhi;

    //aerodynamic-to-body matrix initialization
    transBA.d11 = cosAlpha*cosBeta;
    transBA.d21 = sinBeta;
    transBA.d31 = sinAlpha*cosBeta;
    transBA.d12 = -cosAlpha*sinBeta;
    transBA.d22 = cosBeta;
    transBA.d32 = -sinAlpha*sinBeta;
    transBA.d13 = -sinAlpha;
    transBA.d23 = 0;
    transBA.d33 = cosAlpha;

    //store final aerodynamic-to-ground transformational matrices
    transformationMatrixBA = transBA;
    transformationMatrixIB = transIB;
    transformationMatrixIA = mat_multiply(transIB,transBA);
}
...
```

The second method is the main computational algorithm. At the beginning the airspeed vector in aerodynamic frame is initialized. Then the transformational matrices are created using method described above and then the components of wind speed vector are computed. Finally the result containing the vector component of the wind speed is returned as output value of this function.

```
...
vec3 computeWindSpeed(vec3 gpsSpeeds, telemetryData data) {
    //declaration of airSpeed vector in aerodynamicFrame
    vec3 airSpeedW;

    //declaration of airSpeed = [Va; 0; 0]
```

```
    airSpeedW.x = data.airSpeed;
    airSpeedW.y = 0;
    airSpeedW.z = 0;

    //create transformational Matrices
    createTransformationMatrix(data);

    //compute airSpeed vector in groundFrame using transformation matrix
    vec3 airSpeed = mat_vec_multiply(transformationMatrixIA, airSpeedW);

    //difference between groundSpeed and airSpeed
    vec3 windSpeed = diff(gpsSpeeds, airSpeedI);

    //return windSpeed vector in groundFrame
    return windSpeed;
}
...
```

The critical part of this implementation is the long computational time. If this computation takes too much time, the on–board microprocessor unit can skip some important parts of the software. As an result the plane can perform some unwanted maneuvers or even crash in extreme case.

Fortunately execution of wind prediction algorithm takes only 0.8 milliseconds in average of computational time. With periodicity of 50 Hz the wind speed computation takes 1/20 of one cycle time. This is quite a little time, but several simplifying steps can be done.

One example of simplification is shown below. Since the air speed vector in aerodynamic frame has only one non–zero component, several components of wind–to–body transformation matrix can be zero to save some computational time.

```
...
    //aerodynamic-to-body matrix initialization
    transBA.d11 = cosAlpha*cosBeta;
    transBA.d21 = sinBeta;
    transBA.d31 = sinAlpha*cosBeta;
    transBA.d12 = 0;        // -cosAlpha*sinBeta
    transBA.d22 = 0;        //  cosBeta
    transBA.d32 = 0;        // -sinAlpha*sinBeta
    transBA.d13 = 0;        // -sinAlpha
    transBA.d23 = 0;
    transBA.d33 = 0;        //  cosAlpha
...
```

This simplification can save up to 0.05 milliseconds. Another simplification can be done reducing this transformational matrix into vector. Then the computation reduces from multiplication of transformational matrix (aerodynamic–to–body) and vector (air speed) into multiplication of vector (reduced aerodynamic–to–body transformational matrix) and scalar (air speed V_a).

For possible future implementations of advanced algorithms all parts of wind speed computations are left in their basic unsimplified form. The computed transformational matrices can be used for different algorithms in the future.

Chapter 7
Sensor calibration

7.1 Pitot tube and pressure sensors

Precise calibration of differential pressure sensors is very important to correct estimation of the aerodynamic angles — angle–of–attack and sideslip angle — and for correct computation of the wind speed by developed algorithm (in Chapter 6). Several experiments in wind tunnel in Aerospace Research and Test Establishment in Prague. These experiments were performed for collecting and analysis of measurement samples and then for developing correct estimation algorithm for aerodynamic angles.

Figure 7.1. Wind tunnel. a) Overall view of wind tunnel with VZLU/CVUT aircraft, b) Detail of the multi–hole pitots tube

The wind tunnel used for the multi–hole pitots tube calibration (Fig. 7.1b) has 1.8 m in diameter. The maximum air speed in this tunnel is $v_{max} = 55$ m/s, which is unnecessary for our purposes. The VZLU/CVUT airplane model (see 8.2, Fig. 7.1a) used for testing of our algorithm has maximum construction speed at 30 m/s.

7.1.1 The airspeed

The first part of multi–hole pitots tube calibration was calibration of the airspeed of the aircraft. Raw output from differential pressure sensors is in Figure 7.2. In this experiment the airspeed in wind tunnel was increasing by 5 m/s per step until maximum value of 25 m/s.

From differential pressure sensor datasheet [21] we can adopt equation for conversion of output sensor voltage into dynamic pressure as follows

(1)

$$V_{out} = V_s \cdot (0.2 \cdot P + 0.5)$$

where V_s is input voltage, V_{out} is output voltage and P is differential pressure in [kPa]. After some adjustments in this computation we obtain final equation for dynamic pressure estimation ready for software implementation in on–board microcontroller

(2)

$$P_{dyn} = 5 \cdot (\frac{ADC_{out}}{4096} - c)$$

where P_{dyn} is dynamic pressure in [kPa], ADC_{out} is integer from 12 bit ADC converter (values between 0 and 4096), c is correction value $c = 0.565$. This correction value was obtained using data for airspeed $v_{air} = 0$ m/s to fulfil previous equation (dynamic pressure equals 0 Pa for 0 m/s).

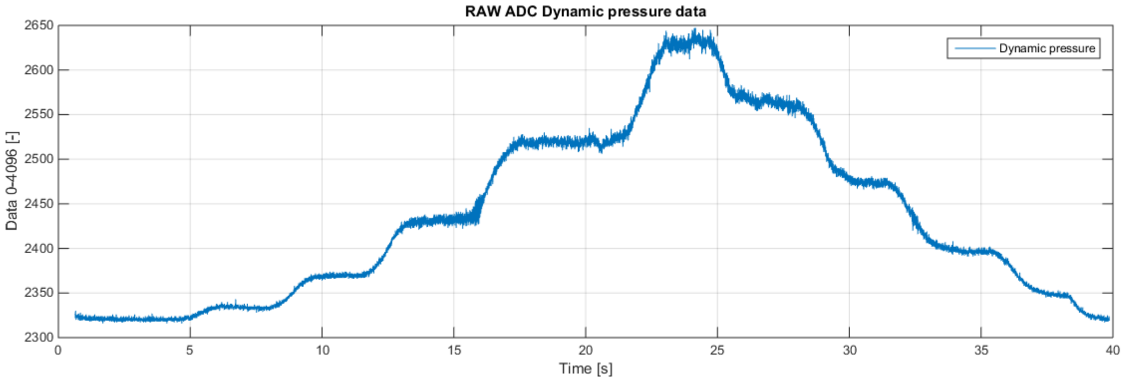


Figure 7.2. Dynamic pressure calibration. Raw differential pressure sensor data.

