Master Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Measurement

Aerometric System for Unmanned Aerial Vehicles

Bc. Tomáš Zelinka

Supervisor: Ing. Martin Šipoš, Ph.D.
Field of study: Cybernetics and Robotics - Aerospace Systems
May 2022

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Zelinka Tomáš**	Personal ID number:	**495798**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Measurement**		
Study program:	**Cybernetics and Robotics**		
Branch of study:	**Aerospace Systems**		

II. Master’s thesis details

Master’s thesis title in English:

Aerometric System for Unmanned Aerial Vehicles

Master’s thesis title in Czech:

Aerometrický systém pro bezpilotní prostředky

Guidelines:

Design and realize a system for measuring barometric altitude, airspeed (TAS, CAS), vertical speed, sideslip angle and angle of attack. The system will consist of modules that will digitize and pre-process data from differential and absolute pressure sensors. The data will then be sent to the central unit, which will also provide synchronization (approx. 1 Hz). Communication will take done via the CAN bus with the CANaerospace format. The design of the entire system will include analysis of the appropriate number and location of probes for measuring the required pressures (probe distance/accuracy of determination of sideslip angle), including the design of a modified probe shape for measuring the angle of attack and sideslip angle, calibration of pressure sensors, sensitivity analysis and testing in real conditions (car, UAV experiments). The implementation will therefore include both HW design of the system and SW implementation, data processing using advanced methods (filtering, data fusion, etc.) and their evaluation.

Bibliography / sources:

[1] Ly, Jack Kevin: "ANGLE OF ATTACK DETERMINATION USING INERTIAL NAVIGATION SYSTEM DATA FROM FLIGHT TESTS. " Master's Thesis, University of Tennessee, 2017. https://trace.tennessee.edu/utk_gradthes/4757
[2] R.P.G. Collinson: "Introduction to Avionics Systems", Third Edition, Springer Dordrecht Heidelberg London New York , DOI 10.1007/978-94-007-0708-5, e-ISBN 978-94-007-0708-5 ISBN 978-94-007-0707-8, 2011.
[3] M. Soták, M. Sopata, R. Bréda, J. Roháč a L. Váci: "Integrácia navigačných systémov", Košice: Bréda Róbert, 2006.

Name and workplace of master’s thesis supervisor:

Ing. Martin Šipoš, Ph.D. katedra měření FEL (13138)

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **22.01.2021** Deadline for master's thesis submission: **20.05.2022**

Assignment valid until:
by the end of summer semester 2022/2023

Ing. Martin Šipoš, Ph.D.	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature		Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

I would like to thank the thesis supervisor Ing. Martin Šipoš, Ph.D. for his advice, discussion, recommendations and cooperation during all testing procedures. I am also thankful to Ing. Ivana Beshajová Pelikánová, Ph.D. for assistance with soldering during board assembly and Ing. Jakub Suchý for supporting during wind tunnel testing.

Declaration

"Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky použité informačné zdroje v súlade s Metodickým pokynom o dodržiavaní etických princípov pri príprave vysokoškolských záverečných prác."

V Prahe, máj 2022

“I declare that this work is all my own work and I have cited all sources I have used in the bibliography in accordance with the Methodological Instruction on observance of ethical principles in the preparation of university theses.”

Prague, May, 2022

. . . .

Abstract

The main goal of master thesis is design and practical realization of the aerometric system for unmanned aerial vehicle with fixed wings. Developed system consists of two measuring modules with pressure transducers and main module that provides data synchronization and communication.

The theoretical part of the thesis includes the general description of the aerometric parameters and ways how to determine them.

There are also described hardware design of printed circuit boards and parts selection, mainly pressure transducers. Section dedicated to software implementation presents algorithms and software resources used in modules. Software of each module includes real-time operating system FreeRTOS. Another part of thesis is dedicated to communication busses and protocols used in modules. System provides aerometric data using CAN bus with CANaerospace frame.

Special important part of the thesis discusses testing of realized system, especially pressure transducers. Described tests also include calibration procedures of transducers, environmental testing with focus on temperature and testing in aerodynamic tunnel. The last mentioned test approach simulates the real condition test with experimental analysis of the pitot-static probes with focus on angle of sideslip measurement. There is also discussed state of art of determination of the aerodynamic angles using pitot-static tube.

Keywords: embedded system, aerometric system, unmanned aerial vehicle, CAN bus, GPS, GNSS, aerodynamic tunnel, aircraft, angle of attack, angle of sideslip, CANaerospace protocol

Supervisor: Ing. Martin Šipoš, Ph.D.

Abstrakt

Hlavním cílem diplomové práce je návrh a praktická realizace aerometrického systému pro bezposádkový prostředek s pevnými křídly. Vyvinutý systém se skládá z dvou měřících modulů se senzory tlaku a hlavního modulu, který poskytuje synchronizaci dat a komunikaci.

Teoretická část práce zahrnuje všeobecný popis aerometrických parametrů a způsobů jak je měřit.

Také je popsaný hardwarový návrh desek plošných spojů a výběr komponent, především senzorů tlaku. Část věnovaná softwarové implementaci popisuje algoritmy a softwarové prostředky využité v modulech. Software každého modulu zahrnuje operační systém reálného času FreeRTOS. Další část práce je věnovaná komunikačním sběrnicím a protokolům využitých v modulech. Systém poskytuje aerometrická data prostřednictvím CAN sběrnice s rámcem CANaerospace.

Zvláštní důležitá část práce popisuje testování zhotoveného systému, především senzorů tlaku. Popsané testy zahrnují kalibraci senzorů, klimatické testování se zaměřením na teplotu a testování v aerodynamickém tunelu. Poslední zmíněný postup simuluje test v reálných podmínkách s experimentální analýzou pitot-statických trubic se zaměřením na měření úhlu skluzu. Taktéž jsou popsané současné trendy v odvození úhlu náběhu pomocí pitot-statické trubice.

Klíčová slova: embedded system, aerometrický systém, bezpilotní prostředek, CAN sběrnice, GPS, GNSS, aerodynamický tunel, letadlo, úhel náběhu, úhel vybočení, CANaerospace protokol

Překlad názvu: Aerometrický systém pro bezpilotní prostředky

Contents

Figures

Tables

Chapter 1
Introduction

Nowadays, the aerometric system is a required component for any modern aircraft. This "black-box" system is necessary for safe flights. It provides the aerometric data essential for pilots (or operators) during all flight phases as barometric altitude or true airspeed. Also, these parameters are inputs for other aircraft systems - flight control systems or guidance systems. However, in the case of unmanned aerial vehicles, it is not the standard equipment. These types of aircraft for flight data acquisition can use received GNSS data in combination with an inertial measurement unit. Sometimes, small lightweight aircraft for short-range flights do not use any system.

Like other types of electronic systems, aerometric systems had been developing over decades. The first aerometric systems were based on the simple mechanical principle of instruments with direct reading. Later were used analogue circuits, and nowadays, aerometric digital modular units with digital signal processing and many other functionalities are standard. The current trend is to make these systems smaller and lighter, but another crucial part is the software implementation regarding the optimal operation, functionality, safety and security.

Modern small-sized commonly available microcontrollers with many peripherals are a valuable solution for small unmanned aircraft. It means that the pressure transducers can be used for direct sensing of aerometric parameters and microcontrollers as an aerometric system for real-time data processing and further communication. This approach was used to design and realize the aerometric system for unmanned aircraft. This realization included hardware and software development. Major selected components - pressure transducers - were tested in the Aeronautical Systems and Instrumentation laboratory (Department of Measurement, The Faculty of Electrical Engineering, CTU) for accuracy and temperature properties. Finally, the whole system was tested in realistic conditions in the aerodynamic tunnel. Another experimental analysis of the selected pitot-static probes regarding the determination of aerodynamic angles was carried out.

1.1 State of the Art

There are already many available aerometric systems for small unmanned aircraft as market products, e.g., the Sensor for airspeed measurement ASS-100 ADV. This system provides measuring of airspeed in range $0\,\mathrm{km\,h^{-1}}$ to $360\,\mathrm{km\,h^{-1}}$ [24].

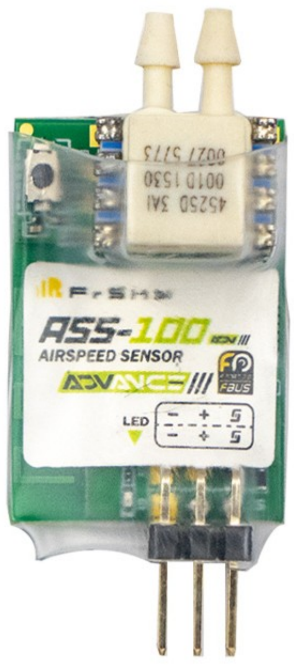


Figure 1.1: Sensor for airspeed measurement ASS-100 ADV [24].

The another digital sensor system MNAV100CA [19] is designed also for air robotic vehicle navigation and control. It includes the following measurement devices: GPS receiver module, 3-axis accelerometers, 3-axis angular rate sensors, 3-axis magnetometers, but mainly absolute pressure sensor and differential pressure sensor. For communication with system is used RS-232 bus.

Chapter 2
Measurement of Air Data Parameters

In general, aerometric system, as significat part of aircraft avionics, provides accurate information on air data parameters such as barometric (pressure) altitude H, indicated airspeed V_{IAS}, true airspeed V_{TAS}, calibrated airspeed V_{CAS}, vertical speed VS, Mach number M, static air temperature T_S and total air temperature T_{TAT}. These quantities are derived from sensed parameters including static pressure p_h, total pressure p_t and static temperature T_s.

The mentioned air data parameters are important for safe flight, flight control and they are shown in real-time on flight instruments to the pilot. However, H and V_{TAS} are essential for the flying of any aircraft. Barometric altitude is the actual height of the aircraft above sea level and accurate measurement of the H is fundamental for the piloting in the vertical plane. V_{TAS} is the speed of the aircraft relative to the air mass through which is actually moving. The aerodynamic lift force L_w [2.1], which is generated by the motion of the aircraft through the air and is acting on the airfoil, is a function of the V_{TAS} (dynamic pressure), the lift coefficient C_l, surface area of the wing S and air density at given level ρ_h.

$$L_w = \frac{1}{2}\,\rho_h\,V_{TAS}^2\,S\,C_l\,[N] \tag{2.1}$$

The aircraft service capabilities and limits of a design and construction refer to flight (performance) envelope. This operating limits are in term of a airspeed and altitude [Figure 2.1].

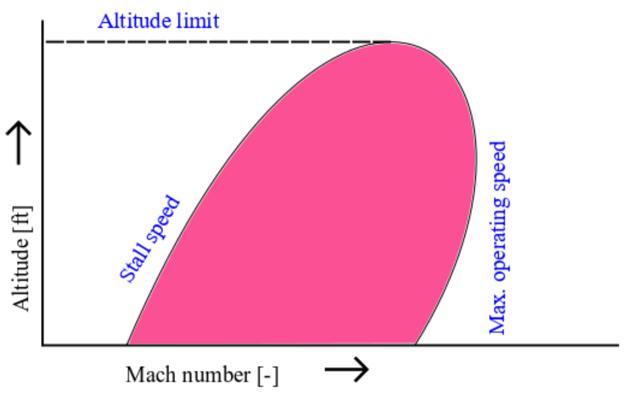


Figure 2.1: Flight envelope that defines stall and maximum operating speed.

2.1 Principles and Methods of Measuring Air Data Parameters

Measurement of air data parameters is based on the principle of sensing barometric parameters, because atmospheric pressure, air temperature and density vary with the distance above sea level. These changes are standardized by the International Standard Atmosphere (ISA) [22] and this property is used for measurement of pressure altitude. Related changes of static pressure p_h (local atmospheric pressure) in altitude [figure 2.2] above mean sea level (MSL) H are described by equation [2.2].

$$p_h = p_0 \left(1 \; + \; \frac{\tau}{T_0} H\right)^{\frac{g_0}{R\tau}} \; [Pa] \tag{2.2}$$

p_0 - Standart static pressure in MSL $= 101\,325\,\mathrm{Pa}$; T_0 - Standard temperature in MSL $= 288.15\,\mathrm{K}$; R - Universal gas constant for air $= 287.039\,\mathrm{J\,kg^{-1}\,K^{-1}}$; τ - Standard temperature lapse rate $= -0.0065\,\mathrm{K\,m^{-1}}$; g_0 - gravitational acceleration at sea level $= 9.806\,\mathrm{m\,s^{-2}}$;

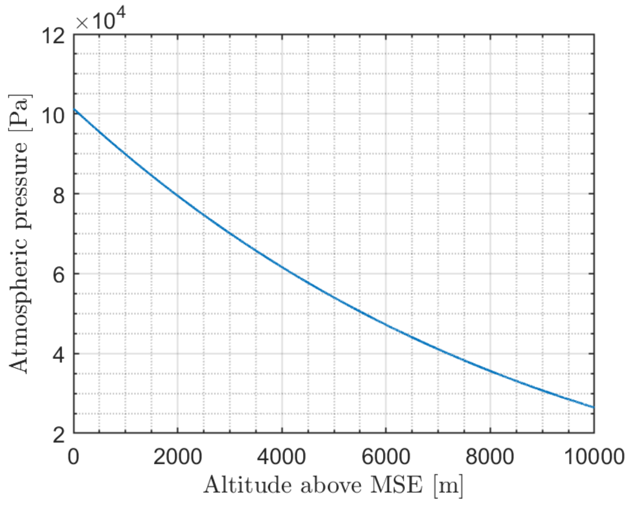


Figure 2.2: Atmospheric pressure - International Standard Atmosphere.

Therefore vertical speed VS is derived from the rate of change of static pressure [2.3].

$$VS = -\frac{RT_s}{g_0 \; p_h} \; \Delta p_h = \Delta H \; [m/s] \tag{2.3}$$

Following air data parameter, total pressure p_t, is measured facing the moving airstream using Pitot tube. This measures the impact pressure Q_c, which is function of the V_{TAS}, plus the static pressure p_h [2.4].

$$p_t = Q_c \; + \; p_h \; [Pa] \tag{2.4}$$

Dynamic pressure is described by Bernoulli's law [2.5].

$$p_t - p_h = \frac{1}{2}\rho_h V_{TAS}^2 = Q_c\,[Pa] \tag{2.5}$$

Assuming incompressible conditions, the relationships between V_{TAS}, V_{CAS} and dynamic pressure Q_c are [2.6] and [2.7].

$$V_{TAS} = \sqrt{\frac{2\kappa}{\kappa - 1} R T_0 \left(\frac{p_h}{p_0}\right)^2 \left[\left(\frac{Q_c}{p_h} + 1\right)^{\frac{\kappa-1}{\kappa}} - 1\right]}\,[m/s] \tag{2.6}$$

κ - Specific heat ratio for air $= 1.4$;

$$V_{CAS} = \sqrt{\frac{2\kappa}{\kappa - 1} \frac{Q_c}{\rho_0} \left[\left(\frac{Q_c}{p_0} + 1\right)^{\frac{\kappa-1}{\kappa}} - 1\right]}\,[m/s] \tag{2.7}$$

ρ_0 - Standard air density at MSL $= 1.225\,\mathrm{kg\,m^{-3}}$;
In general Mach number is equal to [2.8] and for $M \leq 1$ is [2.9].

$$M = \frac{V_{TAS}}{a}\,[-] \tag{2.8}$$

a - Speed of sound at given outside air temperature $[m/s]$;

$$M = \sqrt{\frac{2\kappa}{\kappa - 1} \left[\left(\frac{Q_c}{p_h} + 1\right)^{\frac{\kappa-1}{\kappa}} - 1\right]}\,[-] \tag{2.9}$$

2.1.1 Methods of Measuring Air Data Parameters

Pitot-static probes, static ports and other parts of pitot-static system with various types of pressure sensors are used for sensing air data parameters. [Figure 2.3].

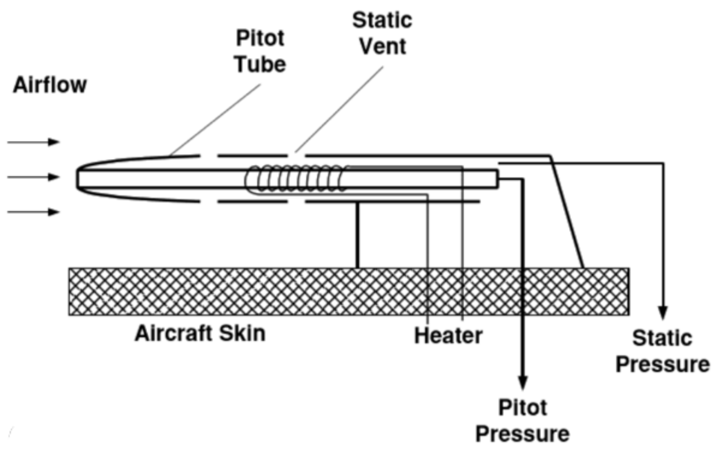


Figure 2.3: A electrically heated pitot-static probe [38].

Nowadays in modern aircraft are frequently used electrical pressure transducers, which include Micro Electro Mechanical Structure (MEMS) or resonant pressure transducers.

Resonant pressure transducers are sensing changes in the natural resonant frequency of vibrating element. These changes are caused by the input pressure. Transducer provides frequency output related to the pressure being measured. Sensor includes a thin walled cylinder with the input pressure acting on the inside and outside space at vacuum reference pressure [Figure 2.4][57]].