The dynamic pressure can be converted into airspeed using following equation

(3)

$$v_{air} = \sqrt{\frac{2 \cdot P_{dyn}}{\rho_{air}}}$$

where v_{air} is the airspeed, P_{dyn} is the dynamic pressure [Pa] and ρ_{air} is the air density.

The computed differential pressure is shown in Figure 7.3 and the final estimated airspeed is shown in Figure 7.4.

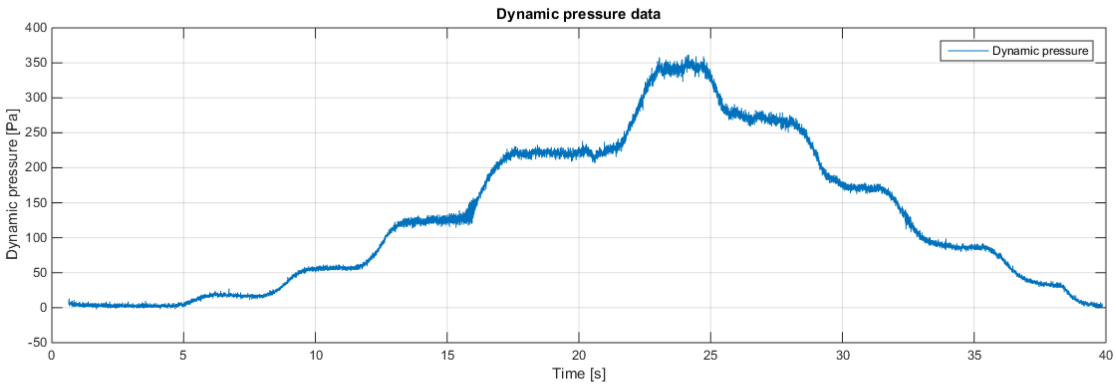


Figure 7.3. Dynamic pressure calibration. Computed dynamic pressure data.

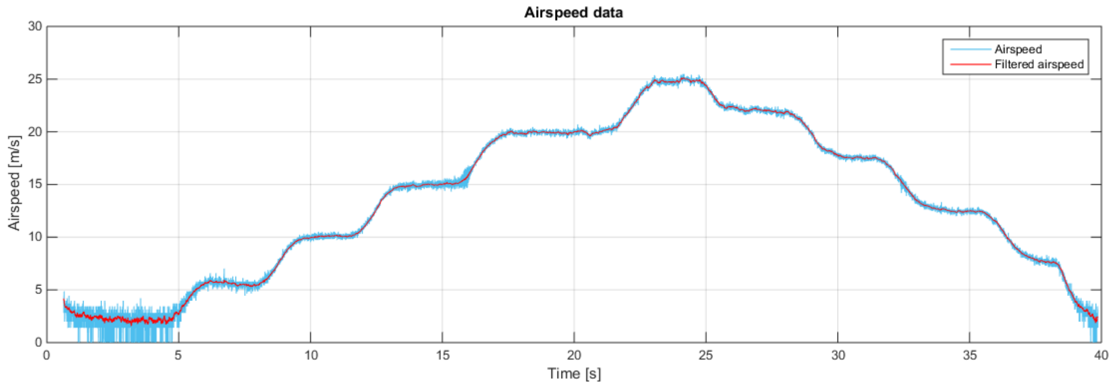


Figure 7.4. Dynamic pressure calibration. Computed airspeed and filtered airspeed.

In Figure 7.4 it is clearly visible, that airspeed under 5 m/s is not correctly computed. However, this error is not important to us, because the minimal airspeed of used VZLU/CVUT aircraft is around 10 m/s. During the software implementation we must pay attention to not compute square root from negative numbers — the on–board microcontroller software could fail and stop the computation when airspeed is zero to prevent division by zero.

Other values are properly calibrated even after simple low–pass filter implementation. Small overshoot between 6^{th} and 7^{th} second of measurement is created by the wind tunnel control system — the settling time to 5 m/s is too long and the next measurement (v_{air}= 10 m/s) was set too quickly by the operator.

7.1.2 Angle–of–attack and sideslip angle

Angle–of–attack and sideslip angle estimation are almost the same — only difference is in holes connected to the differential pressure sensor.

The aerodynamic angle can be computed using special coefficients — c_α and c_β. This coefficients can be computed as follows

(4)
$$c_\alpha = \frac{\Delta p_\alpha}{P_{dyn}} \qquad c_\beta = \frac{\Delta p_\beta}{P_{dyn}}$$

where Δp_α is differential pressure between upper and lower hole, Δp_β is differential pressure between left and right hole of multi–hole pitots tube, P_{dyn} is dynamic pressure.

The relation between α and c_α coefficient should be linear and invariant to change of airspeed. This relation can be found in 7.6a.

The calibration process is slightly different from calibration of the airspeed — we must set different angle–of–attack by tilting the mounting of the aircraft (Fig. 7.1a). However, the increase of the air speed through the wind tunnel remained the same (from 5 m/s to 25 m/s).

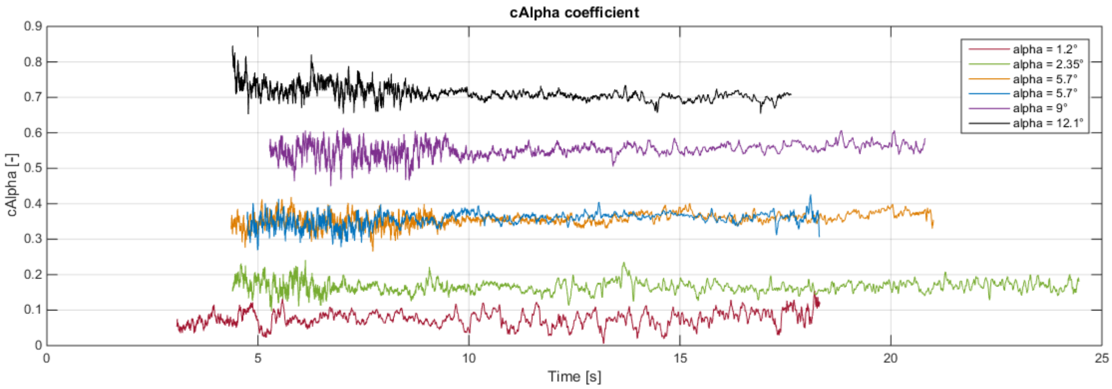


Figure 7.5. Angle–of–attack calibration. Measured c_α for different angle–of–attack α.