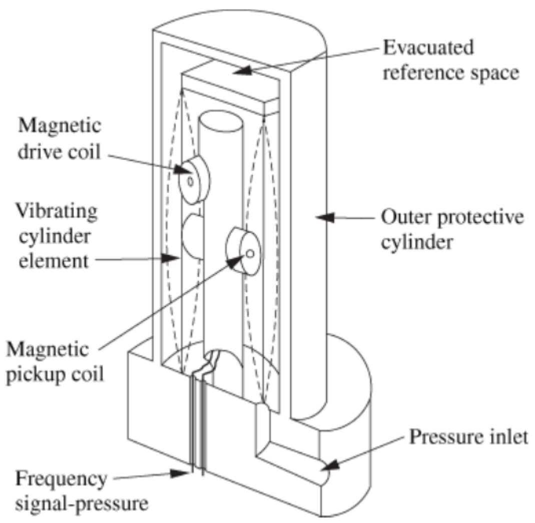

Evacuated reference space
Magnetic drive coil
Vibrating cylinder element
Outer protective cylinder
Magnetic pickup coil
Pressure inlet
Frequency signal-pressure

Figure 2.4: Resonant pressure transducer [57].

Whole cylinder is maintained in vibration mode by making it part of a feedback oscillator by sensing the cylinder wall displacement, processing and amplifying the output signal and feeding it back to a suitable force producing device. However, cylinder is density sensitive and it causes changes in frequency of the output signal, but this change is not more significant than that due to changes in input pressure [57].

Currently, resonant pressure transducers are essential part of the modern aircraft and they are placed inside the air data computers (ADC). ADCs provide all air data parameters for other aircraft systems which are connected by fieldbus e.g. CAN (Controller Area Network), ARINC 429 (Aeronautical Radio, Incorporated), CDSB (Commercial Standard Digital Bus) or MIL STD 1553. ADCs are using digital data processing and calculation from an pitot-static system, which is attached by ports on the cover of ADC. This approach is part of the integrated modular avionics and provides better reliability, an easy operation for maintenance and easy system backup.

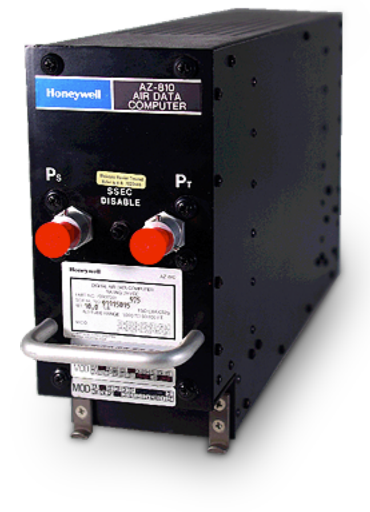


Figure 2.5: Honeywell AZ-810 Air Data Computer [7].

Above-mentioned MEMS technology uses capacitive and piezoresitive pressure sensors. The biggest advantage of MEMS pressure transducers is small size, low weight and very easy integration with the other electronics. They are sensitive to very small changes in pressure [1].

MEMS pressure sensors are mainly used in the Unmanned Aerial Vehicles (e.g., drones), due to their small dimensions. Usually they are placed in the modules together with related electronics (e.g., microcontroller unit (MCU)). This module has equivalent function like the ADC [13].

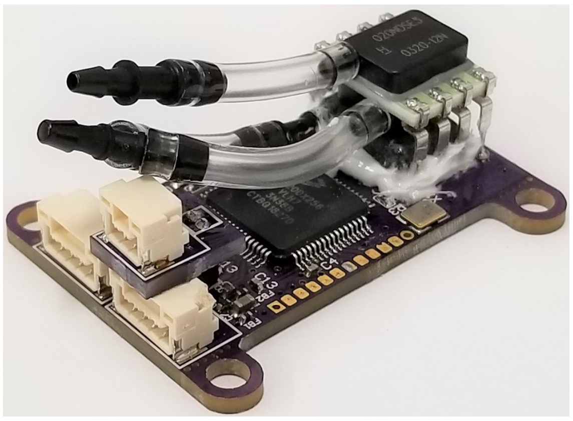


Figure 2.6: Avionics Anonymous MicroADC Air Data Computer [13].

2.1.2 Angle of Attack & Angle of Sideslip

Aerodynamic angles, angle of attack α and angle of sideslip β are related angles between aerodynamic axis frame F_A and body axis frame F_B [Figure 2.7].

$$F_B = \{x_B, y_B, z_B\} \tag{2.10}$$

$$F_A = \{x_A, y_A, z_A\} \tag{2.11}$$

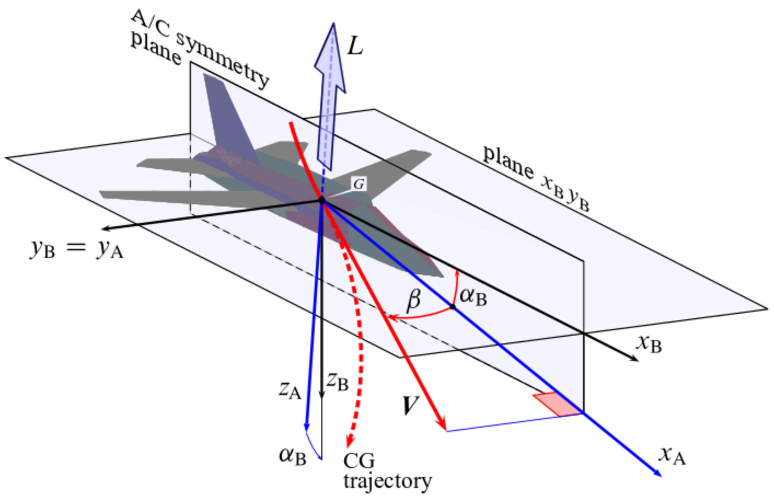


Figure 2.7: Aerodynamic frame, defining the aerodynamic angles α and β [31].

G, CG - centre of gravity; V - relative wind velocity vector; Lw - lift force; x_B - roll axis; y_B - pitch axis; z_B - yaw axis;

Angle of attack (angle of incidence) is angle between the direction of the air velocity relative wind and the wing chord line [Figure 2.8]. Angle of sideslip is angle between direction of the relative wind velocity vector and its projection on the plane x_B z_B [18].

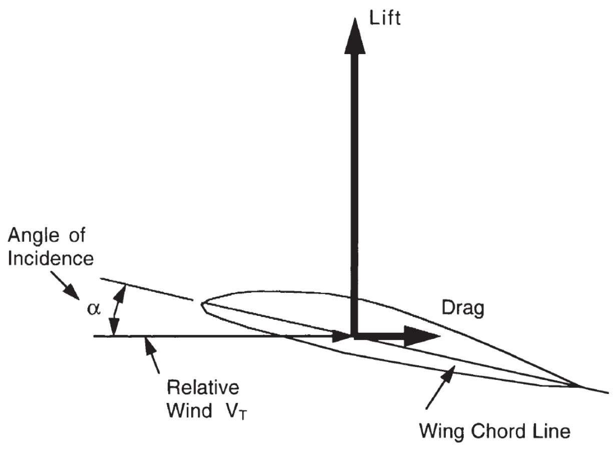
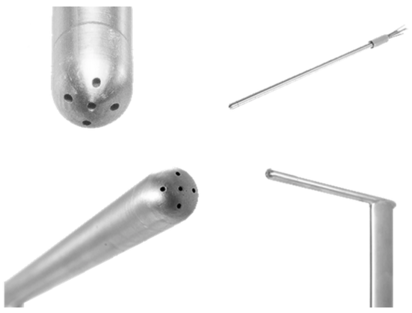


Figure 2.8: Angle of attack α [18].

This aerodynamic angles are measured by vanes [15][45][32] or by special Pitot tubes with one or more holes (pressure ports) on the front face of the probe which are designed to be sensitive to aerodynamic angles [21][2][figure 2.9]. These probes are called as differential-pressure pitot-static probes.


(a) : Air Data Probes [2].

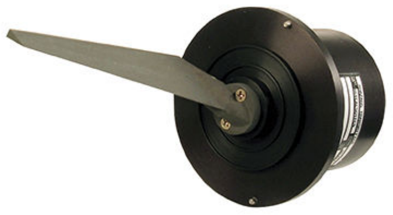

(b) : Wing-mounted angle-of-attack probe [17].

Figure 2.9: Measuring devices for angle of attack.

Proper measurement of aerodynamic angles is important for aircraft control system and related flight properties. In equation 2.1 C_l denotes lift coefficient which is a function of the angle of attack and it expresses the effectiveness of the airfoil section in generating lift. Relationship between C_l and α is linear up to a certain value of $\alpha_{critical}$ when the lift force is maximal. Then the airflow starts to break away from the upper surface of the wing and lift force falls off rapidly [18].

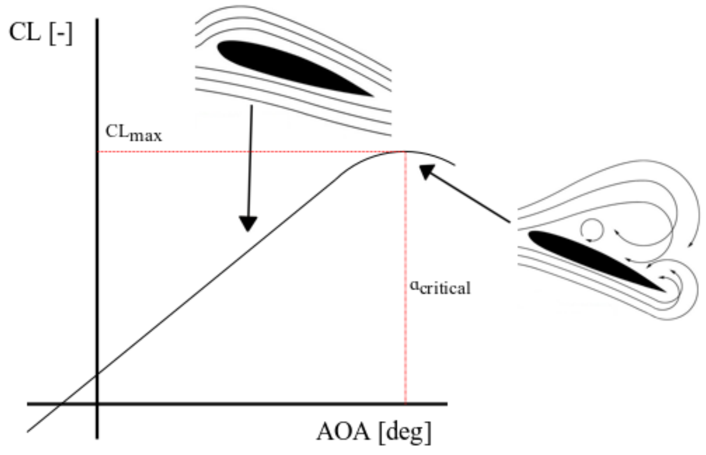


Figure 2.10: Angle of attack vs lift coefficient.

2.1.3 Differential-pressure Pitot-Static Probes

Differential-pressure pitot-static probes, which are intended to be used for an angle of attack measurement, has two holes placed and oriented at equal angles from side of the logitudinal axis of the pitot-static tube. In case of any changes in wind flow direction (changes in α) the pressure difference which exists at the two holes is measure of the angle of attack. This pressure difference directly depends on the shape of the nose of the pitot-static probe, on the angular position of the holes and also depends on Mach number and Reynolds number [Figure 2.11] [40].

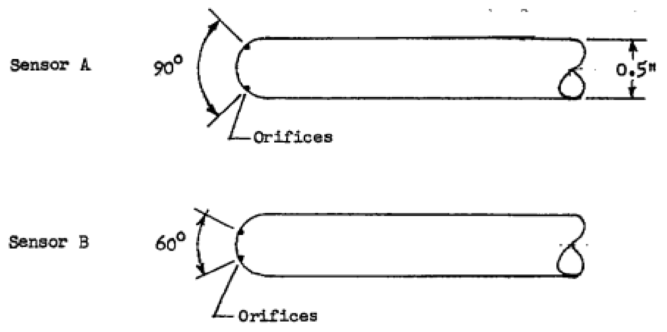


Figure 2.11: Various shapes of differential pitot-static probes [40].

For this type of measurement are necessary two aerodynamic quantities: differential pressure and impact pressure. In this case, the ratio of the differential pressure and impact pressure is determined as function of angle of attack [Equation 2.12] [Figure 2.12] [40].

$$\frac{\Delta p}{q} = f(\alpha) \tag{2.12}$$

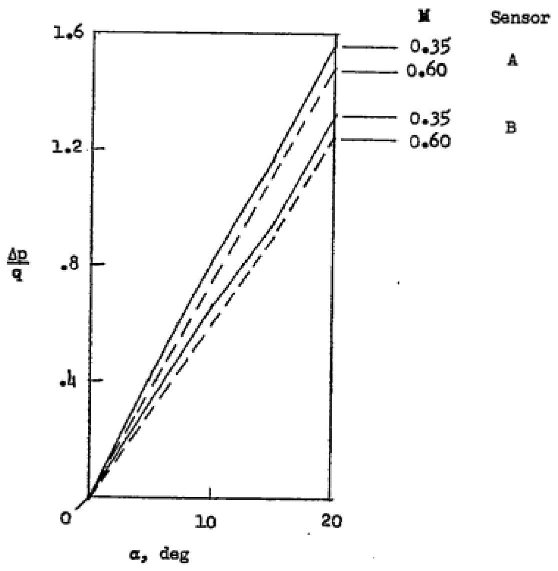


Figure 2.12: Variation of $\Delta p/q$, measured by two types of differential pitot-static probes with hemispherical nose shapes. Measured at two values of airspeed. β = 0° [40].

The sensing holes, which are spaced by higher angle, show more sensibility to changes in angle of attack.

In [43] is described the differential pressure tube that works as pneumometric sensor for measuring aerodynamic angles. Operation of this sensor is based on measuring differential pressure as previous type of probe. Tube has major central hole for total pressure measurement and two pairs of holes placed symmetrically in horizontal and vertical planes with respect to the aircraft. Angle of attack is determined as the difference of pressure measured by holes the vertical plane. Similarly, the sideslip angle is determined as pressure difference measured by holes in the horizontal plane.

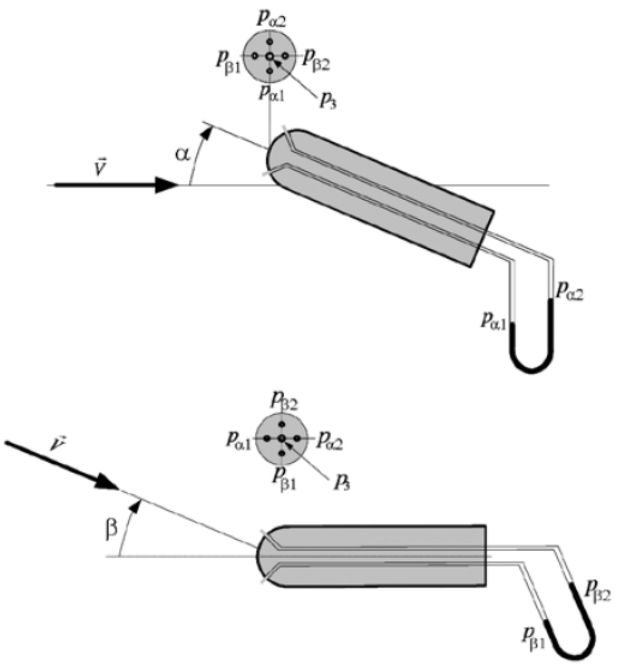


Figure 2.13: Calibration of the differential pitot-static probe [40].

These aerodynamic angles can be determined from [Equation 2.13] [Equation 2.14]. Coefficients k_1 and k_2 should be selected experimentally.

$$\alpha = \frac{p_{\alpha1} - p_{\alpha2}}{k_1 \left(p_3 - \frac{p_{\beta1} + p_{\beta2}}{2} \right)} \tag{2.13}$$

$$\beta = \frac{p_{\beta1} - p_{\beta2}}{k_1 \left(p_3 - \frac{p_{\beta1} + p_{\beta2}}{2} \right)} \tag{2.14}$$

Paper [58] describes the set of modified pitot-static probes for measuring aerodynamic angles. In this case, four aluminium probes were attached to a standard pitot probe. Determination of aerodynamic angles is based, as previous, on measuring differential pressure.

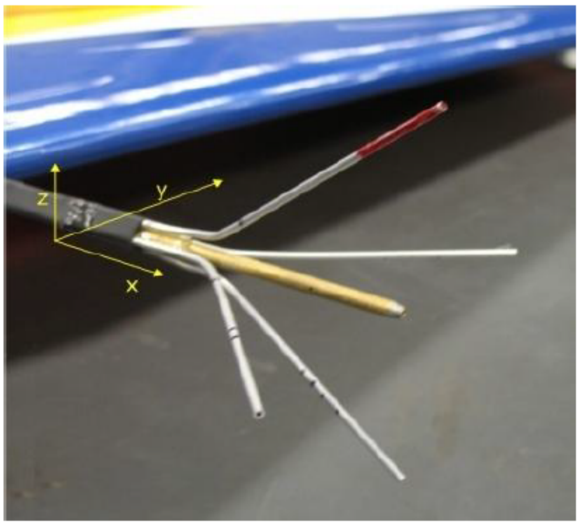


Figure 2.14: Wing mounted installation of angle of attack, sideslip angle and pitot probe [58].

System for determination of aerodynamic angles considers pressure differences measured by probes and pressure distribution along the wing chord measured by specific static ports.

Chapter 3

Concept of Aerometric System for Unmanned Aerial Vehicle

The proposed aerometric system for UAV includes the following features:

1. measuring barometric altitude
2. measuring airspeed (V_{TAS}, V_{CAS}),
3. measuring vertical speed VS,
4. optional: GPS (Global Positioning System) position and UTC time (Coordinated Universal Time),
5. communication via CAN bus using CANaerospace protocol [51].

Based on above-mentioned points, system has the following functionalities:

1. pressure sensing by pitot-static probe;
2. pressure sensing measurements using static pressure transducer,
3. differential (dynamic) pressure measurements using differential pressure transducer,
4. optional: determination of the position and UTC time from GPS using GNSS (Global Navigation Satellite System) transceiver,
5. data processing and calculation of the air data parameters using MCU,
6. data synchronization using MCU,
7. data communication with other systems in UAV using CAN bus transceiver.

Another requirements related to the proposed aerometric system:

1. pressure sensors shall be located as close as possible to the pitot-static probe to eliminate long air pressure hoses and related issues,

2. hardware and assembled printed circuit boards shall have minimal size and weight,
3. hardware shall have resistance to environmental conditions of application (temperature, pressure, humidity, vibrations, electromagnetic compatibility).

3.1 Operational Requirements

The proposed aerometric system is intended to be used for UAV with fixed wings with the following operational parameters:

- V_{TAS} for range: $0\,\mathrm{m\,s^{-1}}$ to $110\,\mathrm{m\,s^{-1}}$,
- H for range: $0\,\mathrm{m}$ to $1000\,\mathrm{m}$.

However, assuming ISA and variety in static pressure at $H = 0\,\mathrm{m}$, ideal measured range for static pressure p_h is 850 hPa to 1030 hPa.

Intended range of the V_{TAS} is equivalent to dynamic pressure Q_c, for range from 0 Pa to 1960 Pa (for $H = 0\,\mathrm{m}$). Optimal measured range for dynamic pressure Q_c is from 0 Pa to 2500 Pa.

3.2 System Architecture

For further various tests and measurements, the proposed system has an ability to measure air data parameters from two pitot-static probes. Due to this extension, system consists of two measuring modules: Module A and Module B and also one main module [Figure 3.1]. However, system can be optionally extended to more measuring modules. Each module represents stand-alone assembled printed circuit board (PCB) with related functionality.

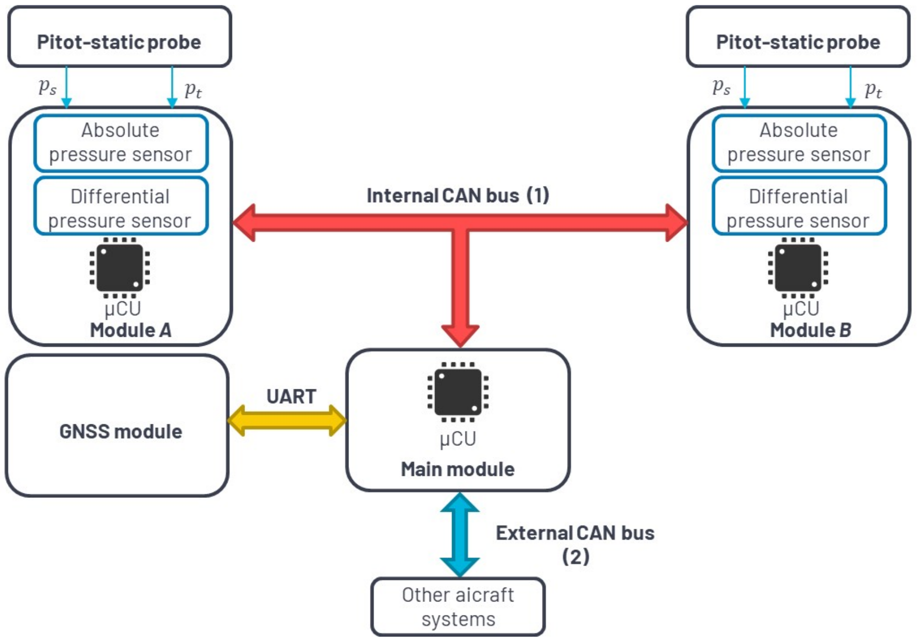


Figure 3.1: Top level architecture of the designated aerometric system. *µCU* - Microcontroller Unit.

Based on the previous points, measuring modules are intended to be mounted inside the aircraft wings. Measuring modules and main module are using CAN bus with CAN Aerospace protocol for data communication. CAN Aerospace protocol requires bus speed $1\,\mathrm{Mbit\,s^{-1}}$. The bus is immune to electromagnetic interference, because the signals on the both CAN lines are subject to the same electromagnetic influences, and so the difference in voltages between these two lines does not vary. There are two separated data buses:

1. Internal - for data communication between main module and measuring modules;
2. External - for data communication between aerometric system and other aircraft systems.

From the top level design point of view, the measuring modules with pressure sensors convert digital signal from pressure transducers to related values with correction and also establish data communication by CAN bus.

The primary function of the main module is data transfer between aerometric system and other aircraft systems by CAN bus. Module also carries out process of the GPS data (UTC time and GPS position) from GNSS module, which is connected to module by UART bus (Universal Asynchronous Receiver-Transmitter).

The all above mentioned features and functionalities are implemented in MCU in related modules.

Chapter 4
Hardware Design

Hardware design of the measuring and main modules includes electronic parts and components selection, mainly pressure sensors following PCB design of modules. PCBs were created using the professional software Altium Designer [3]. Finally, all modules were assembled using manual soldering.

4.1 Component Selection

Selected components satisfy the mentioned requirements in [Chapter 3]. The selection was also based on previous experience with the related components. However, the other properties were taken into account and firmware support and development, reliability or availability of components in the markets.

In the case of the pressure sensors, the selection was focused on sensors with digital interface due to spare space on PCB. PCBs were assembled by Surface Mount Device (SMD) components.

4.1.1 Absolute Pressure Sensor

In table [Table 4.1] are described parameters of two considered absolute pressure sensors. Absolute pressure sensor MPRL has a higher resolution of the internal ADC than the HSC sensor. For example, one single output count of the ADC (HSC sensor) is approx. 7 Pa. In terms of the barometric altitude H it could be considered as approx. 0.58 m. However, for the chosen MPRL sensor, one single count is mathematical considered as >0.1 Pa [Table 4.1]. The sensor MPRL, as mentioned above, was also selected due to its smaller dimensions. MPRL sensor has compensated temperature range from 0 °C to 50 °C and typical power consumption is 10 mW.

Parameter	MPRLS0015PA0000SA [28]	HSCMAND015PA2A3[27]
Range [Pa]	$0 - 103421,35$	$0 - 103421,35$
Bus interface	SPI	I2C
TEB	$\pm 1,5$ % *FSS*	± 1 % *FSS*
Power Supply	$3,3$ *V*	$3,3$ *V*
Size [mm]	5 x 5	10 x $13,3$
ADC	24 *bit*	12 *bit*

Table 4.1: Considered absolute pressure sensors.

FSS - Full Scale Span; *TEB* - Total Error Band; *I2C* - Inter-integrated circuit interface; *SPI* - Serial peripheral interface; Analog to Digital Converter;

Figure 4.1: Absolute pressure sensor MPRLS0015PA0000SA [26].

4.1.2 Differential Pressure Sensor

Both considered differential sensors which are described in [Table 4.2], has the same resolution of the ADC, but 4525DO sensor has the wider range, which is useless for the application [Table 4.2]. It can be used for the wider airspeed range. Also, the 4525DO sensor has sided ports, and this solution is impractical. This sensor is included in the mentioned kit for airspeed measuring ASS100ADV. The chosen differential sensor, NPA-700B-10WD, has compensated temperature range from 0 °C to 60 °C.

Parameter	NPA-700B-10WD [5]	4525DO-DS3AI001DS[52]
Range [Pa]	± 2490	$\pm 6894,75$
Bus interface	I2C	I2C
TEB	$\pm 1,5$ % *FSS*	± 1 % *FSS*
Power Supply	5 *V*	$3,3$ *V*
Size [mm]	$11,36$ x $7,52$	$12,4$ x $9,9$
ADC	14 *bit*	14 *bit*

Table 4.2: Considered differential pressure sensors.

Sensor NPA-700B-10WD operates in a continuous measurement mode with current consumption of approximately 3 mA [6].

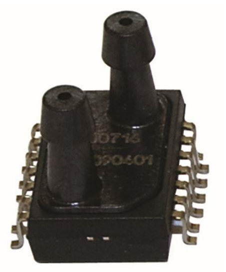


Figure 4.2: Differential pressure sensor NPA-700B-10WD [8].

4.1.3 Microcontroller

The chosen microcontroller STM32F446RC (manufactured by STMicroelectronics) has ARM Cortex-M4 32-bit RISC (Reduced Instruction Set Computer) core [48]. This microcontroller has the main following features, which are necessary and required for the application:

- Core operating at a frequency of up to 180 MHz;
- Floating point unit (FPU);
- 256 kB of Flash memory (internal);
- 128 kB of Static Random-Access Memory (SRAM);
- Timers: General-purpose: 10, Advanced-control: 2 and Basic: 2;
- Communication interfaces: CAN bus: 2, I2C: 4, SPI: 4, USART: 4, UART: 2;
- Operating voltage: 1.7 V to 3.6 V;
- Package: LQFP64 (SMD);
- Direct memory access (DMA): 2 ports with 8 streams each;
- High Abstraction Layer libraries (HAP) - Application Programming Interface (API) provided by the manufacturer;
- Independent watchdog;
- 5 V tolerant input/output pins;
- ARM Embedded Trace Macrocell (ETM);
- Serial Wire Debug interface (SWD);

Based on microcontroller specification, the ARM (Advanced RISC Machines [11]) core on operating frequency 180 MHz is necessary for CAN bus transmission at required speed (1 Mbit s^{-1}). FPU is useful for more powerful computation. Above mentioned values of the flash memory and SRAM should provide sufficient resources for related functionalities. ETM and SWD are necessary for application debugging during software development [47].

The used microcontroller is also fully compatible with selected pressure sensors. Although the differential pressure sensor is working with 0 V to 5 V I2C lines, it does not present any issue from the microcontroller side because it has 5 V tolerant input/output pins. Microcontroller normally works from 0 V to 3.6 V levels.

Maximum current consumption is 42 mA for the following conditions: RUN mode, external clock, phase-locked loop (PLL) on, all peripherals disabled, operating core frequency 180 MHz and ambient temperature 25 °C.

4.1.4 CAN Bus Transceiver

CAN bus transceiver MCP2551, developed by MICROCHIP, is widely used as industrial high-speed CAN transceiver. It supports up to 1 Mbit s^{-1} operation speed. Power supply is 4.5 V to 5.5 V [35]. Maximum current consumption is 75 mA for dominant condition.

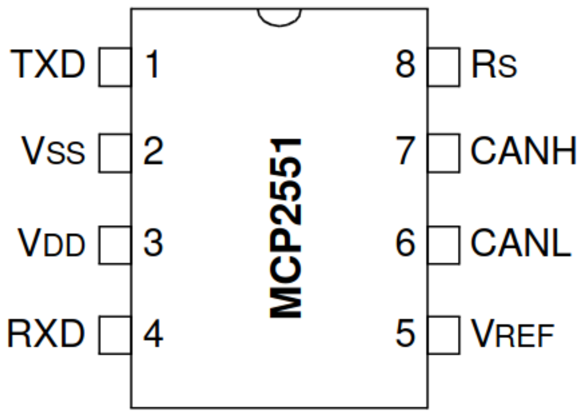


Figure 4.3: MCP2551 Pinout [35].

TXD - Transmit Data Input; *Vss* - Ground; *Vdd* - Supply Voltage; *RXD* - Receive Data Output; *Rs* - Slope-Control Input; *CANH* - CAN High-Level Voltage Input/Output; *CANL* - CAN Low-Level Voltage Input/Output; *Vref* - Reference Output Voltage;

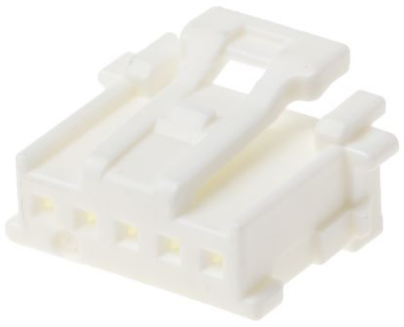
TXD is a transistor-transistor-logic (TTL) compatible input pin and all input/out pins of the microcontroller are Complementary Metal Oxide Semiconductor (CMOS) and TTL compliant (no software configuration required). This part was selected in SOIC package.

For CAN bus transceiver was selected slope-control operating mode which further reduces electromagnetic interference (EMI) by limiting the rise and fall times of CANH and CANL. The slope, or slew rate (SR), is controlled by connecting an external resistor R_{ext}. The chosen value of this resistor is 10 kΩ and it is equivalent to maximum value of slew rate, approx. 24 V/µs.

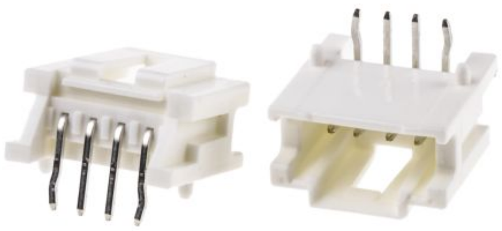
All boards are including 120 Ω resistors for termination.

4.1.5 Module Connectors

The selected MicroClasp Wire-to-Board System, manufactured by Molex, satisfies the operational conditions and has minimal size. The most significant property of this connection system is resistance to vibrations, because the connectors (Crimp Housing [36]) are locked inside the right-angled PCB headers [37] [Figure 4.4]. These connectors provide the connection for the CAN bus and the power supply.



(a) : MicroClasp-to-MicroClasp Off-the-Shelf Cable Assembly (Crimp Housing) (molex 51382 Series) [9].



(b) : Wire-to-Board Header (PCB Header) (molex 51382 Series) [10].

Figure 4.4: Board connectors.

4.1.6 GNSS Module

As GNSS module for acquiring GPS signal was selected board with assembled GNSS receiver NEO-M8N-0-10, receiving GNSS antenna and power supply circuit. Communication with the selected module is provided by the UART bus. Board is powered with 5 V [55].

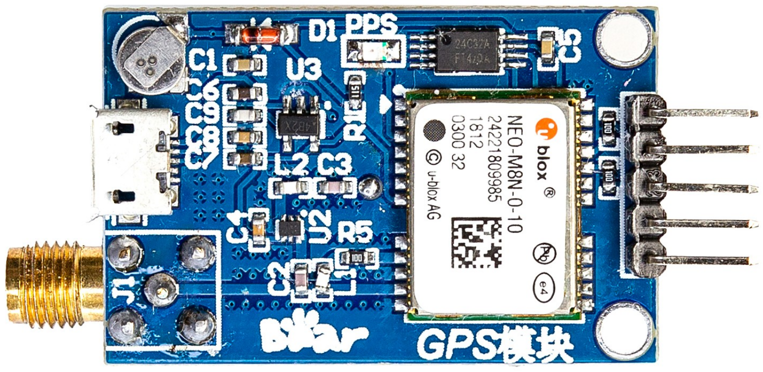


Figure 4.5: GNSS module with NEO-M8N-0-10 GNSS receiver [8].

4.1.7 Other Electronic Parts

SMD resistors and capacitors were selected from the corresponding values in the 0805 package for measuring and main modules. All modules are powered from the same 12 V battery, so the voltage regulators for 5 V and 3.3 V power supply are necessary. Modules design includes the following voltage regulators, in series connection:

- BD50GA5MEFJ-LBH2 [44], manufactured by ROHM Semiconductor, from max. 14 V to 5 V, HTSOP-J8 package;

- TPS73633DBVT [54], manufactured by Texas Instruments, from max. 5.5 V to 3.3 V, SOT-23-5 package;

Measuring modules include digital temperature sensor DS18B20 in the TO-92 package, manufactured by Maxim Integrated. The sensor communicates over a 1-Wire bus and provides resolution of the temperature to digital conversion to 12 bit [33]. The temperature sensor can be used for real-time temperature compensation that should be included in software implementation.

4.2 Module Design

As previously mentioned in the section [Section 3.2], it was necessary to design two different boards (layouts) with different functionalities: the measuring module and the main module. However, these PCBs have the same components: voltage regulators, microcontrollers, CAN interface or connectors. It means that boards are similar at some points.

4.2.1 Schematic Design

PCB schematic design is based on a system-level block diagram of the related module [Figure 4.6] [Figure 4.7]. PCB schemes are in [Appendix A.1] and in [Appendix A.2].

Modules are powered from the same 12 V battery over the main connector into BD50GA5MEFJ-LBH2 voltage regulator. 5 V is the power supply for the CAN interface, differential pressure sensor or another voltage regulator TPS73633DBVT. The second voltage regulator provides power supply 3.3 V for the microcontroller, absolute pressure sensor or temperature sensor. The approximate current consumption of one module is 70 mA. Values and types of the related capacitors were chosen based on manufacturers' recommendations. Diode *D5* (in main module schematic) 1N4007 is used as protection against reverse polarity.

Other diodes, ESDAVLC6-2BLY, protect the CAN bus lines against electrostatic discharge (ESD) and work as a transient volt suppressor (*D7*, *D8*, *D3* and *D4* in main module schematic). For CAN buses, DLW43SH101XK2L (*FL1* and *FL2* in main module schematic) common mode chokes are used for common-mode noise suppression.

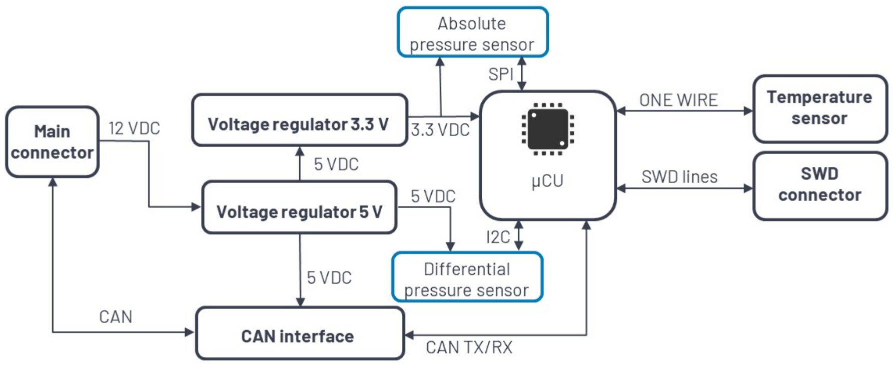


Figure 4.6: System-level block diagram of the measuring module.

Inductors ,BLM21AG102BH10D, *BEAD1* (in main module schematic) were later replaced with 0 Ω resistor due to a high voltage drop which caused not sufficient voltage for the microcontroller. This inductor is recommended by manufacturer for ADC power supply. However, ADC functionality is not implemented in the design.

Resistor *R2* (in main module schematic) selects internal flash memory as source for the boot. Crystal ABM3B-8.000MHZ-10-1-U-T provides 8 MHz for microcontroller. This frequency is using by internal PLL and switched to the required 180 MHz.

The design of all boards includes SWD connectors for debugging during the firmware development process. SWD interface is compatible with ST-LINK/V2 in-circuit debugger/programmer [46].

The I^2C bus is used with internal pull-up resistors with a typical value of 40 kΩ.