Computed values of c_α coefficient can be found in Figure 7.5. Several values between 0 a 5 seconds was not used — the dynamic pressure was close to 0 Pa so in the equation (4) was division by zero. It can be easily shown that after 10 seconds the values are less noisy due to increasing the airspeed and the dynamic pressure.

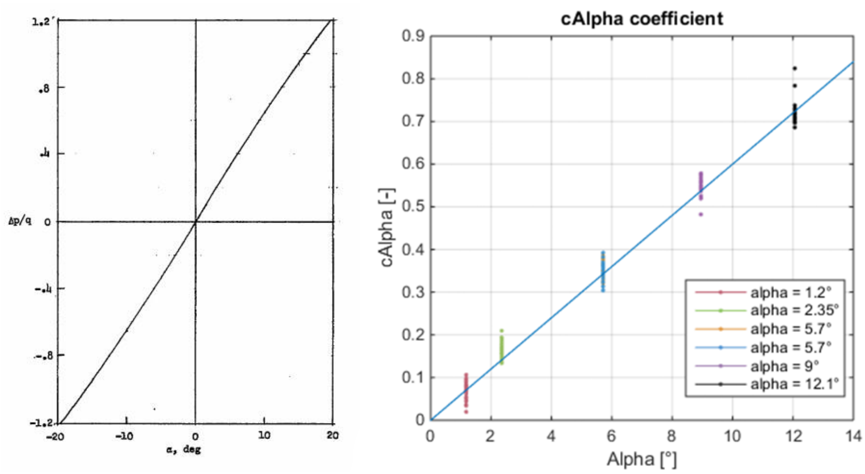


Figure 7.6. Angle–of–attack calibration. a) Theoretical relation between c_α and α [22], b) Measured relation between c_α coefficient and angle–of–attack α.

The values of c_α coefficient in relation with α are in Figure 7.6b. It can be easily shown, that the relation is nearly linear. There is big spread of values, it is probably due to noisiness of the differential pressure sensor and oscillations of the aircraft and multi–hole pitots tube in wind tunnel during bigger air speeds.

The computation of the beta coefficient c_β is the same as alpha coefficient c_α, but different holes are used for differential pressure – left and right. This computation is shown in figure 7.7. Data between 5 and 10 seconds are quite noisy due to slow air speed trough wind tunnel. When the air speed is over 10 m/s, the raw data from pitots tube are less noisy.

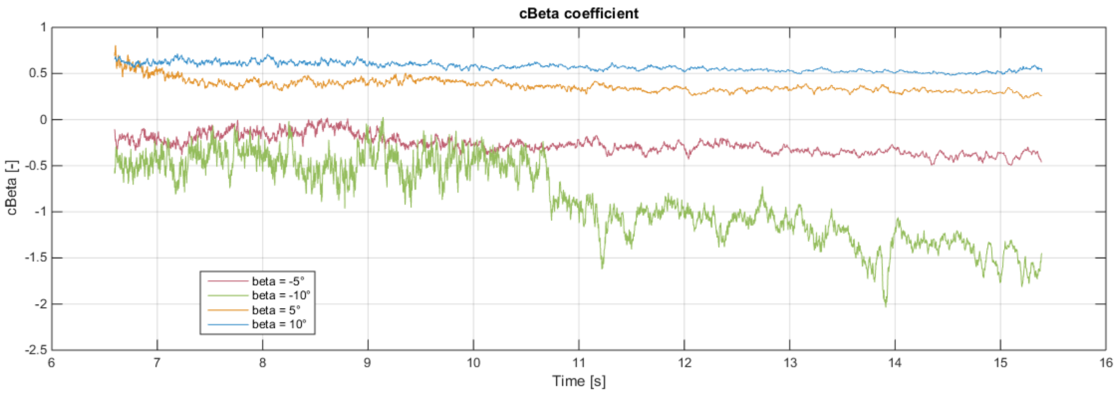


Figure 7.7. Sideslip angle calibration. Measured c_β for different sideslip angles β and air velocities.

The green line ($\beta = 10^o$) is very noisy and not correct, probably the unstable construction and mounting of the plane in wind tunnel was causing oscillations of the pitots tube. Hovewer, the measured values of sideslip angle should be between plus and minus 5 degrees.

The relation of computed values of beta coefficient c_β and sideslip angle β is given in Figure 7.8. These values are aligned with line from Figure 7.6a.

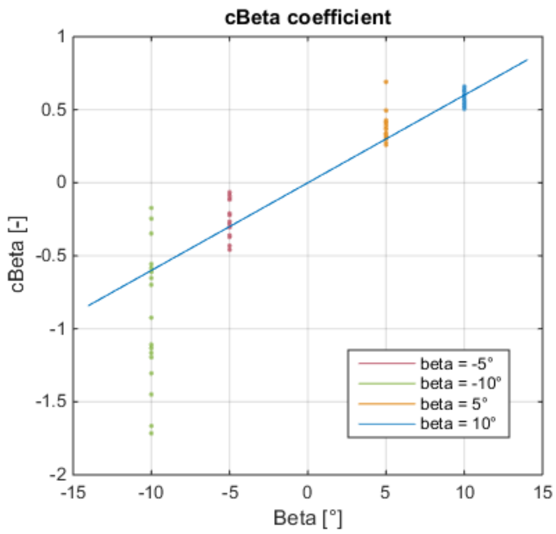


Figure 7.8. Sideslip angle calibration. Measured relation between c_β coefficient and sideslip angle β.

During the wind tunnel data processing the estimation of the airspeed of the aircraft is not working correctly for bigger aerodynamic angles — the estimated airspeed was lower than actual airspeed.

The reason of this phenomenon is that measured airspeed is in body reference system. The airspeed should be transformed into aerodynamic reference frame (see Sec. 3.1) using following equation

(5)

$$V_{air} = \frac{u}{\cos\alpha\cos\beta}$$

where α and β are aerodynamic angles, u is the measured airspeed and V_{air} is the resulting airspeed in aerodynamic reference frame.

Later research showed that this correction was not enough. The estimated airspeed was still lower than the actual airspeed. This phenomena can be caused by placement of the multi–hole pitots tube near the top of the UAV fuselage and near the elevator control surface.

Since the aerodynamic angles can be measured correctly, the airspeed can be corrected by some correction coefficient, which is dependent on actual value of angle–of–attack and sideslip angle — see Eq. (6).

(6)

$$V_{air} = \frac{u}{\cos\alpha\cos\beta} + f(\alpha,\beta)$$

7.2 Magnetometer calibration

Near metallic objects and even nearby power supply wires has influence on magnetic field surrounding magnetometer. Difference in Earth's magnetosphere, which is not constant everywhere on Earth, has big influence to. Due to this phenomena raw magnetometer data must be corrected, all readings from simple magnetometer should not be used without precise calibration.