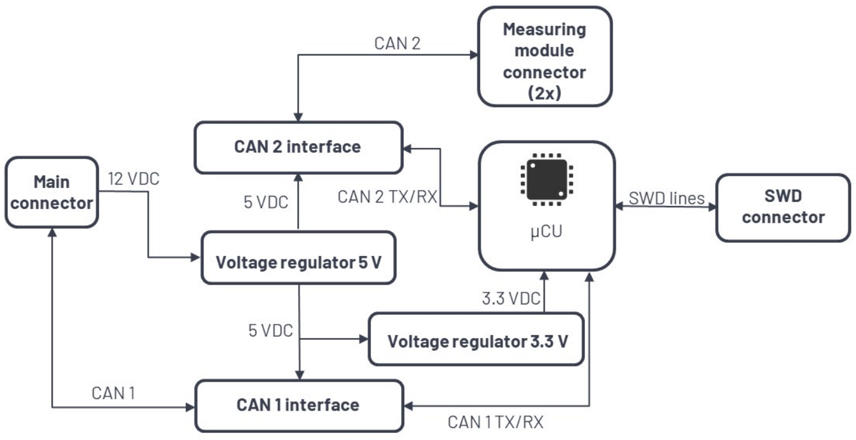


Figure 4.7: System-level block diagram of the main module.

4.2.2 Board Desing and PCB Assembly

Modules were designed as two-sided PCBs with no more internal layers. Boards were designed with minimal size as possible. However, some physical properties of components, such as dissipated heat, were considered. For better heat conduction were designed wider tracks or placed thermal vias around voltage regulators or CAN bus transceivers. On all boards were designed on both sides ground planes. Board were manufactured by JLCPCB.COM [30]. Also, the technical capabilities of the PCB manufacturer were taken into account, such as minimum clearance of the 0.15 mm. Manufactured boards are depicted in [Appendix A.3] .

All modules were assembled using manual soldering tools and devices at home condition. However, the absolute pressure sensors were soldered in the laboratory in the Department of Electrotechnology of the Czech Technical University in Prague (The Faculty of Electrical Engineering). It was not possible to solder this sensor without any damage due to its small size. Sensor has 12 connection pads on the bottom side [Figure 4.8].

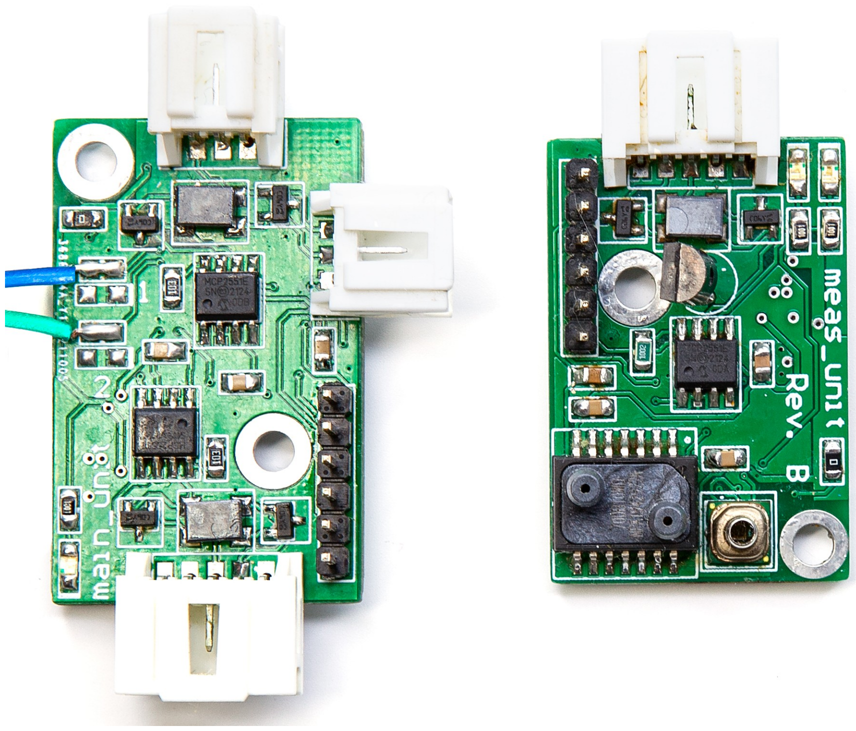


Figure 4.8: Assembled modules - top side: Main module (left) and measuring module (right).

Dimensions of the PCBs are: main module 38.5 mm x 24.5 mm and measuring module 35.5 mm x 24 mm. Thickness of the PCB is 1.6 mm. Each board has two holes of 3.2 mm diameters placed diagonally. M3 Hexagon Socket Head Cap Screws (DIN 912 norm) shall be used for mounting.

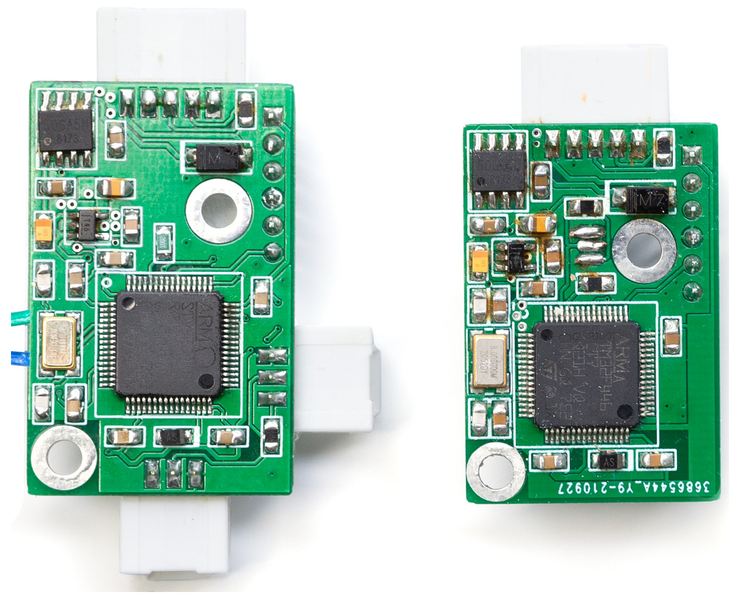


Figure 4.9: Assembled modules - bottom side: Main module (left) and measuring module (right).

After main module assembly, two LEDs and related resistors (marked as *R8*, *D6*, *R4* and *D2*) were disassembled and replaced by two wires. These wires provide communication with GNSS module using UART bus.

Chapter 5

Communication Busses and Protocols

5.1 Controller Area Network

CAN bus, developed by BOSCH, is a multi-master, half duplex message broadcast system with maximum speed 1 Mbit s^{-1} (at maximum distance 40 m). Originally it was developed for automotive industry for sending short messages or values. CAN bus is also defined as serial communication bus by International Standard Organization (ISO-11898: 2003). CAN bus protocol includes standard Open Systems Interconnection (OSI) model [16][29].

5.1.1 CAN Frame and Specifications

CAN bus specifies standard format [Figure 5.1] and extended format.

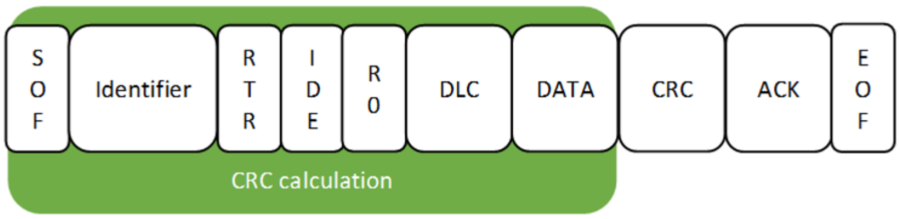


Figure 5.1: CAN bus standard frame.

The meaning of the each bit field of standard format is following:

- SOF - Start of frame (dominant bit), marks start of a message;
- Identifier - 11 bits identifier. Identifier also determines priority of the message. The lower value, the higher is priority;
- RTR - Single remote transmission request, if dominant the information is required from node with specified identifier and data bytes are not used in transmission;
- r0 - reserved bit;

- DLC - 4 bits data that contains the number of bytes for transmission;
- ACK - 2 bits field, one is acknowledgment bit and the second is delimiter bit. Every node in network which receives message overwrites this recesive bit in original message with a dominate bit. It indicates that message has been sent without any errors;
- DATA - maximum 64 bits of message data;
- CRC - 16 bits field, cyclic redundancy check contains the checksum of the data;
- EOF - 7 bits, field for the end of CAN frame. In case of bit stuffing error this bit field is dominant. Bit stuffing is used for maintaining synchronization. When there are 5 bit of the same logic level in frame, one bit of the opposite logic level is placed into the bits.

The extended format is the same as standard format with the addition of the following:

- SRR - Substitute remote request which replace RTR bit;
- IDE - This recesive bit indicates the following 18 bits;
- r1 - reserved bit;

In CAN bus all nodes always receive all request and data, but in the same time two nodes must not transmit message with the same identifier. The remote frame is used for transmission request from another node. In this frame are not present data and RTR bit is recessive [53].

Another specification is used for arbitration. In case that two or more nodes start transmission of the message at the same time, the conflict is resolved by bitwise arbitration using frame identifier. If there is simultaneously transmission of the data frame and remote frame with the same identifier, the data frame prevails over the remote frame. For error detection is also used cyclic redundancy check with defined calculation [16].

5.1.2 CANaerospace Frame

CANaerospace can be used with 11 bit and 29 bit identifiers. CANaerospace specifies using the DATA bits in CAN frame [51].

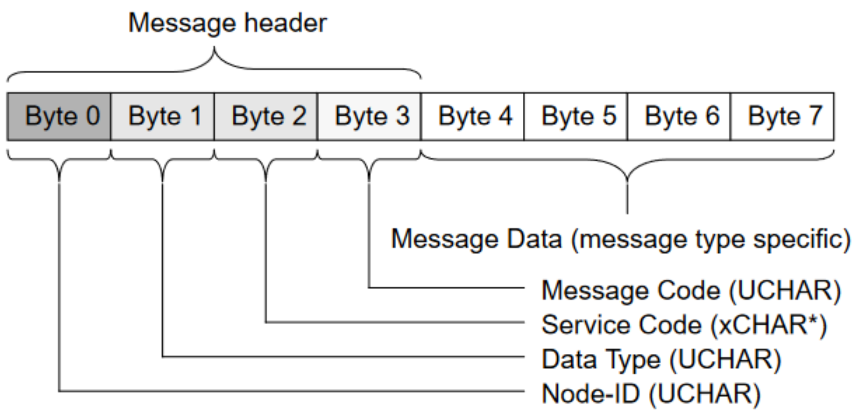


Figure 5.2: CANaerospace frame [51]

The meaning of data fields of message header is following:

- Node-ID - The node ID in range 0 - 255;
- Data Type - The data type of the transported message data;
- Service Code - The service code consists of 8 bits which may be used as required by the specific data, but should be set to zero if unused;
- Message Code - The message code is incremented by one for each message and may be used to monitor the sequence of receiving messages.

CANaerospace specifies the CAN identifiers for flight state, flight parameters or flight controls. Example of some commonly used identifiers for flight state parameters is in [Table 5.1].

CAN identifier	Flight state parameter name	Suggested data types	Units
332 ($14C)	True altitude	FLOAT SHORT2	m
339 ($153)	Total pressure	FLOAT SHORT2	hPa

Table 5.1: Example of defined CANaerospace message identifiers[51]

CAN identifiers in range 1300 – 1499 are not exactly defined. They are reserved for future use.

Transmitted data are represented by basic data formats. In CANaerospace frame are these formats defined [51].

Data type	Range	Bits	Explanation	Type
FLOAT	1-bit sign 23-bit fraction 8-bit exponent	32	Single precision floating-point value according to IEEE-754-1985	2 ($02)
CHAR4	-128 to +127	4 x 8	4 x 2's complement char integer	15 ($0F)
DOUBLEH	1-bit sign 52-bit fraction 11-bit exponent	32	Most significant 32 bits of double precision floating-point value according to IEEE-754-1985	30 ($1E)
DOUBLEL		32	Least significant 32 bits of double precision floating-point value according to IEEE-754-198	31 ($1F)

Table 5.2: Example of defined CANaerospace data types[51]

.

5.1.3 CAN Bus Implementation

In application is CAN bus used for communication between modules and it is used as major interface for aerometric system with other aircraft systems with speed of 1 Mbit s^{-1}. Standard CAN frames which are transmitted from aerometric system contains measured aerometric parameters.

Selected MCU has CAN bus periphery called bxCAN that provides CAN bus communication up to two independent CAN buses (dual CAN operation). Periphery configurations provide setting of bit timing at required bus speed. Periphery speed was set to 1 Mbit s^{-1} [47].

Module name	Node-ID
Main module	100
A module	101
B module	102

Table 5.3: Node-ID in application.

Used node IDs in application are listed in 5.3 and used 11 bit CAN identifiers are listed in [Appendix B.2].

Periphery also provides acceptance filters for selecting incoming messages. In addition, the filtering is not aplicated only to CAN identifier, but it can be used for filtering frames with RTR bits. This feature improves the performance of CAN bus communication, because for filtering process any software resources are not necessary (without disturbing the software). MCU has two FIFOs (First In, First Out) buffers for each CAN periphery, with size 3 frames, which provide storage for accepted messages. Message which is not accepted is discarded. In application are used multiple FIFOs to receive required frames to avoid FIFO overflow.

In application are used filters in list mode with the exact CAN identifiers and RTR bits which should be accepted. After placing accepted message in related FIFO, the interrupt is generated to signal that received message is pending in FIFO. Then message is taken from FIFO and processed by SW.

For message transmission are used transmit mailboxes. In order to send a message, it is necessary to select one empty transmit mailbox by SW. Transmit priority was set to transmit request order. When message has been successfully transmitted, mailbox becomes empty again.

```c
/* GPS module connected */
CAN_FilterTypeDef sFilterConfig_GPS_MODULE;
sFilterConfig_GPS_MODULE.FilterFIFOAssignment =
    CAN_FILTER_FIFO0;
/* Setting CAN ID list accodring to documentation */
/* CAN_FxR1[31:24] */
sFilterConfig_GPS_MODULE.FilterIdHigh = (ID_UTC << 5);
/* CAN_FxR1[23:16] */
sFilterConfig_GPS_MODULE.FilterIdLow = (
    ID_GPS_AIRCRAFT_HEIGHT_ABOVE_ELLIPSOID << 5);
/* CAN_FxR1[15:8] */
sFilterConfig_GPS_MODULE.FilterMaskIdHigh = (
    ID_GPS_AIRCRAFT_LATITUDE << 5);
/* CAN_FxR1[7:0] */
sFilterConfig_GPS_MODULE.FilterMaskIdLow = (
    ID_GPS_AIRCRAFT_LONGITUDE << 5);
sFilterConfig_GPS_MODULE.FilterScale =
    CAN_FILTERSCALE_16BIT;
sFilterConfig_GPS_MODULE.FilterMode =
    CAN_FILTERMODE_IDLIST;
sFilterConfig_GPS_MODULE.FilterActivation = ENABLE;
sFilterConfig_GPS_MODULE.FilterBank = 4;
sFilterConfig_GPS_MODULE.SlaveStartFilterBank = 14;
/* Configure the filter */
if (HAL_CAN_ConfigFilter(&hcan1, &sFilterConfig_GPS_MODULE
    ) != HAL_OK)
    /* Filter configuration was not setup */
    ErrorHandler();
```

Listing 5.1: Implementation of the configuration of the bxCAN registers for CAN ID filtering using *HAL library*.

Visualization of CAN bus transmission with decoded frames in application is depicted in [Appendix B.1]. For data request from measuring modules and main module are implemented remote frames with RTR recessive bit. Modules are responding by frames with requested data. Each parameter can be individually received by sending related CAN identifier. Maximum frequency for sending this remote frames is 20 Hz. However, in designed aerometric system is implemented functionality, which provides data frames in row with all available parameters. This stream of data frames can be received by sending one remote frame with CAN identifier 1499. Maximum frequency for requesting the mentioned frame is 10 Hz. Detailed description of software implementation and testing of the CAN bus is in [Section 6].

5.2 Inter-Integrated Circuit

Inter-Integrated Circuit (I^2C), designated by NXP, is serial half-duplex bus with two bus bidirectional lines connected to a positive voltage supply with pull-up resistor, marked as [41]:

- SDA - serial data line, for data transmission between the master and slave device;
- SCL - serial clock line, provides the clock signal generated by the master device;

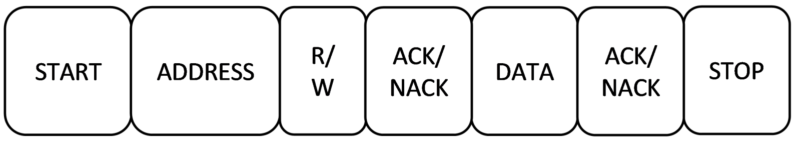


Figure 5.3: I2C frame.

5.2.1 Implementation in Application

In application is I^2C used for data communication with differential pressure sensor NPA-7000B-10WD with 100 kHz speed. Description of the communication frame is in [41].

For I^2C data transfer was implemented simple peripheral driver according to reference manual published by MCU manufacturer. Another option was to use peripheral driver developed by MCU manufacturer (*HAL library*), but this option caused timing issues.

5.3 Communication with GNSS Module

GNSS module provides communication using NMEA and UBX protocol using UART bus.

5.3.1 NMEA Protocol

NMEA frame is depicted in 5.4. Each message starts with $ and it ends with <CR><LF> ASCII characters. Messages are sent in one stream without any request [39][56].

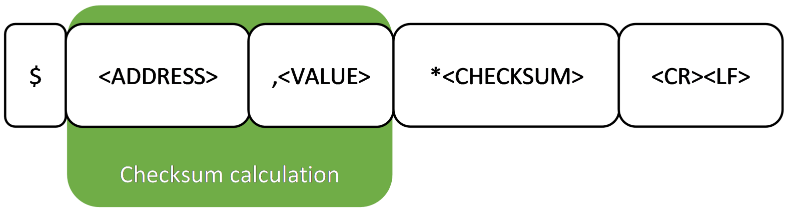


Figure 5.4: NMEA frame.

```
$GPGGA,183828.80,5006.39186,N,01435.11612,
E,1,04,7.74,166.0,M,44.2,M,,*5F
$GPGSA,A,3,19,22,32,14,,,,,,,,,17.88,7.74,16.12*08
$GPGSV,3,1,12,01,80,308,,03,47,253,,
04,08,195,,08,23,183,*7C
$GPGSV,3,2,12,10,00,068,,14,10,276,
17,17,25,315,08,19,07,326,15*77
$GPGSV,3,3,12,21,69,126,19,22,50,078,
15,31,09,109,23,32,36,055,31*71
$GPGLL,5006.39186,N,01435.11612,E,
183828.80,A,A*69
$GPRMC,183828.90,A,5006.39191,N,01435.11619,
E,3.945,17.38,010522,,,A*50
$GPVTG,17.38,T,,M,3.945,N,7.307,K,A*08
```

Listing 5.2: Received NMEA frames from connected GNSS receiver.

In NMEA frame, the first two characters of the address field describes talker identifier and next three characters are related to the message content. Value (data) field with ASCII content is followed by checksum, which is calculated with XOR function.

In application are filtered by SW only frames marked with *GPGGA* address that containts Global Positioning System Fix Data.

5.3.2 UBX Protocol

UBX protocol is defined by uBlox as proprietary protocol [55]. Frame is depicted in [Figure 5.5]. Every message starts with two same bytes: 0xB5 and 0x62. Class defines the basic subset of the message. Next is one byte value of the message ID. Length is defined as the length of the payload only and payload has variable length. Two last bytes are calculated checksum

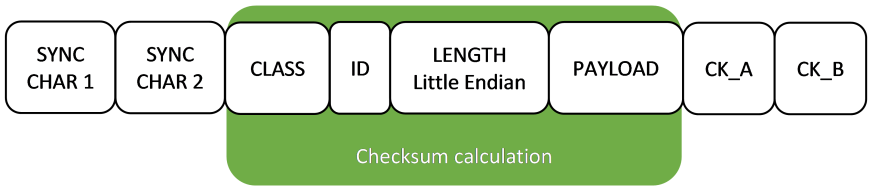


Figure 5.5: UBX frame.

Main module SW includes this protocol for configuration of the GNSS module after start-up. GNSS module is configured for measurement period of 20 Hz and UART baud rate is set to 921 600 bit s^{-1}. This configuration is used for more frequent update of parameters, especially UTC time.

5.4 Other Communication Protocols

5.4.1 Universal Asynchronous Receiver-Transmitter Bus

Universal Asynchronous Receiver-Transmitter (UART) is serial full-duplex communication bus. For connection are used only two wires, marked as Rx and Tx. UART Frame consists of start bit (logic 0), 8 data bits, special parity bit and stop bit (logic 1). On the same bus are always present only one master device and one slave device.

In SW implementation in Main module, UART provides communication with GNSS module with speed 9600 bit s^{-1} and 921 600 bit s^{-1}. MCU receives data stream from UART bus and stores it in memory buffer using DMA (Direct Memory Access). MCU also provides hardware interrupts when the UART bus is in idle state, buffer is filled in half or buffer is full. In interrupt routine service is memory buffer copied into another memory for following parsing of the NMEA protocol and other processing. The DMA provides faster data transmission without core processing.

5.4.2 Serial Peripheral Interface Bus

Serial Peripheral Interface (SPI) is a serial, full duplex bus and includes only one master device and one or multiple slave devices. For physical connection are used four wires: MOSI (master output slave input), MISO (master input slave output), CLK (clock signal) and SS (slave select).

The SPI bus is included measuring module design for communication with absolute pressure sensors. Communication is implemented according to manufacturer specification and it operates in blocking mode (without DMA), where MCU core directly receives the data.

5.4.3 One Wire Protocol

One wire protocol (*1-Wire*) provides bidirectional, serial, half duplex data communication at low speed over single wire. Bus consists of one master device and one or more slave devices. Each slave device has unique 64 bits identification number [34].

One wire is used for communication with temperature sensors on measuring modules. One wire protocol describes four states: write logical 1, write logical 0, read and reset. Differences between these states are in period duration of low bus state. Selected MCU does not have periphery for One wire protocol, so it has to be emulated by another periphery. In application one wire is emulated using single GPIO pin. During transmission the GPIO pin is controlled for input mode or output mode.

Chapter 6
Software Design

Described software is firmware implemented in all three microcontrollers and provides the required and predefined functionality of whole aerometric system. Software handles communication between all boards using CAN bus (using interrupts) and also provides communication with sensors.

6.1 Overview

Developed software is stored in internal flash memory and during start-up of the MCU is loaded into RAM memory. Designed aerometric system is embedded based system. In this manner, implemented software can be divided into layers [Figure 6.1]. Hardware abstraction layer peripheral drivers (the libraries developed by MCU manufacturer) allow low-level operations in MCU using registers. At this level are configured interrupts, communication buses or clocking configuration [49].

For implementation was selected real-time operating system (RTOS) FreeRTOS, as demonstration of distributed measuring system with real-time properties [4]. Implemented application represents algorithms for defined behaviours.

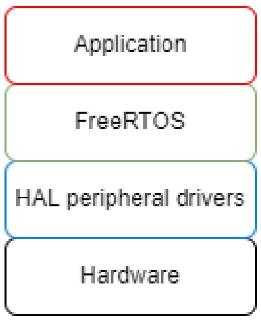


Figure 6.1: Software architecture.

6.1.1 Development Tools

The C source code was developed with the STM32CubeIDE toolchain [50] and generated code for HAL drivers. In code are placed comments for documentation purposes. For code debugging and programming was used the ST-LINK/V2 in-circuit debugger via the SWD interface. Software was build up by Arm GNU Toolchain compiler [12]. In case of any RTOS usage, a unique debugging technique called tracing is necessary. Tracing provides a view of individual tasks or events. Trace debugging was carried out by Percepio Tracealyzer [42]. Both measuring modules have the same source code, only user definition of CAN bus identifiers is different.

6.1.2 Common Functionality

In software for each module is implemented functionality called independent watchdog (IWDG), which is intended for device reset when a problem occurs, or as a free-running timer for application timeout management. It is configurable using low level registers.

IWDG includes internal timer that should be reloaded by software. In case that it reaches zero, the device will reset. There is software timer in application with period of 400 ms and timer callback function provides IWDG reload.

6.2 FreeRTOS

As was previously mentioned, FreeRTOS has multitasking feature that provides execution of concurrent tasks using single core MCU. Due to this FreeRTOS user-defined tasks require definition of priorities.

6.2.1 Task Management

FreeRTOS operates with multiple tasks, but only single task is executed at a time. FreeRTOS, as software middleware component, using scheduler switches between individual tasks based on their priority, state and preemption conditions. It allows concurrent task execution on a single processor core. Task states are depicted in [Figure 6.2].

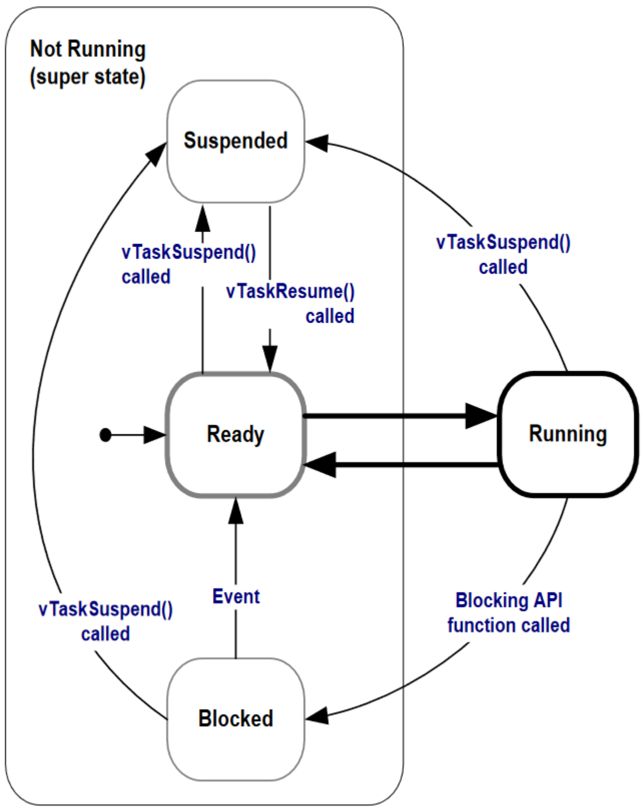


Figure 6.2: Task states [14]

From the application point of view, there are two states: running and not running. However, the not running state can be divided into:

- Suspend - task has been deactivated by application;
- Ready - tasks is ready for execution, but task with higher priority is running;
- Blocked - tasks is blocked and waits for synchronization event;

During blocked state, task does not consume any hardware resources and allows execution of the other tasks.

As scheduling algorithm was selected prioritized pre-emptive scheduling with time slicing. Pre-emptive means that running task with lower priority is moved to ready state and allow to task with higher priority to enter to the running state. Task with higher priority pre-empts task with lower priority. Time slicing shares processing time between tasks of the same priority, in case that the tasks do not yield or enter the blocked state. In terms of the FreeRTOS, function taskYIELD is for reschedule request.

6.2.2 Inter Process Communication

Inter process communication (IPC) provides data transfer between tasks and also provides synchronization events. In FreeRTOS for IPC can be used queues, message buffers, semaphores, mutexes, task notifications or event groups.

Queues are used for data transfer for multiple tasks and can provide synchronization events. Queues are working as FIFO buffers. In application are queues used for transfer received CAN frames to related tasks for further processing.

In case of pending CAN message in FIFO there is triggered interrupt callback function HAL_CAN_RxFifo0MsgPendingCallback. In interrupt routine each received frame from CAN FIFO is placed in back of queue. Queues have maximum defined size in application of 20 items.

```c
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan
    ) {
    /** \brief Required for xQueueSendToBackFromISR() */
    BaseType_t xHigherPriorityTaskWoken = pdFALSE;
    /** \brief Frame for received message */
    QUEUE_FRAME_t TxQueueData;
    /* Read mesage from FIFO 0 */
    if (HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0,
    &TxQueueData.Message, TxQueueData.bytes) == HAL_OK) {
        /* Put frame into back of queue */
        if (xQueueSendToBackFromISR(CAN_BUS_Queue,
        (void*)&TxQueueData,
        &xHigherPriorityTaskWoken) != pdPASS) {
            ErrorHandler();
        }
    }
    /* Switch to another task. */
    portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
}
```

Listing 6.1: Implementation of the CAN FIFO Callback function with placing message into queue.

Another task (e.g. can_bus_rx_task()) is in the blocked state because the function xQueueReceive() is indefinitely waiting for receiving queue. After receiving the queue item task is unblocked, the item is deleted from the queue. Items are received from the queue in the order that they have been sent.

```c
void can_bus_rx_task(void *argument) {
    QUEUE_FRAME_t Received_Queue;
    /** \brief for can bus check */
    /* Infinity loop for task */
    while (1) {
        /* Wait forever for queue receive CAN frames */
        /* Queue is received from CAN1 FIFO interrupts */
        xQueueReceive(CAN_BUS_Queue, &Received_Queue,
            portMAX_DELAY);
        /* etc... */
    }
}
```

Listing 6.2: Implementation of the call xQueueReceive for queue receiving and unblocking can_bus_rx_task.

Next type of synchronization event is provided using task notification. This feature allows sending of chosen 32 bits value from one task to another. Also, this notification value represents bit value, so more tasks can notify the same task and for values will be used OR function. Task that will obtain this value is waiting in blocked state. After receiving notification value, task is being unblocked. This notification values can be used for state machine implementation.

```
void can_request_rx_task(void *argument) {
    /* Some definitions, declarations etc.... */
    while (1) {
        /* Queue receive, data processing */
        xTaskNotify(synchronization_task_handler,
            notify_value_queue,
        eSetBits);
        /* etc... */
    }
}

void synchronization_task(void *argument) {
    /* Some definitions, declarations etc.... */
    while (1) {
    xTaskNotifyWait(0, (notify_value_queue |
        notify_value_individual), &interrupt_status,
        portMAX_DELAY);
    /* etc... */
    if ((interrupt_status & notify_value_queue) != 0) {
        /* some processing in queue mode */
    }
    else if ((interrupt_status & notify_value_individual) !=
        0) {
        /* some processing in individual mode  */
    }
}
```

Listing 6.3: Implementation of task notification (IPC). can_request_rx_task() notifies and unblocks synchronization_task with value that defines mode of operation.

Another types of task notification are event groups. They work in similar manner. Function xEventGroupWaitBits holds the task in blocked state, until predefined value (bit combination) is available. However, the above mentioned blocking functions stay in blocked state for defined time. Macro portMAX_DELAY will put them into blocked state indefinitely.

```
void can_data_rx_task(void *argument) {
    /* Some definitions, declarations etc.... */
    while (1) {
        /* Queue receive, data processing */
        /* Switch to delivered parameter */
    switch (Received_Queue.Message.StdId) {
        /* Absolute pressure module A */
        case ID_ABSOLUTE_PRESSURE_A:
            /* Save data from queue */
            if (receive_bytes(&Received_Queue,
            A_module_data.absolute_pressure)) {
                /* Set bits in event group */
                xEventGroupSetBits(xEventGroup_received_data,
                ABSOLUTE_PRESSURE_DELIVERED);
            }
            break;
        /* Differential pressure module A */
        case ID_DIFFERENTIAL_PRESSURE_A:
            /* Save data from queue */
            if (receive_bytes(&Received_Queue,
            /* Set bits in event group */
            A_module_data.differential_pressure)) {
                xEventGroupSetBits(xEventGroup_received_data,
                DIFFERENTIAL_PRESSURE_DELIVERED);
            break;
            }
        /* etc... */
    }
}

void synchronization_task(void *argument) {
    /* Some definitions, declarations etc.... */
    static EventBits_t uxBits_rx_data;
    while (1) {
    /* etc... */
    /* Queue mode */
    /* wait for max 100ms for set all bits in
        xEventGroup_received_data,
    result save in uxBits_rx_data */
    uxBits_rx_data = xEventGroupWaitBits(
        xEventGroup_received_data,
         ALL_DATA_DELIVERED, pdTRUE, pdTRUE, xMaxBlockTime_queue)
            ;
    /* etc... */
    }
}
```

Listing 6.4: Usage of event group in application. can_data_rx_task informs the synchronization_task of which data has been received.

6.3 Module Software

6.3.1 Measuring Module Software

Module software includes four tasks listed in [Table 6.2]. There is implemented binary semaphore functionality in software. This IPC is used to regularly unblock DATA_EVALUATION task. This task is used for handling data acquisition from sensors and also calculating other aerometric parameters.

Parameter	Note
Absolute pressure	Measured
Differential pressure	Measured
Temperature	Measured
Altitude	Calculated
TAS	Calculated
CAS	Calculated
VS	Calculated

Table 6.1: Parameters acquired in DATA_EVALUATION_TASK.

Unblocking is done using vTimerCallback task each 30 ms with priority value of 5. This callback is triggered by software timer FreeRTOS_TIMER.

Task name	Priority
Idle task	1
DATA_EVALUATION	4
CAN_RX	3
CAN_TX	3

Table 6.2: Tasks implemented in measuring module software.

However, after unblocking and executing the DATA_EVALUATION task in one iteration, at the beginning of the next iteration and calling function xSemaphoreTake blocks the execution of the task and allows the running of another task. That sequence provides the regular execution of DATA_-EVALUATION task. Also, in the same task is granted access for writing to memory (using DMA) space which stores measured and calculated aerometric parameters. Another task, CAN_TX task that is used for data transmission to main module, reads data from the same memory. To avoid conflict access to memory during writing or reading, the semaphore for mutual exclusion (mutex) is used.