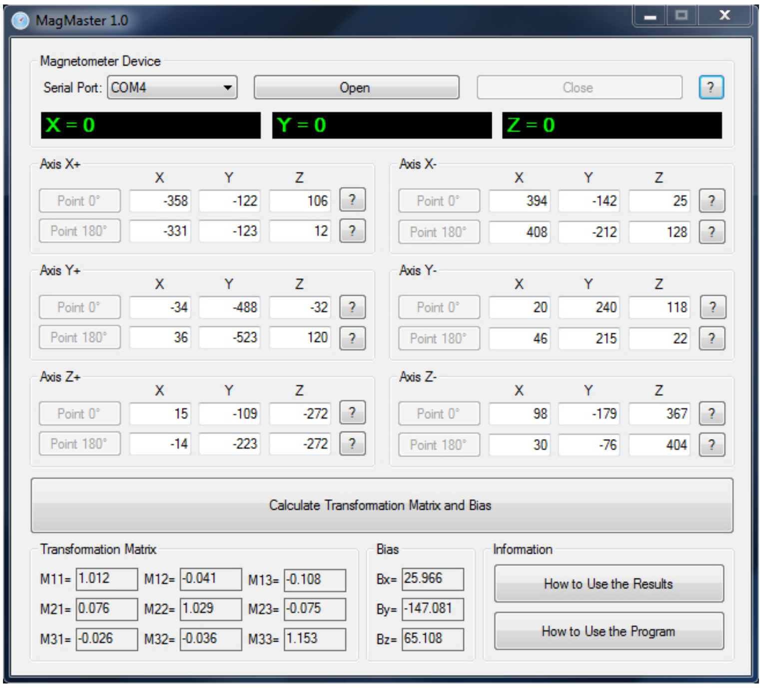


Figure 7.9. MagMaster - magnetometer calibration application.

The easiest and probably the simplest way to calibrate magnetometer is using calibration matrix and bias vector. Raw and uncalibrated data can be corrected using this mathematical expressions in following equation

(7)

$$\mathbf{M_{cal}} = \mathbf{T_m} \times (\mathbf{M_{raw}} - \mathbf{B_m})$$

where $\mathbf{M_{cal}}$ is vector of calibrated magnetometer data, $\mathbf{M_{raw}}$ is vector of raw measured magnetometer data, $\mathbf{T_m}$ is transformation matrix and $\mathbf{B_m}$ is bias vector.

The transformation matrix and the bias vector can be easily obtained using the MagMaster software (Fig. 7.9, [23]).

The final calibration matrices are defined as follows

(8)

$$\mathbf{T_m} = \begin{bmatrix} 1.012 & -0.041 & -0.108 \\ 0.076 & 1.029 & -0.075 \\ -0.026 & -0.036 & 1.153 \end{bmatrix} \qquad \mathbf{B_m} = \begin{bmatrix} 25.966 \\ -147.081 \\ 65.108 \end{bmatrix}$$

Finally the difference between uncalibrated (a) and calibrated (b) magnetometer data is shown in Fig. 7.10. This difference is clearly visible - uncalibrated magnetometer data are not formed in sphere, but in some directions this sphere is flattened. The calibrated data are formed correctly in sphere.

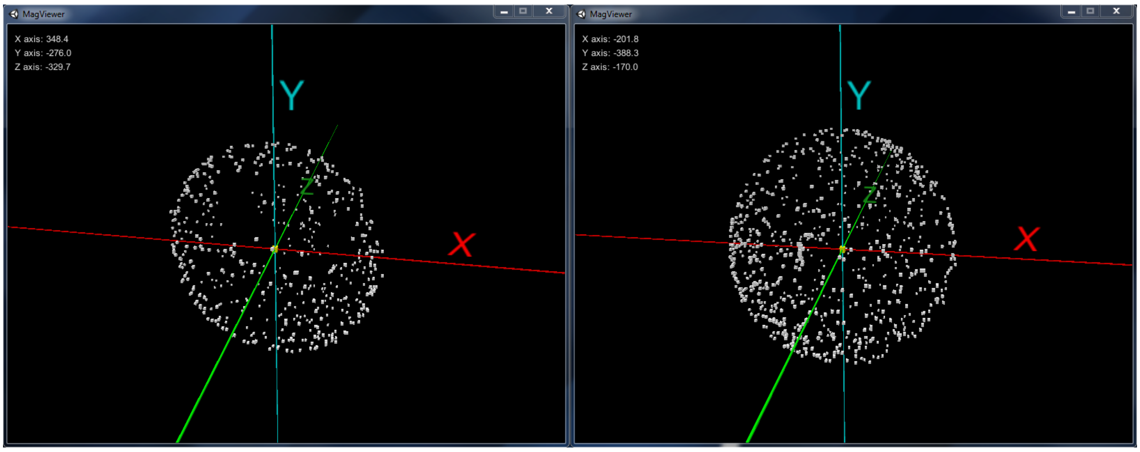


Figure 7.10. Magnetometer calibration a) uncalibrated measurements, b) calibrated measurements.

Chapter 8
Aircrafts for experiments

The on–board flight board was originally developed for RC model of Cessna 182 described in Section 8.1, but its modularity makes it possible to implement this control board into various vehicles – such as boats, cars or airships [3]. In the following section all used test planes are briefly described.

8.1 Cessna 182

The first test plane is commercially available EPP foam model of Cessna 182. This plane is very popular among the aircraft modellers due to easy manipulation with aircraft, very stable flight and good manoeuvrability.

Wing span	1410	mm
Wing area	27.5	dm^2
Length	1100	mm
Weight	1600	g
Material	EPP	foam

Table 8.1. Cessna 182 model description.

Several different experimental test results and flight data measured using on–board flight control board implemented in this plane can be found in [3] and [24].

Figure 8.1. RC model of Cessna 182.

8.2 VZLU/CVUT aircraft

The second aircraft was originally developed at Faculty of Mechanical Engineering (CTU in Prague) within Petr Adámek's Diploma thesis [25] with collaboration of Czech Aerospace Research and Test Establishment. This plane is also currently the main topic of Diploma thesis of Vojtěch Rubáš [15], which follows previous works on the plane.

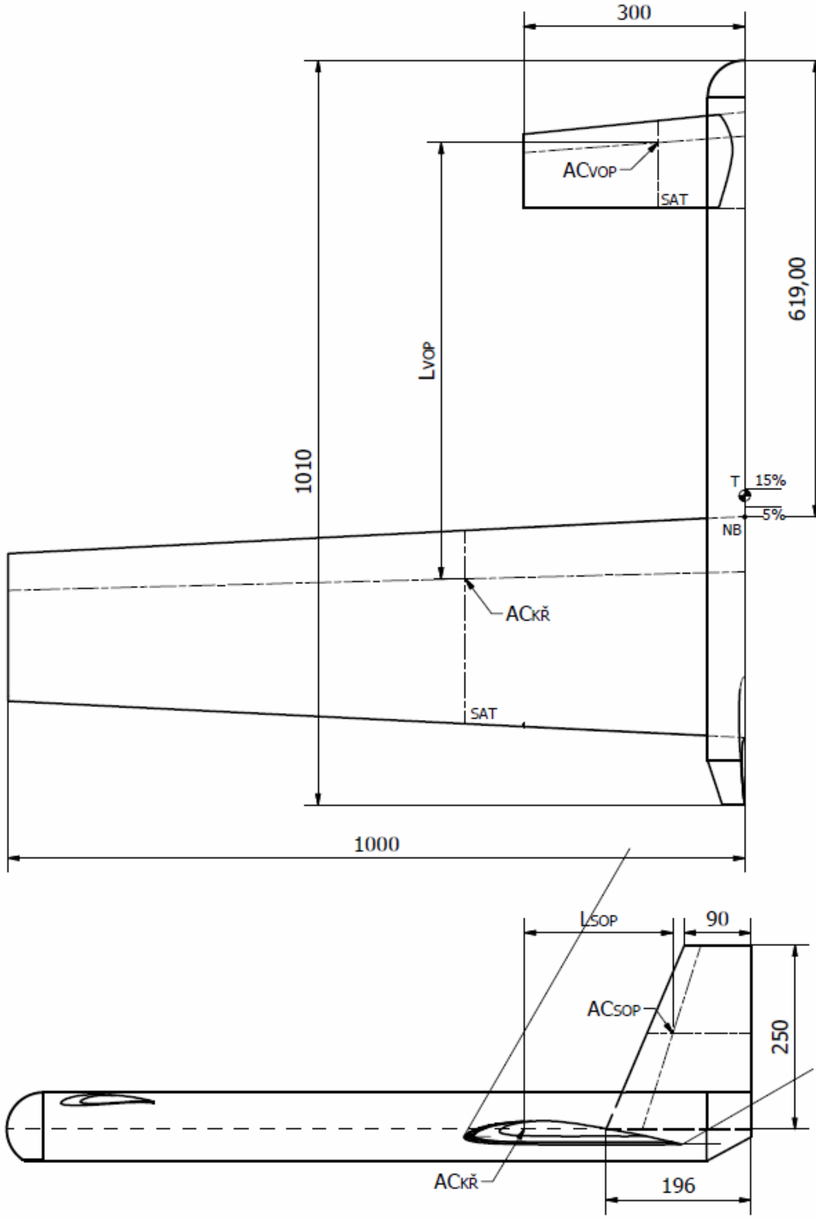


Figure 8.2. VZLU/CVUT aircraft drawing.

The part of this thesis was implementation of flight control board into VZLU/CVUT aircraft and technical help with recording flight data during flight performance tasks within measuring flight data [15]. This included precision calibration of the pitots tube in wind tunnel and the support with measuring of the moment of inertia of this plane. The pitots tube calibration was described in section 7.1 and some results from the moment of inertia measuring can be found in [15].

Wing span	2000	mm
Length	1010	mm
Weight	4500	g
Material	Wood	

Table 8.2. VZLU/CVUT aircraft description.

Figure 8.3. VZLU/CVUT aircraft model.

The external on–board camera can be implemented in the transparent front of this UAV. The camera can be used only to extend the informations about the aircraft by the view from the cockpit or for example some algorithms for tracking ground object can be developed. These ideas might be used as another thesis theme in the future.

Chapter 9
Summary of fulfilled goals

Most of the goals of this thesis given in Chapter 2 were fulfilled. Summary of these goals with references is listed as follows.

- **Wind estimation algorithm**

  - The computational algorithm for wind prediction was developed and tested within several realistic simulation scenarios in Section 6.1 and 6.3. Then the algorithm measurement sensitivity to sensor error was researched in Section 6.2.

- **Software implementation**

  - The developed computational algorithm was written in C programming language. This estimation algorithm was then implemented into existing on–board software and then it was extended by several new functionalities of newly added hardware (e.g. magnetometer — Section 5.1.2). Several samples of source code were given in Chapter 5.

- **On-board hardware**

  - The on–board flight board developed at DCE FEE CTU was used to compute, measure and log all flight variables (see Section 4.1). Related hardware for multi–hole pitots tube was introduced in Section 4.2 and results of several experimental tests and calibration in wind tunnel were given in Section 7.1. Finally external magnetometer was implemented into on–board system and results of related measurement calibration were given in Section 7.2.

- **Additional activities**

  - All electrical components in VZLU/CVUT aircraft were reviewed and updated. The multi–hole pitots tube with differential pressure measurement board was implemented into this aircraft (see Section 8.2). This aircraft was then completed and prepared for experimental flight test.

Unfortunately the experimental field tests have not been performed till this moment. They are planned for Summer/Autumn 2015 as one of the main near–future objectives for the follow–up phases of this project.

Upon agreement with the Bachelor's thesis supervisor, I intend to pursue this research in form of the Semestral project and Diploma thesis in 2016 and 2017.

Chapter 10
Publication outputs

Two conference papers on selected results presented in this thesis have been published.

- **On–line on–board wind estimation system for small UAVs**
  - Authors: Martin Mondek, Martin Hromčík
  - 20th International Conference on Process Control
  - 9.-12. 6. 2015, Štrbské Pleso, Slovakia

This paper [26] described the status, achieved results and plans related to the wind estimation system. It introduced the theory of wind speed and direction computation and introduced simulation results of this computation. Then the sensitivity to measurement errors was carefully analysed in this work. Finally, performance of the algorithms is demonstrated in realistic simulation scenarios.

This report was was presented as a lecture–paper at 20th International Conference on Process Control held in Štrbské Pleso in Slovakia in June 2015.

- **Onboard wind prediction system**
  - Author: Martin Mondek
  - AIAA Pegasus student conference 2015
  - 20.-22. 4. 2015, Salon–de–Provence, France

In this paper [27] basics of the wind estimation system and several special software solutions (e.g. MAVLink communication protocol) were introduced. However, mainly the hardware and sensor part of wind estimation system was introduced and described. The function of special differential pressure measurement board was tested in wind tunnel with professional multi–hole pitots tube.