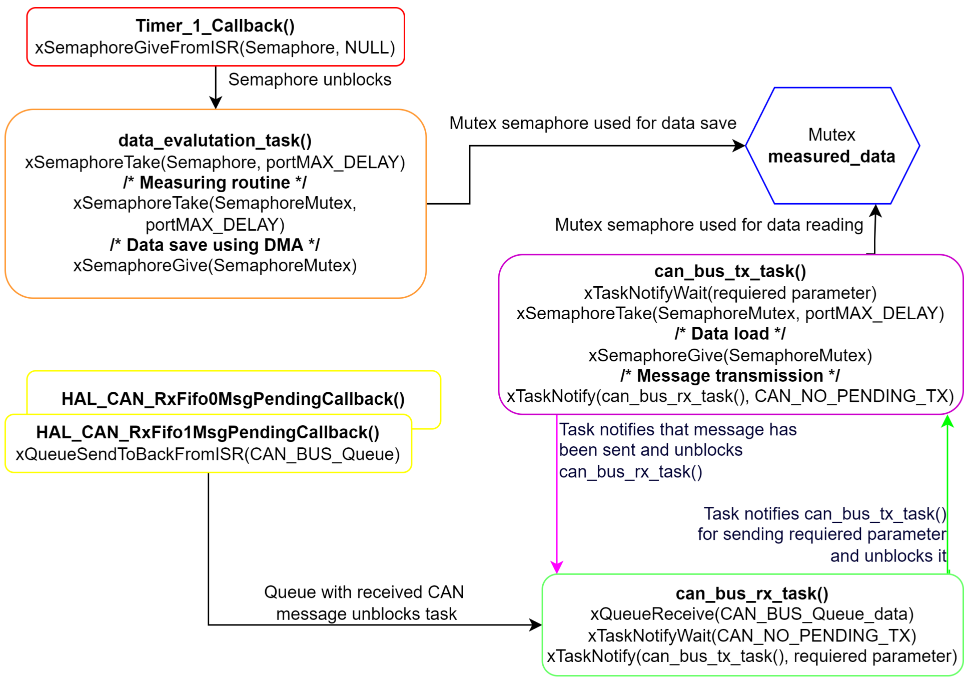

Timer_1_Callback()
xSemaphoreGiveFromISR(Semaphore, NULL)
Semaphore unblocks
data_evalutation_task()
xSemaphoreTake(Semaphore, portMAX_DELAY)
/* Measuring routine */
xSemaphoreTake(SemaphoreMutex, portMAX_DELAY)
/* Data save using DMA */
xSemaphoreGive(SemaphoreMutex)
Mutex semaphore used for data save
Mutex measured_data
Mutex semaphore used for data reading
can_bus_tx_task()
xTaskNotifyWait(requiered parameter)
xSemaphoreTake(SemaphoreMutex, portMAX_DELAY)
/* Data load */
xSemaphoreGive(SemaphoreMutex)
/* Message transmission */
xTaskNotify(can_bus_rx_task(), CAN_NO_PENDING_TX)
HAL_CAN_RxFifo0MsgPendingCallback()
HAL_CAN_RxFifo1MsgPendingCallback()
xQueueSendToBackFromISR(CAN_BUS_Queue)
Task notifies that message has been sent and unblocks can_bus_rx_task()
Task notifies can_bus_tx_task() for sending requiered parameter and unblocks it
Queue with received CAN message unblocks task
can_bus_rx_task()
xQueueReceive(CAN_BUS_Queue_data)
xTaskNotifyWait(CAN_NO_PENDING_TX)
xTaskNotify(can_bus_tx_task(), requiered parameter)

Figure 6.3: Algorithm of measuring module software.

When one task takes semaphore, the other task can not access the same resource until the first task gives back the semaphore. In case that the second task waits for semaphore, it holds block state. Additionally the mutex semaphore includes priority inheritance. This means that if a mutex is hold by lower priority task and task with higher priority is waiting for it (is in block state), then the priority of the task that holds mutex is raised to value of priority of blocked task. It solves priority inversion issue, because it is necessary to hold the higher priority task in block state for the shortest time.

CAN_RX task processes the received CAN frames that are delivered from interrupt by queue. Received single CAN frame with identifier represents request for given aerometric parameter. After the selection of the required parameter, task is put in blocked state and waits for notification from task CAN_TX (with CAN_NO_PENDING_TX value). It is used to inform that there is no pending transmission on CAN bus. After unblocking CAN_RX, task notification with related value is send to unblock CAN_TX task. This task handles the transmission of the required data to main module and after successful transmission it sends task notification (waiting for notification with CAN_NO_PENDING_TX value) to unblock CAN_RX task for next processing.

6.3.2 Main Module Software

Main module software incorporates 7 tasks [Table 6.3]. In hardware level are activated both CAN peripherals for internal and external buses. For internal CAN is used CAN 2 periphery (marked as slave) and for external CAN bus is configured CAN 1 (master CAN). For each periphery are assigned two FIFOs (using CAN filters). Received frames from interrupts are handled in related tasks - CAN1_REQUEST_RX and CAN2_DATA_RX.

Task name	Priority
Idle task	1
SYNCHRONIZATION	4
CAN1_DATA_TX	4
CAN2_DATA_RX	5
CAN1_REQUEST_RX	3
CAN2_REQUEST_TX	2
GPS_PARSE	3

Table 6.3: Tasks implemented in main module software.

Software works in two modes, queue or individual. The mode is selected by CAN identifiers of received request (frame with RTR bit from external CAN) by CAN1_REQUEST_RX task. Queue mode is selected by CAN identifier of 1499 and another received frames are individual requests for each parameter. Software also includes set of IPC tools and workflow is described by [Figure 6.4] In case of individual mode, the same request is send to internal CAN bus for measuring module using CAN2_REQUEST_TX task. After receiving data frame from measuring module (using CAN2_DATA_RX task) the frame is directly transmitted to external CAN bus (using CAN1_DATA_TX task).

When is set queue mode, after receiving request, the CAN2_REQUEST_-TX task sends the individual requests for all parameters to the internal bus. SYNCHRONIZATION task provides maximal 100 ms window (the worst case scenario) for synchronization by using xEventGroupWaitBits function. During this time, the task is in blocked state. After receiving all data, the task is unblocked and the data are transmitted to external bus using CAN1_-DATA_TX task. In case that blocking time expires, CAN1_DATA_TX task send all available data. This specific blocking is implemented only for aerometric parameters which are received from measuring modules. For parameters obtained from GNSS module is not used time blocking, because they are available always when the GNSS correctly acquire GNSS signal.

Data received from GNSS module are processed in the following way: Data from UART bus are transmitted into memory using DMA and circular buffer. In interrupt routine are data copied from circular buffer into MainBuffer. Another IPC function xMessageBufferSendFromISR transfers content from this buffer into GPS_PARSE task and unblocks that task. GPS_PARSE task explores, parses the required GPS frame and stores the data.

For storing all received data (from measuring modules and GNSS module) is used memory space that can be accessed by obtaining mutex semaphore.

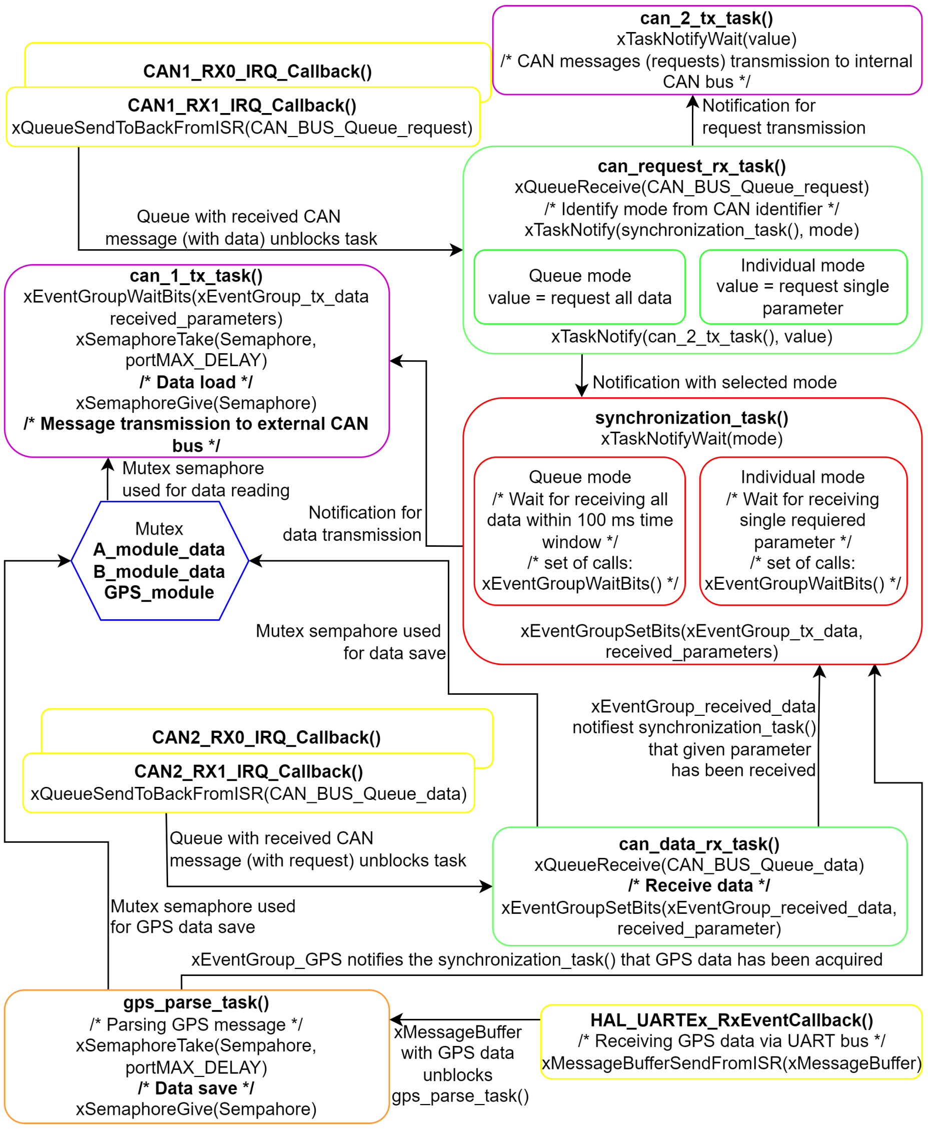

can_2_tx_task()
xTaskNotifyWait(value)
/* CAN messages (requests) transmission to internal CAN bus */
CAN1_RX0_IRQ_Callback()
CAN1_RX1_IRQ_Callback()
xQueueSendToBackFromISR(CAN_BUS_Queue_request)
Notification for request transmission
can_request_rx_task()
xQueueReceive(CAN_BUS_Queue_request)
/* Identify mode from CAN identifier */
xTaskNotify(synchronization_task(), mode)
Queue with received CAN message (with data) unblocks task
Queue mode value = request all data
Individual mode value = request single parameter
xTaskNotify(can_2_tx_task(), value)
can_1_tx_task()
xEventGroupWaitBits(xEventGroup_tx_data received_parameters)
xSemaphoreTake(Semaphore, portMAX_DELAY)
/* Data load */
xSemaphoreGive(Semaphore)
/* Message transmission to external CAN bus */
Notification with selected mode
synchronization_task()
xTaskNotifyWait(mode)
Queue mode /* Wait for receiving all data within 100 ms time window */ /* set of calls: xEventGroupWaitBits() */
Individual mode /* Wait for receiving single requiered parameter */ /* set of calls: xEventGroupWaitBits() */
xEventGroupSetBits(xEventGroup_tx_data, received_parameters)
Mutex semaphore used for data reading
Notification for data transmission
Mutex A_module_data B_module_data GPS_module
Mutex sempahore used for data save
xEventGroup_received_data notifiest synchronization_task() that given parameter has been received
CAN2_RX0_IRQ_Callback()
CAN2_RX1_IRQ_Callback()
xQueueSendToBackFromISR(CAN_BUS_Queue_data)
Queue with received CAN message (with request) unblocks task
can_data_rx_task()
xQueueReceive(CAN_BUS_Queue_data)
/* Receive data */
xEventGroupSetBits(xEventGroup_received_data, received_parameter)
Mutex semaphore used for GPS data save
xEventGroup_GPS notifies the synchronization_task() that GPS data has been acquired
gps_parse_task()
/* Parsing GPS message */
xSemaphoreTake(Sempahore, portMAX_DELAY)
/* Data save */
xSemaphoreGive(Sempahore)
xMessageBuffer with GPS data unblocks gps_parse_task()
HAL_UARTEx_RxEventCallback()
/* Receiving GPS data via UART bus */
xMessageBufferSendFromISR(xMessageBuffer)

Figure 6.4: Algorithm of measuring module software.

6.3.3 Testing

All developed algorithms were tested. As test module for request transmission to external bus was used NUCLEO-F446RE development board with the STM32F446RE MCU. Requests were transmitted with frequency of 10 Hz. Received data were written to COM port [Listing 6.5].

Another testing approach is in [Figure 6.5] and it shows trace log of measuring unit. Log depicts the mentioned events in order: DATA_EVALUATION task blocks on the next iteration (xSemaphoreTake (Semaphore #1 blocks)) and CAN_RX task receives the request (xQueueReceive(Queue #2)) and following task CAN_TX transmit the data. Log also shows sharing access for the memory space between tasks by Mutex.

```
GPS     Time:   18:35:39:80
GPS     Latitude:       50.10635
GPS     Height Above Ellipsoid: 158.10
GPS     Longitude:      14.58523
A       Aboslute Pressure:      98627.7
A       Differential Pressure:  74.2
A       Temperature:    37.25
A       Altitude:       227.50
A       TAS:    10.76
A       CAS:    0.29
B       Aboslute Pressure:      98862.5
B       Differential Pressure:  107.87
B       Temperature:    33.18
B       Altitude:       207.7
B       TAS:    13.10
B       CAS:    0.43
```

Listing 6.5: Received CAN messages (via COM port) containing data from modules in user (serial terminal) interface.

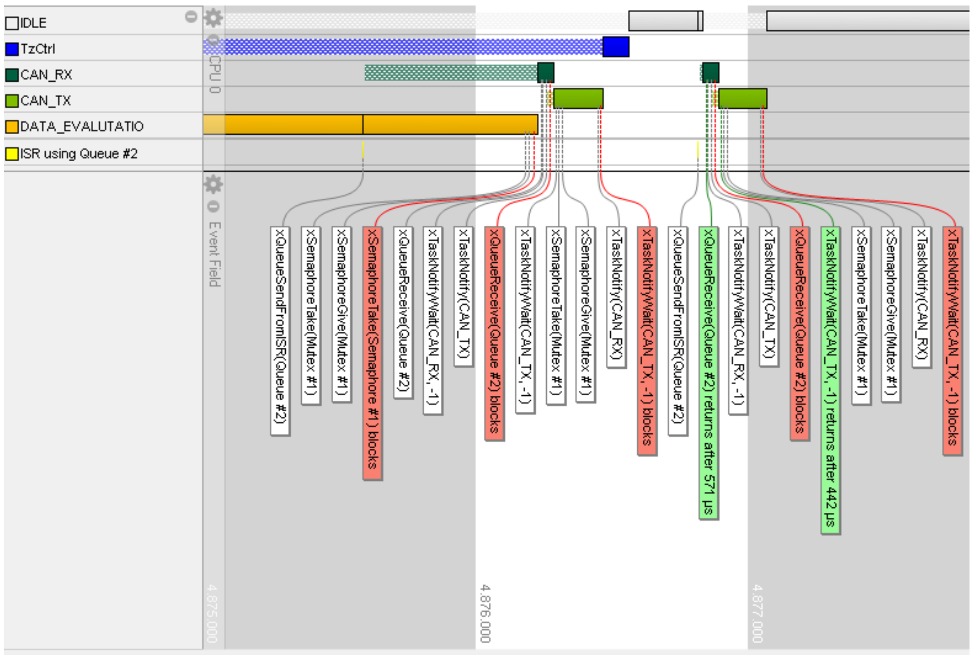

IDLE
TzCtrl
CAN_RX
CAN_TX
DATA_EVALUTATIO
ISR using Queue #2
CPU 0
Event Field
xQueueSendFromISR(Queue #2)
xSemaphoreTake(Mutex #1)
xSemaphoreGive(Mutex #1)
xSemaphoreTake(Semaphore #1) blocks
xQueueReceive(Queue #2)
xTaskNotifyWait(CAN_RX, -1)
xTaskNotify(CAN_TX)
xQueueReceive(Queue #2) blocks
xTaskNotifyWait(CAN_TX, -1)
xSemaphoreTake(Mutex #1)
xSemaphoreGive(Mutex #1)
xTaskNotify(CAN_RX)
xTaskNotifyWait(CAN_TX, -1) blocks
xQueueSendFromISR(Queue #2)
xQueueReceive(Queue #2) returns after 571 µs
xTaskNotifyWait(CAN_RX, -1)
xTaskNotify(CAN_TX)
xQueueReceive(Queue #2) blocks
xTaskNotifyWait(CAN_TX, -1) returns after 442 µs
xSemaphoreTake(Mutex #1)
xSemaphoreGive(Mutex #1)
xTaskNotify(CAN_RX)
xTaskNotifyWait(CAN_TX, -1) blocks
4.875.000
4.876.000
4.877.000

Figure 6.5: Trace log of measuring module.

Chapter 7

Test and Evaluation

7.1 Calibration Procedures

Assembled measuring modules were calibrated using Druck PASE6000 as pressure reference device [20].

For absolute pressure sensors was implemented in microcontrollers transfer function [Equation 7.1] prescribed by the manufacturer of the sensor [28].

$$P = \frac{(d - d_{min.})\; P_{max.} - P_{min.}}{d_{max.} - d_{min.}} + P_{min.}\; [Pa] \tag{7.1}$$

P – Pressure reading $[Pa]$; $d_{max.}$ – Output at maximum pressure $[counts]$ = 90 % of 2^{24} $counts$ = 0xE66666; $d_{min.}$ – Output at minimum pressure $[counts]$ = 10 % of 2^{24} $counts$ = 0x19999A; $P_{max.}$ – Maximum value of pressure range $[Pa]$ = 103421.3594 Pa; $P_{min.}$ – Minimum value of pressure range $[Pa] = 0$ Pa; d – Digital pressure reading $[counts]$;

The above mentioned pressure readings were measured [Figure 7.1] and related values of absolute error Δ_p are shown and approximated in [Figure 7.2]. Measured input pressure range represents altitude range from $-138,51$ m to $1457,30$ m considering ISA condition. Approximated values express linear function, which was implemented as calibration function in microcontrollers for A module [Equation 7.2] and for B module [Equation 7.3].

$$P_{Ac} = P_A - [(0.0018 \cdot P_A) - 90.4085]\; [Pa] \tag{7.2}$$

$$P_{Bc} = P_B - [(0.0021 \cdot P_B) - 120.7365]\; [Pa] \tag{7.3}$$

P_x – Absolute pressure reading for given module $[Pa]$; P_{xc} – Corrected absolute pressure reading for related module $[Pa]$;

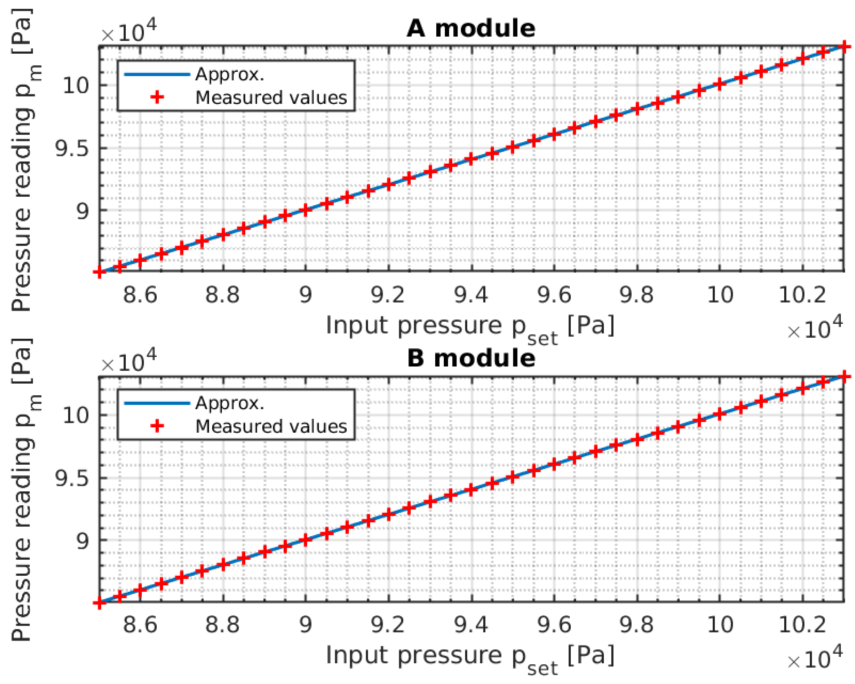


Figure 7.1: Measured absolute pressure.