This report was was presented as a conference paper at student conference Pegasus 2015 organized annually by the world's most prestigious aerospace organization The American Institute of Aeronautics and Astronautics. This paper passed successfully the internal Czech Technical University competition: three papers out of six candidates, authored by CTU BSc., MSc. and PhD. level students, were selected to represent CTU at this prestigious event.

Chapter 11
Conclusion and future work

The main topic of this thesis — the wind prediction algorithm — was introduced in this work. The realistic simulations showed that this algorithm works for all flight scenarios correctly and unlike in other publications even during difficult flight performances. The measurement error analysis provides fundamental information for future implementation of advanced filtering methods or control structures.

The implementation of different filtering methods — for example Kalman filtering, sensor fusion or implementation of mathematical model of the VZLU/CTU aircraft directly to the on–board software — can be the main objective of the following work. As showed in section dedicated to multi–hole pitots tube calibration, the measurement of aerodynamic angles is quite noisy. Several mechanical improvements could be developed (such as better and stronger mounting to the aircraft), but the implementation of Kalman filters or other advanced filtering solutions should bring biggest improvement in aerodynamic angles measurement and the whole wind estimation. The aerodynamic angles measurement can be further improved by developing different pressure measurement board for multi–hole pitots tube.

Several improvements upon existing hardware and software solutions were made to fulfil needs of this project. The following projects can implement flight control system in various different applications. It is recommended to develop simple mechanical box to improve mechanical robustness of this solution.

Finally the informations about wind computed by the algorithm developed in this work can improve existing control algorithms — such as AgentFly — multiagent flight system developed at Czech Technical University. Another suitable usage of this algorithm is for example during collection of flight measurement data for comparing realistic behaviour of the aircraft with simulations of developed mathematical models of the aircraft.

Appendix A
The thesis assignment

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Martin Mondek**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Návrh a realizace modulárního řídicího systému pro malé UAV**

Pokyny pro vypracování:

Navrhněte a realizujete řídicí systém pro malé UAV z Ústavu letadlové techniky na fakultě strojní, s využitím zakoupené modulární HW soustavy.

1. Implementujte čidla řídicího systému do UAV.
2. Otestujte funkce čidel a jejich měření.
3. Implementujte vybrané algoritmy řízení.

Seznam odborné literatury:

Stevens, Lewis, Aircraft simulation and control, Prentice Hall, 2005

Vedoucí: doc. Ing. Martin Hromčík, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 20. 10. 2014