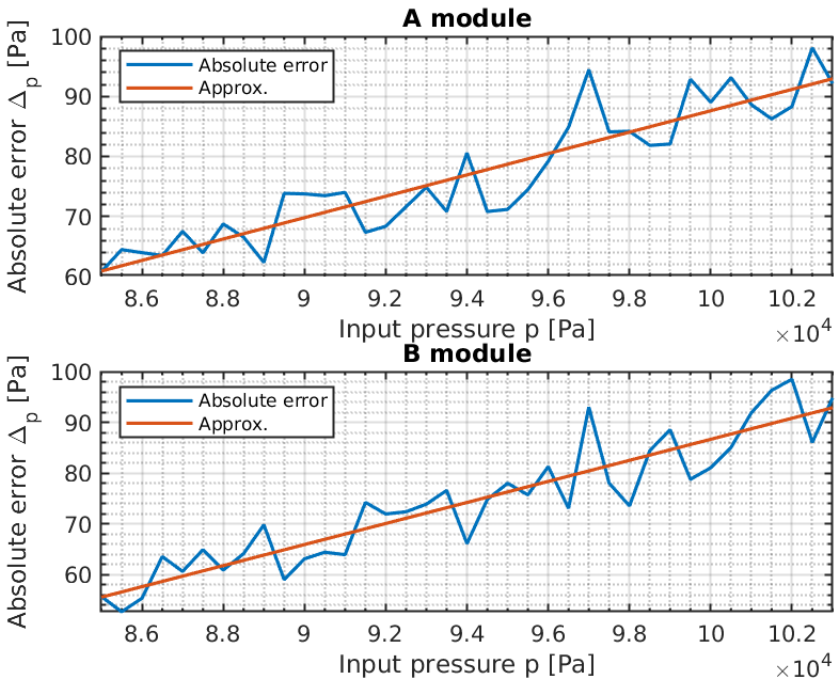


Figure 7.2: Absolute error of measured absolute pressure.

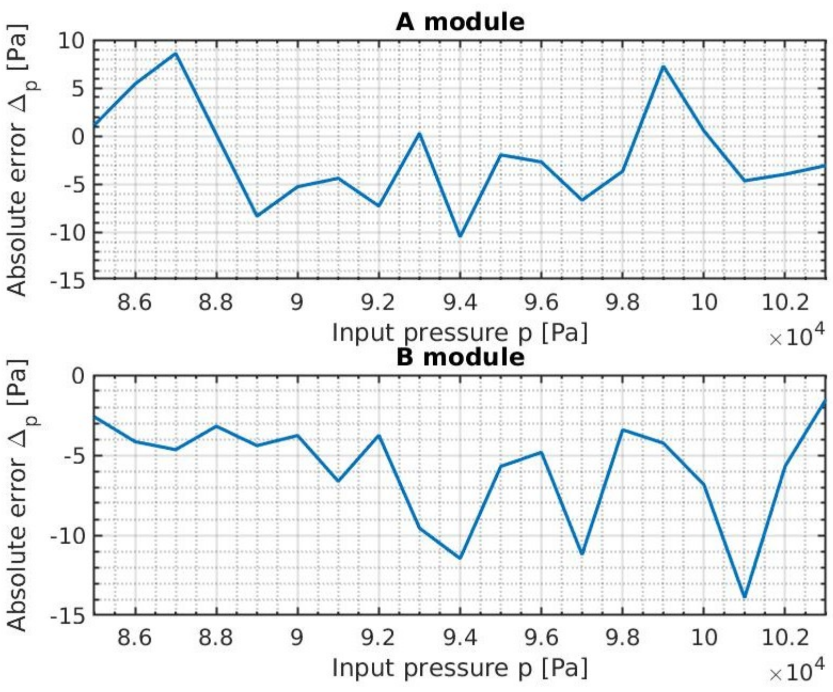


Figure 7.3: Absolute error after calibration.

After the calibration another measurement for validation has been carried out. Measurement results are in [Figure 7.3]. Calculated differences between measured and reference values of the absolute pressure are in the range defined by the manufacturer of the sensor: $\pm 1.5\,\%$ *Full Scale Span* $= 1551.32\,\text{Pa}$.

The related transfer function for differential pressure sensors was obtained by measuring with reference device. Measuring was carried out in whole range of the input differential pressure: 0 Pa - 2500 Pa. The output sensor reading for each sensor are depicted in [Figure 7.4]. [6][5].

Transfer functions for both sensors are determined by [Equation 7.4] and [Equation 7.5] [Figure 7.5].

$$P_A \;=\; d_A \;\cdot\; 0.0026 \;+\; 7.8467 \; [Pa] \tag{7.4}$$

$$P_B \;=\; d_B \;\cdot\; 0.0026 \;+\; 7.944 \; [Pa] \tag{7.5}$$

d_x − Digital differential pressure reading for related module $[counts]$; P_x − Differential pressure reading for given module $[Pa]$;

Module A has negligible deviations in differential pressure reading, and it means that calibration carried out by the manufacturer is satisfactory. For module B was used linear calibration function [Figure 7.6].

Deviation of differential pressure reading from reference pressure has been calculated from the validation measurement after the calibration. Also, there deviations are expressed in the related values of V_{TAS}. For V_{TAS} calculation was taken into account standard value of pressure $p_0 = 101\,325\,\text{Pa}$ [Figure 7.7].

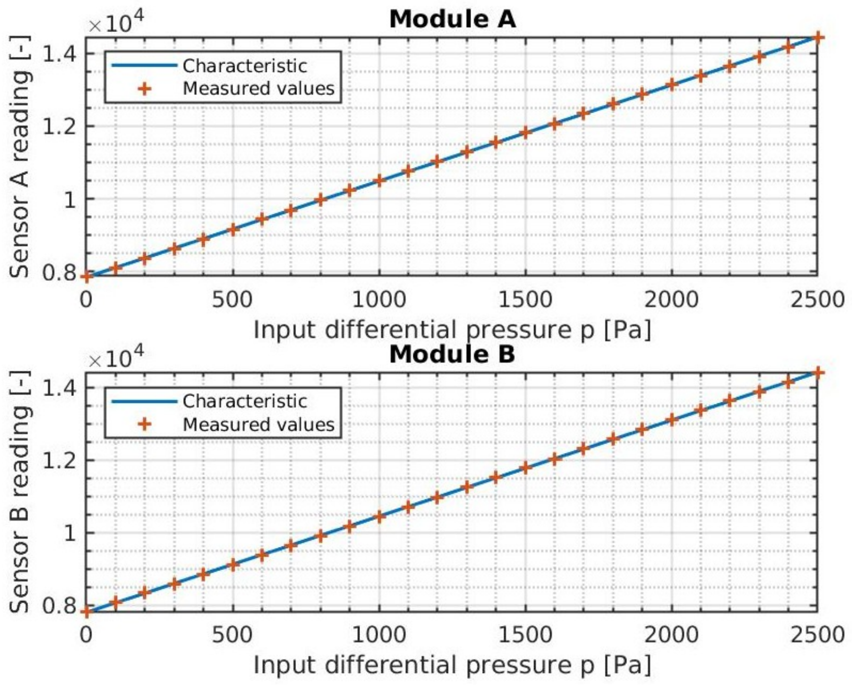


Figure 7.4: Differential sensor reading.

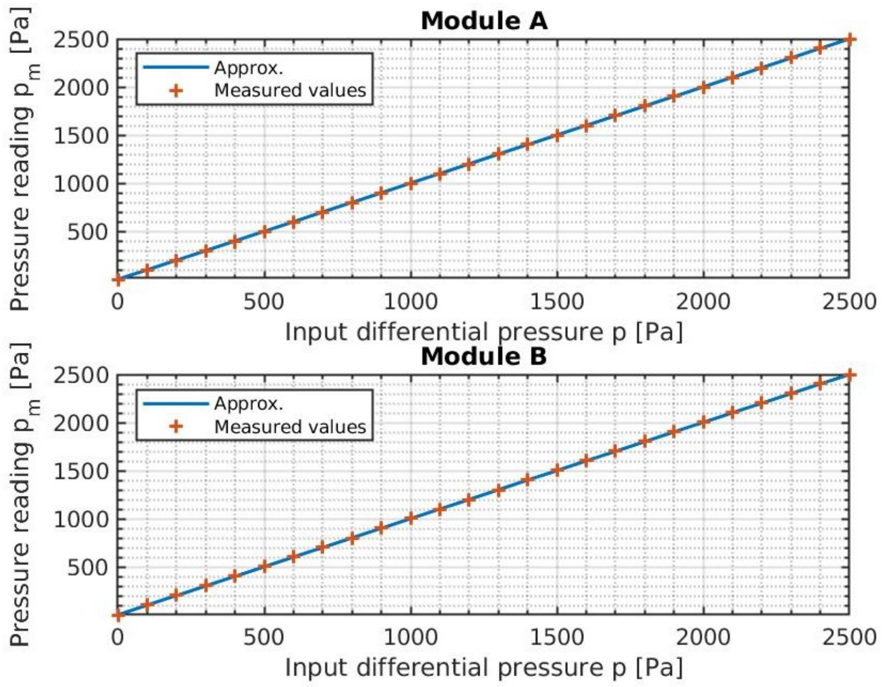


Figure 7.5: Measured differential pressure.

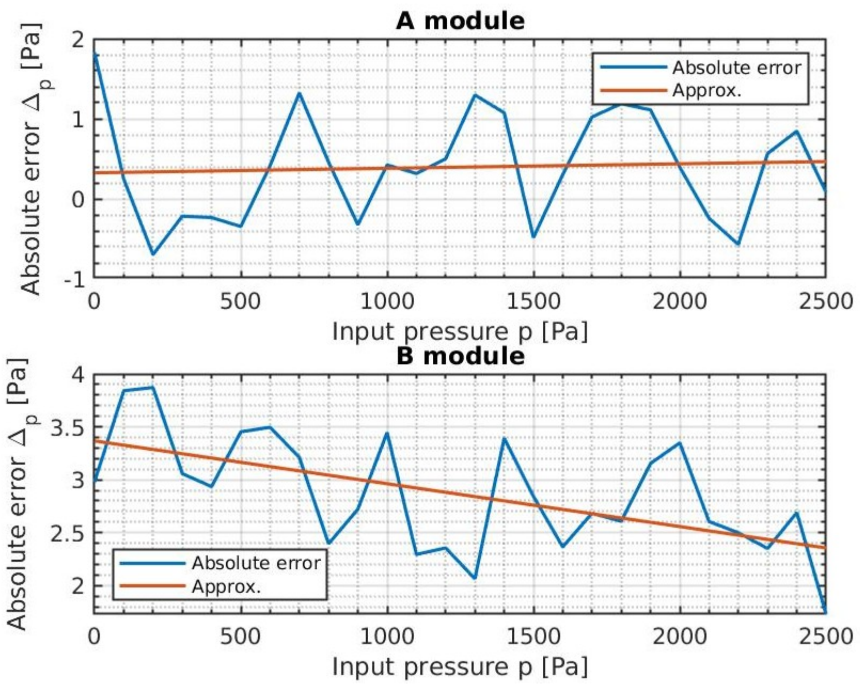


Figure 7.6: Calculated absolute error of measured differential pressure.

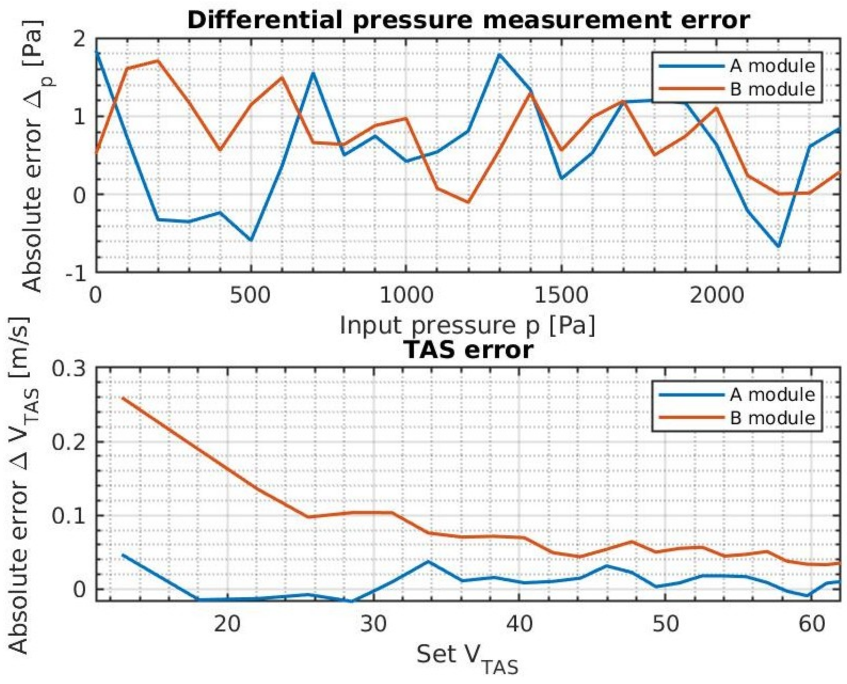


Figure 7.7: Calculated absolute error of measured differential pressure after calibration. Deviations are also expressed in the terms of V_{TAS}.

7.2 Vertical Speed Measurement

Measuring modules are also able to provide values of vertical speed. Vertical speed is derived from the rate of change in altitude. It could be expressed as the first derivative of height with respect to time. In case of the measuring module, it can be used 16 bit timer for time measurement between each calculation. This interval is connected with the pressure measurement cycle. Timer resolution is 1 ms.

```
void VS_calculation(float *Altitude, float *VS) {
    /** \brief previous value of altitude */
    static float last_value = 0;
    /** \brief elapsed time from the last calculation */
    uint16_t elapsed_time = __HAL_TIM_GET_COUNTER(&htm7);
    /* Set timer to zero */
    __HAL_TIM_SET_COUNTER(&htim7, 0);
    /* VS calculation, output value in [m/s]  */
    *VS = (float)((*Altitude - last_value)*1000)/elapsed_time;
    /* save VS for next calculation */
    last_value = *Altitude;
}
```

Listing 7.1: Implementation of the function for vertical speed derivation using timer.

However, the absolute pressure data are noisy ($\sigma = 5.8681\,\text{Pa}$). The calculated vertical rate is ambiguous and suitable noise reduction is necessary [Figure 7.8].

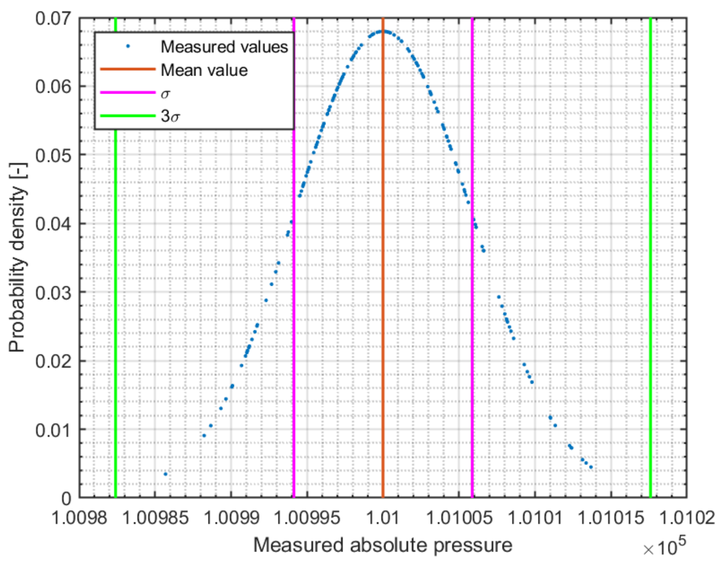


Figure 7.8: Probability density (distribution) of measured absolute pressure for set value = 1010 hPa.