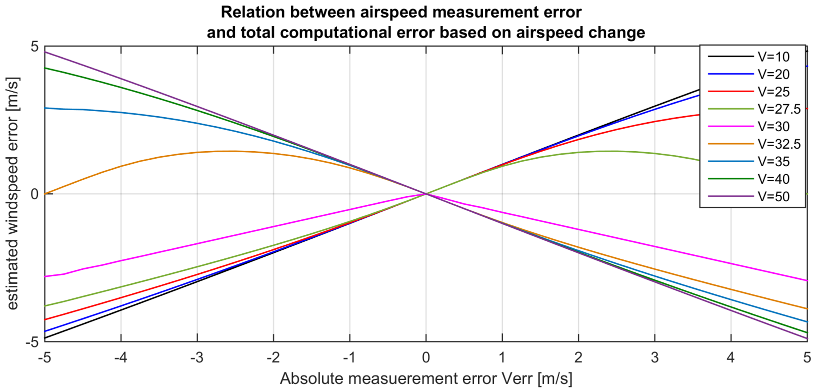
Appendix B
The list of abbreviations

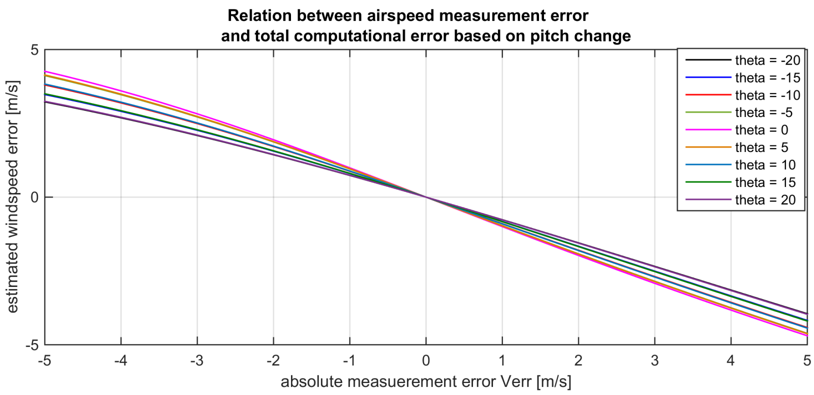
Abbreviation	Explanation
DCE FEE CTU	Department of Control Engineering Faculty of Electrical Engineering Czech Technical University in Prague
UAV	unmanned aerial vehicle
MAVLink	Micro Air Vehicle Link
RC	Radio Controlled
PWM	Pulse–Width Modulation
IMU	Inertial Measurement Unit
GPS	Global Positioning System
ADC	Analog–to–Digital Converter
USART	Universal Synchronous Asynchronous Receiver and Transmitter
SPI	Serial Peripheral Interface
SWD	Serial Wire Debug
IDE	Integrated development environment

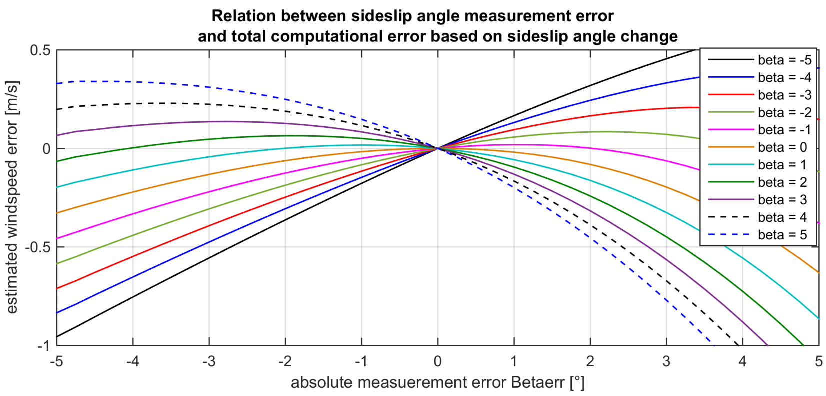
Appendix C
The content of attached CD

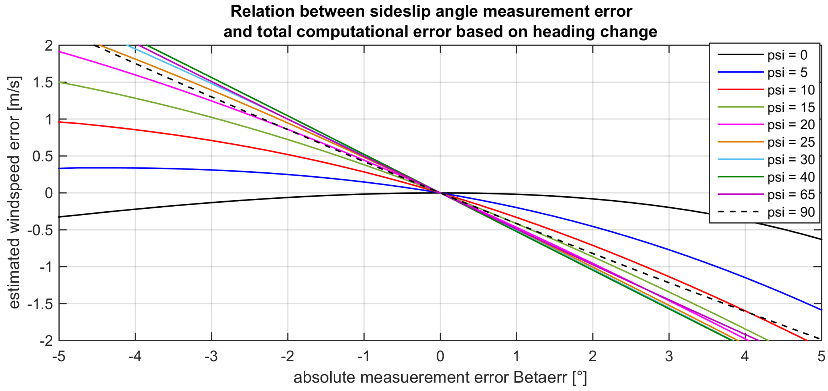
- **Bachelors-Thesis-Mondek-2015.pdf** — electronical version of this work
  - **Measured data** — the calibration data with relevant Matlab scripts
  - **Source code** — projects for CoCoox IDE including source codes in C
    - **Ground station** — software of the ground station
    - **On-board software** — software of the on–board flight control board

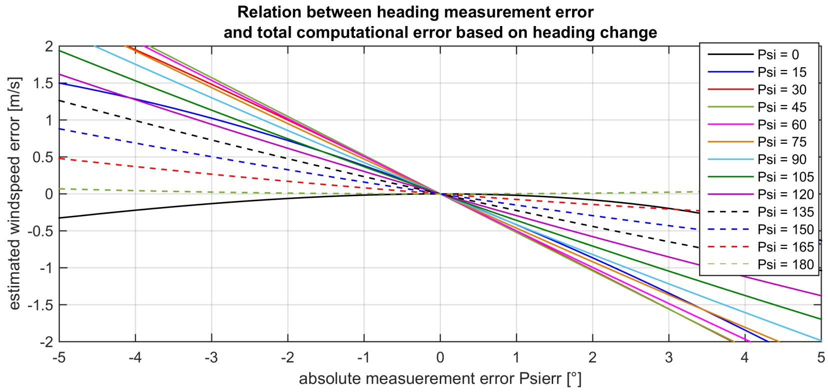
Appendix D
Selection of error measurement relations

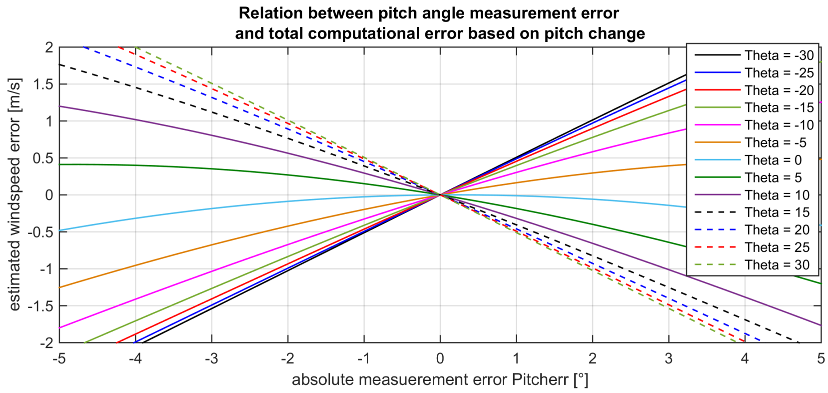


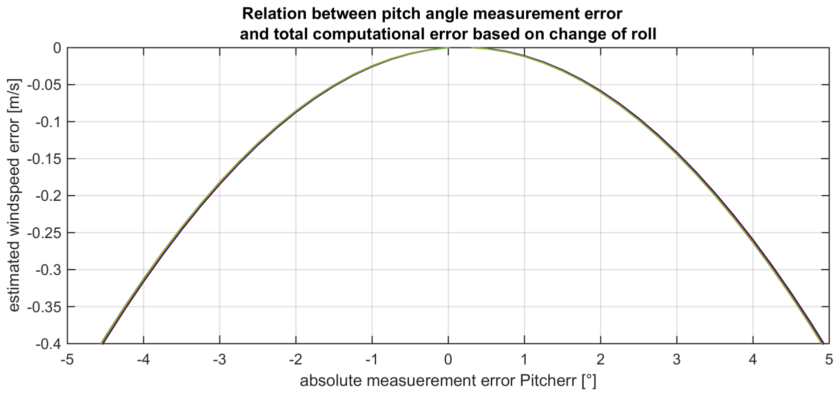


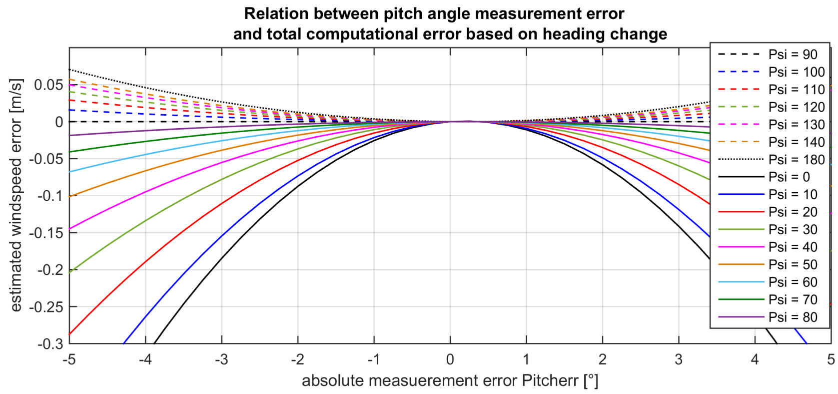


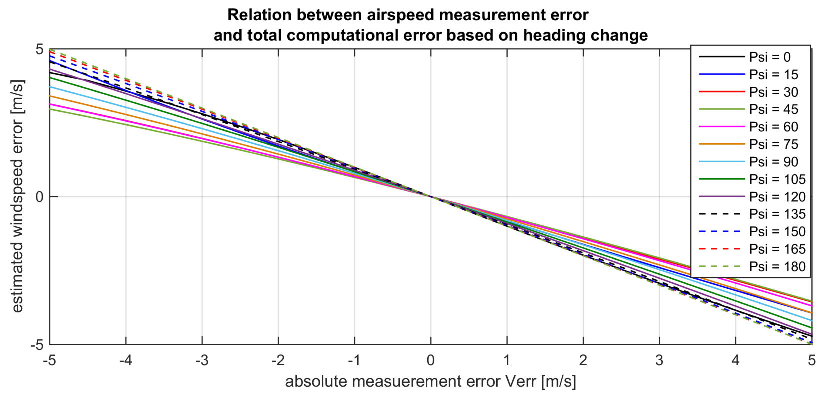

Relation between sideslip angle measurement error
and total computational error based on heading change
estimated windspeed error [m/s]
absolute measuerement error Betaerr [°]
psi = 0
psi = 5
psi = 10
psi = 15
psi = 20
psi = 25
psi = 30
psi = 40
psi = 65
psi = 90


Relation between heading measurement error
and total computational error based on heading change
estimated windspeed error [m/s]
absolute measuerement error Psierr [°]
Psi = 0
Psi = 15
Psi = 30
Psi = 45
Psi = 60
Psi = 75
Psi = 90
Psi = 105
Psi = 120
Psi = 135
Psi = 150
Psi = 165
Psi = 180


Relation between pitch angle measurement error
and total computational error based on pitch change
estimated windspeed error [m/s]
absolute measuerement error Pitcherr [°]
Theta = -30
Theta = -25
Theta = -20
Theta = -15
Theta = -10
Theta = -5
Theta = 0
Theta = 5
Theta = 10
Theta = 15
Theta = 20
Theta = 25
Theta = 30


Relation between pitch angle measurement error
and total computational error based on change of roll
estimated windspeed error [m/s]
absolute measuerement error Pitcherr [°]


Relation between pitch angle measurement error
and total computational error based on heading change
estimated windspeed error [m/s]
absolute measuerement error Pitcherr [°]
Psi = 90
Psi = 100
Psi = 110
Psi = 120
Psi = 130
Psi = 140
Psi = 180
Psi = 0
Psi = 10
Psi = 20
Psi = 30
Psi = 40
Psi = 50
Psi = 60
Psi = 70
Psi = 80


Relation between airspeed measurement error
and total computational error based on heading change
estimated windspeed error [m/s]
absolute measuerement error Verr [m/s]
Psi = 0
Psi = 15
Psi = 30
Psi = 45
Psi = 60
Psi = 75
Psi = 90
Psi = 105
Psi = 120
Psi = 135
Psi = 150
Psi = 165
Psi = 180

Appendix E
References

[1] Amazon.com Inc. *Amazon Prime Air (online)*. (Accessed 22-05-2015).
http://www.amazon.com/b?node=8037720011.

[2] Tanya Lewis. *'Project Wing': Google Unveils New Drone-Delivery System (online)*. (Accessed 22-05-2015).
http://www.livescience.com/47617-google-drone-delivery-project.html.

[3] Jaroslav Halgašík. *Flight Control System Unit for Small UAV. Diploma thesis*. Faculty of Electrical Engineering, Czech Technical Univerzity in Prague, 2014.

[4] Agent Technology Center. *AgentFly (online)*. (Accessed 22-05-2015).
http://agents.felk.cvut.cz/projects.

[5] Petr Pahorecký. *Autopilot pro UAV letoun. Diplomová práce*. Fakulta Elektrotechnická, České vysoké učení technické v Praze, 2015.

[6] ČSN 01 6910. *Úprava písemností zpracovaných textovými editory*. 2007.
http://typotypo.wz.cz/csn016910.pdf.

[7] Josef Novák. *Sestavení matematického modelu a návrh řídících algoritmů pro UAV. Diplomová práce*. Fakulta Elektrotechnická, České vysoké učení technické v Praze, 2015.

[8] LLC Unmanned Dynamics. *AeroSim Blockset User's Guide*. 2002.
http://people.rit.edu/pnveme/EMEM682n/Matlab_682/aerosim_ug.pdf.

[9] *Ardupilot (online)*. (Accessed 22-05-2015).
http://code.google.com/p/ardupilot-mega/wiki/Introduction.

[10] *Openpilot (online)*. (Accessed 22-05-2015).
http://www.openpilot.org/products/openpilot-Revolution-platform.

[11] STMicroelectronics. *STM32F100RB datasheet*. (Accessed 22-05-2015).
http://www.st.com/web/catalog/mmc/FM141/SC1169/SS1031/LN775/PF216844?sc=internet/mcu/product/216844.jsp.

[12] Maestro Wireless Solutions Limited. *GPS Receiver A2035-H (online)*. (Accessed 22-05-2015).
http://www.farnell.com/datasheets/1794015.pdf.

[13] STMicroelectronics. *LSM330DL (online)*. (Accessed 22-05-2015).
http://www.st.com/web/catalog/sense_power/FM89/SC1448/PF253022.

[14] Honeywell. *HMC5883L (online)*. (Accessed 22-05-2015).
www51.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/HMC5883L_3-Axis_Digital_Compass_IC.pdf.

[15] Vojtěch Rubáš. *Identifikace vlastností letadla z letového měření. Diplomová práce*. Ústav letadlové techniky Fakulty Strojní, České vysoké učení technické v Praze, 2015.

[16] Henning Karlsen. *Library: LCD5110_Graph (online)*. (Accessed 22-05-2015).
http://www.rinkydinkelectronics.com/library.php?id=48.

[17] Creative commons (CC BY–NC–SA3.0). *Attribution–NonCommercial–ShareAlike 3.0 Unported (online).* (Accessed 22-05-2015).
http://creativecommons.org/licenses/by-nc-sa/3.0/.

[18] DroidPlanner. *Ground Control Station for Android Devices (online).* (Accessed 22-05-2015).
https://github.com/DroidPlanner/Tower.

[19] *QGroundControl (online).* (Accessed 22-05-2015).
http://qgroundcontrol.org.

[20] J. Langelaan, J. Neidhoefer, and N. Alley. *Wind Field Estimation for Small Unmanned Aerial Vehicles.* Journal of Guidance, Control, and Dynamics, 34(4), pp.1016-1030, 2011.

[21] Freescale Semiconductor Inc. *MPXV7002 Integrated Silicon Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated (online).* (Accessed 22-05-2015).
http://www.farnell.com/datasheets/1888680.pdf.

[22] SpaceAgeControl Inc. *Technical Reference Publication (online).* (Accessed 22-05-2015).
http://www.spaceagecontrol.com/naca-tn-4351.pdf.

[23] Yury Mastelenak. *Advanced hard and soft iron magnetometer calibration for dummies.* (Accessed 22-05-2015).
http://diydrones.com/profile/YuryMatselenak.

[24] Jan Kment. *Návrh pokročilých systémů řízení pro malý RC letoun.* Fakulta Elektrotechnická, České vysoké učení technické v Praze, 2015.

[25] Petr Adámek. *Návrh bezpilotního letounu. Diplomová práce.* Ústav letadlové techniky Fakulty Strojní, České vysoké učení technické v Wind Field Estimation for Small Unmanned Aerial Vehicles.Praze, 2014.

[26] Martin Mondek, and Martin Hromčík. *On-line on-board wind estimation system for small UAVs.* 20th International Conference on Process Controll, Štrbské pleso, Slovakia, 2015.

[27] Martin Mondek. *Onboard wind prediction system.* AIAA Student conference 2015, Salon–de–Provence, France, 2015.