For noise reduction the moving average with Gaussian kernel was selected. It means that Gaussian-weighted moving average over each data window is calculated. The Gaussian kernel function $K_\lambda(x_0, x)$ [Equation 7.6] has a weight function based on the Gaussian density function. Kernel function assigns weights to points x in region around x_0 [25].

$$K_\lambda(x_0, x) = \frac{1}{\lambda} \exp\left[-\frac{|x - x_0|^2}{2\lambda}\right] \tag{7.6}$$

λ – variance of the Gaussian density;

For calculation was used 1 dimensional Gaussian kernel function $G_\lambda(x)$ centered at the origin with standard deviation $\sigma = 1$.

$$G(x) = \exp\left[-\frac{|x - x_0|^2}{2\sigma^2}\right] \tag{7.7}$$

The filtering includes convolution, which provides smoothed data $F(x)$:

$$F(x) = G(x) * H(x) \tag{7.8}$$

$H(x)$ – original noise-corrupted data;

The above mentioned Gaussian filter is type of low-pass filter. Probability density of the smoothed data is show in and standard deviation is $\sigma = 2.0941$.

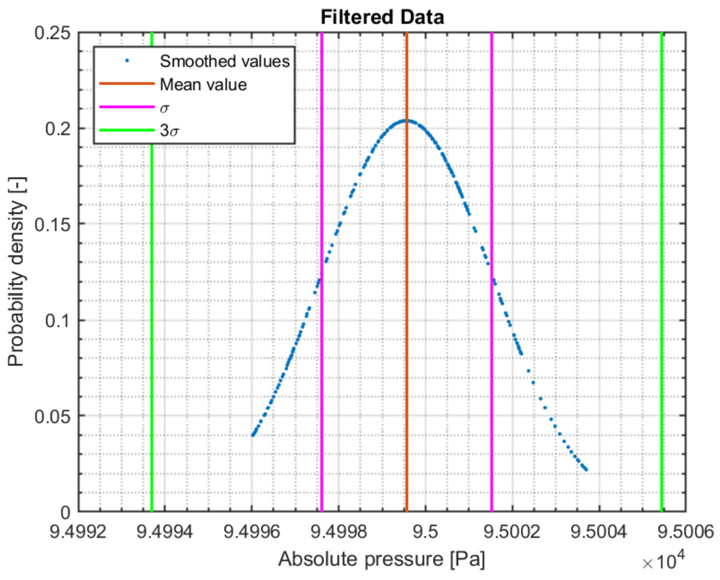


Figure 7.9: Probability density (distribution) of smoothed values of absolute pressure for set value = 950 hPa.

For VS measurement were simulated changes in absolute pressure using reference device. Simulation started from 27.09 m to 540 m with holding and back to 27.09 m. There were two test conditions - with rate 100 Pa s^{-1} (approx. 8.54 m s^{-1}) and 500 Pa s^{-1} (approx. 42.7 m s^{-1}).

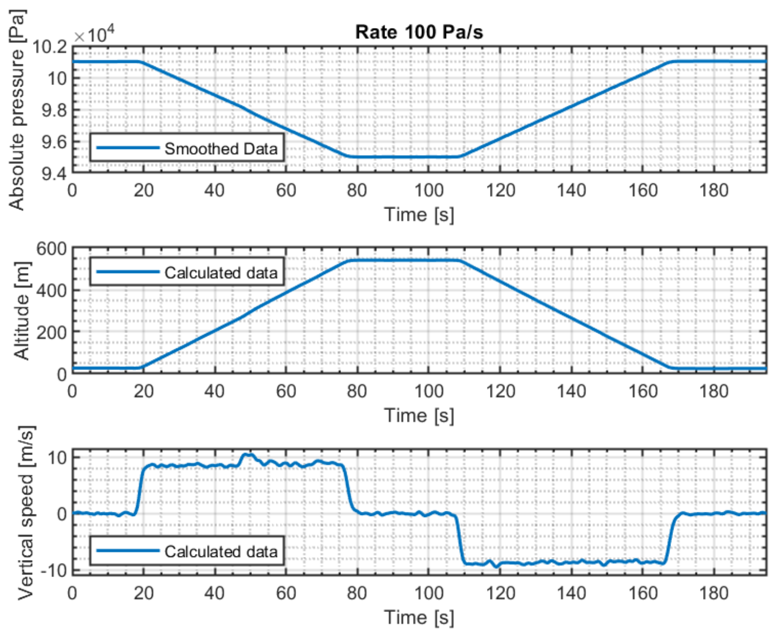


Figure 7.10: Simulated vertical speed measurement with rate 100 Pa s^{-1}.

Calculated value of VS for simulated rate 100 Pa was 8.5 m s^{-1}.

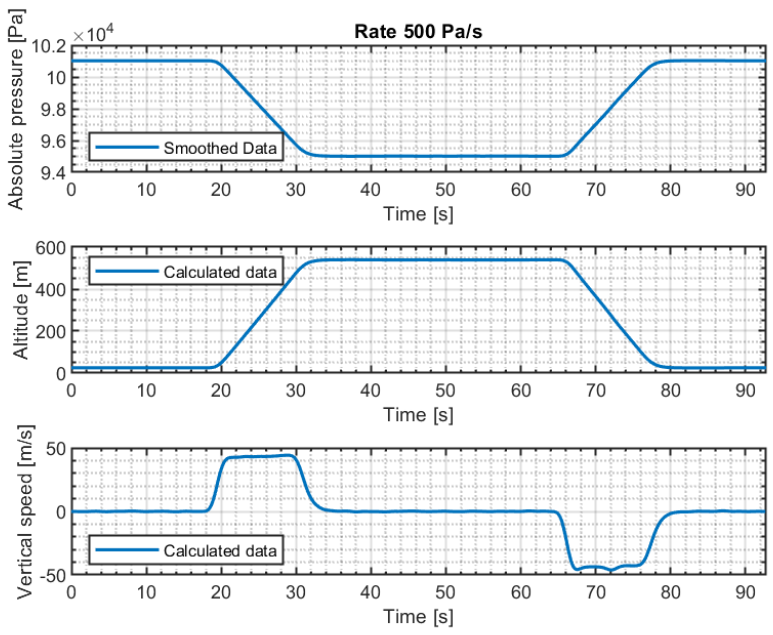


Figure 7.11: Simulated vertical speed measurement with rate 500 Pa s^{-1}.

Calculated value of VS for simulated rate 500 Pa was 43.1 m s^{-1}.

7.3 Environmental Test

Environmental testing in common means measuring the performance or operational parameters of a given device under specified environmental conditions, e.g. high or low temperatures. There was performed temperature test with a focus on selected pressure sensors. Evaluated parameters were measuring accuracy and operation capability.

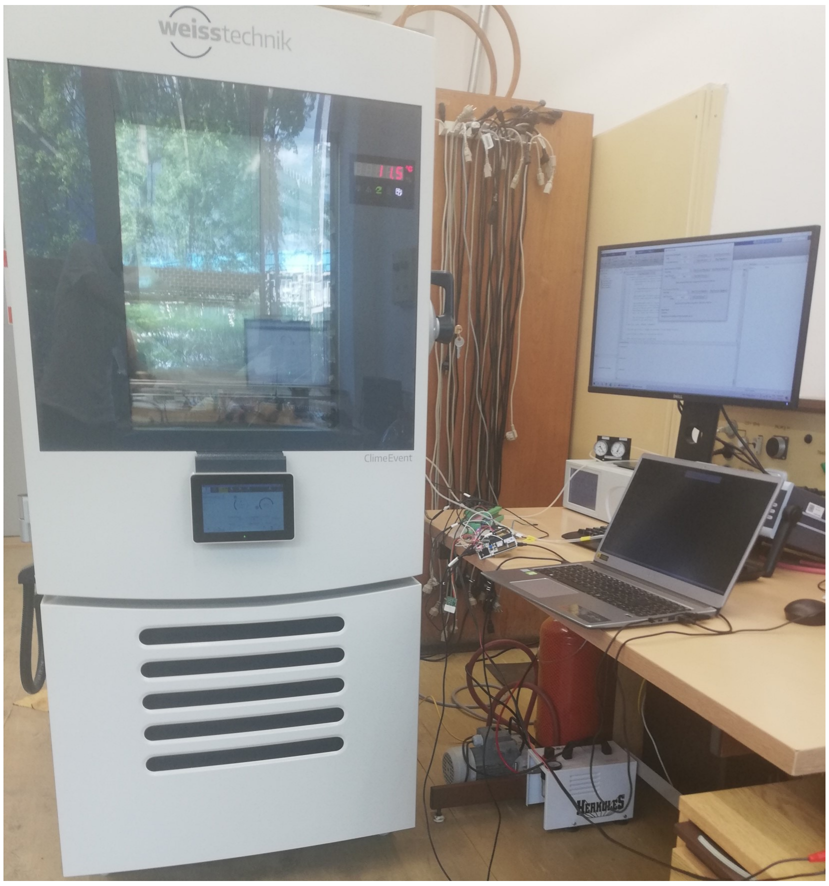


Figure 7.12: Laboratory setup with climatic chamber for the environmental test.

Only measuring module B was placed in a climatic chamber for testing. The differential pressure sensor has an operating temperature range of $-40\,°\mathrm{C}$ to $125\,°\mathrm{C}$ and compensated range of $0\,°\mathrm{C}$ to $60\,°\mathrm{C}$. The absolute pressure sensor has an operating temperature range of $-40\,°\mathrm{C}$ to $80\,°\mathrm{C}$ and compensated range of $0\,°\mathrm{C}$ to $85\,°\mathrm{C}$.

The test was carried out at temperatures $-20\,°\mathrm{C}$ and $60\,°\mathrm{C}$. In case that ambient temperature is lower than $5\,°\mathrm{C}$, the absolute pressure sensor provided a meaningless reading. At the given pressure set of $1000\,\mathrm{hPa}$, the ambient temperature $60\,°\mathrm{C}$ caused the difference in pressure reading $158\,\mathrm{Pa}$. This difference is equal to $\Delta H = 13.08\,\mathrm{m}$ [Figure 7.1].

A comparison of V_{TAS} measurements at given temperatures is in [Table 7.2]. Measured differences $\Delta V_{TASmax} = 0.27\,\mathrm{m\,s^{-1}}$ outside of calibrated temperature range are not significant. For calculation was considered standard pressure.

Ambient temperature [°C]	Measured absolute pressure [Pa]	Calculated altitude H [m]
25	100000	110.88
-20	invalid	n/a
60	100158	97.80

Table 7.1: Temperature test for absolute pressure transducer with reference absolute pressure 100 000 Pa. Altitude H values are calculated from measured absolute pressure.

Set difference pressure [Pa]	measured V_{TAS} at 25 °C [m/s]	measured V_{TAS} at -20 °C [m/s]	difference ΔV_{TAS} at -20 °C [m/s]	measured V_{TAS} at 60 °C [m/s]	difference ΔV_{TAS} at 60 °C [m/s]
100	12.75	12.54	0.20	12.47	0.27
200	18.06	17.88	0.17	17.86	0.20
300	22.12	22.01	0.11	21.96	0.15
400	25.53	25.43	0.09	25.40	0.12
500	28.53	28.44	0.08	28.43	0.10
600	31.25	31.15	0.10	31.15	0.10
700	33.77	33.66	0.11	33.66	0.10
800	36.08	35.96	0.11	35.96	0.11
900	38.28	38.17	0.11	38.16	0.12
1000	40.33	40.22	0.10	40.19	0.13
1500	49.36	49.23	0.13	49.21	0.14
2000	56.94	56.80	0.13	56.82	0.12
2400	62.32	62.17	0.15	62.19	0.12

Table 7.2: Temperature test of differential pressure transducer. Test results with differences expressed by related V_{TAS} values.

7.4 Wind Tunnel Testing

After the sensor calibration procedures, the aerometric system was tested in the wind tunnel at the Department of Fluid Mechanics and Thermodynamics of the Czech Technical University in Prague (The Faculty of Mechanical Engineering) [23].

Direct airflow sensing was carried out by two types of pitot-static probes [Figure 7.13]. The major difference between these two probes is that probe with a plastic part, placed in the entrance part of the total pressure port (probe PT - tube with plastic part), is more sensible for changes in the direction of the air stream (changes in α and β). The second probe (steel tube ST - without plastic part) has a wider entrance part of the total pressure port, and this probe is insensible for lower changes in the direction of the air stream. The above mentioned change can be approx. lower than 5 % in range of $\pm 20°$ in α and β.

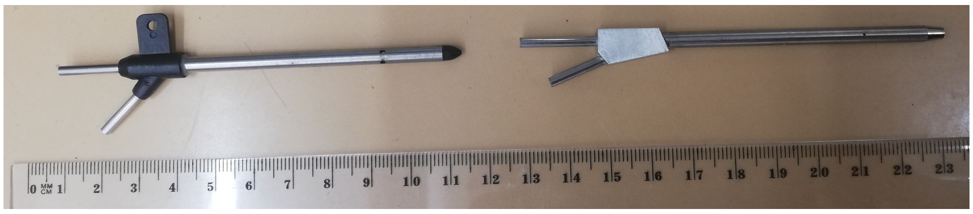


Figure 7.13: Pitot-static probes: left: PT (tube with plastic part); right: ST (steel tube - without plastic part).

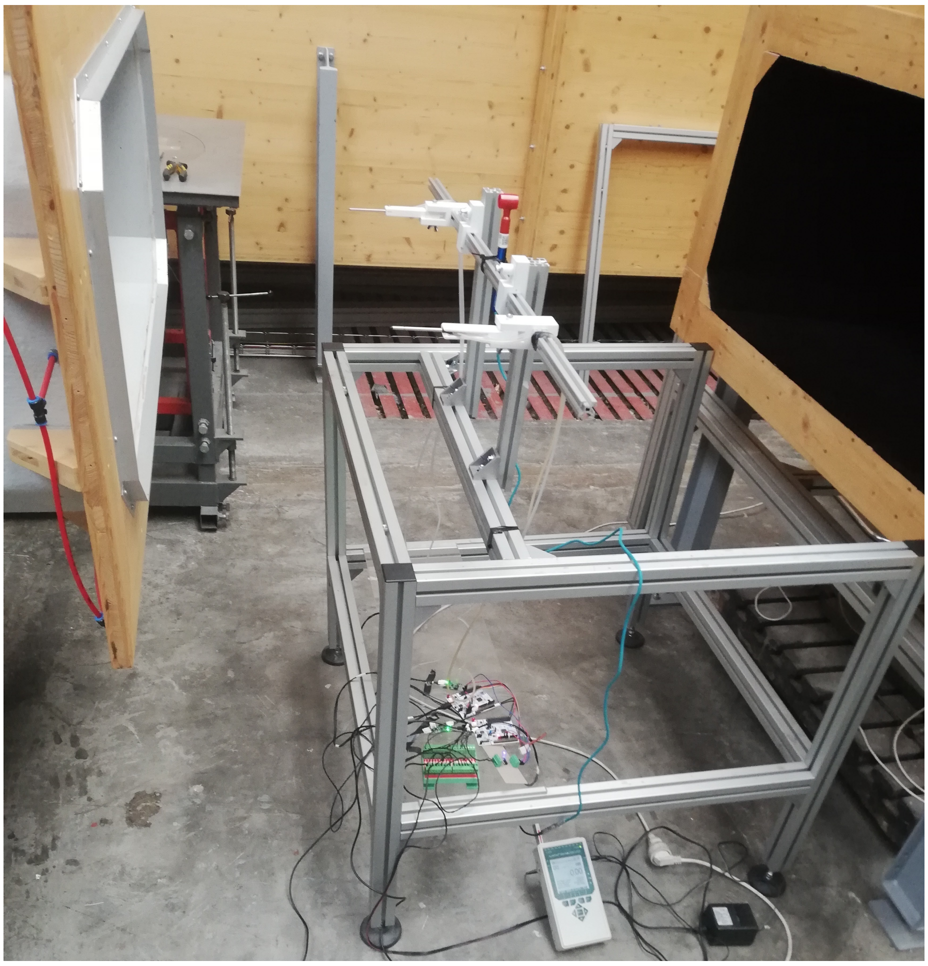


Figure 7.14: Measurement setup.

For testing procedures was used modular construction which provided mounting two pitot-static probes against air flow [Figure 7.14]. This setup was also used for measurement of the aerometric parameters in case of changes in β. Changes in β were simulated by rotating whole modular construction.

The following measurements has been carried out:

1. V_{TAS} measurement;

2. Experimental measurement of the aerometric parameters in case of changes in β.

7.4.1 True Airspeed Measurement

TAS measurement has been carried out using the basic setup as described in previous section. Test results are shown in [Figure 7.15, Figure 7.16]. Dynamic pressure was measured directly by differential pressure. Absolute error of dynamic pressure was calculated as difference between measured dynamic pressure and dynamic pressure related to reference speed. For reference measurement was used U-tube manometer with liquid ethanol placed in air flow.

Differences in V_{TAS} (obtained results) does not present the significant error: maximal $-1.2\,\mathrm{m\,s^{-1}}$ in measured range from $0\,\mathrm{m\,s^{-1}}$ to $36\,\mathrm{m\,s^{-1}}$ (approx. $3.33\,\%$).

Measured static pressure $p_s = 98\,367\,\mathrm{Pa}$. Module A was attached pitot-static probe PT and module B was attached probe ST.

Calculated differences between measured and reference values of the dynamic pressure Q_c [Figure 7.16] are in range described by the manufacturer of the differential pressure sensor: $1\,\%$ *Full Scale Output* $= \pm 75\,\mathrm{Pa}$ [6][5]. Also, these variations can be caused by the imperfections in the surfaces and geometry of the pitot-static probe, as well as by the turbulence levels during testing.

The sensor kit ASS-100 ADV [24] is equipped with 4525-DS3A001DP differential pressure sensor with a measuring range from $0\,\mathrm{Pa}$ to $6894.74\,\mathrm{Pa}$ with accuracy of $\pm 0.25\,\%$ and TEB of $\pm 0.1\,\%$. Assuming that accuracy and standard atmosphere condition ($H = 0\,\mathrm{m}$), the maximum difference in calculated true airspeed can be considered as approx. ± 4 meter/s. However, for further comparison with this commercial product, it is necessary to carry out laboratory tests focusing on device precision.

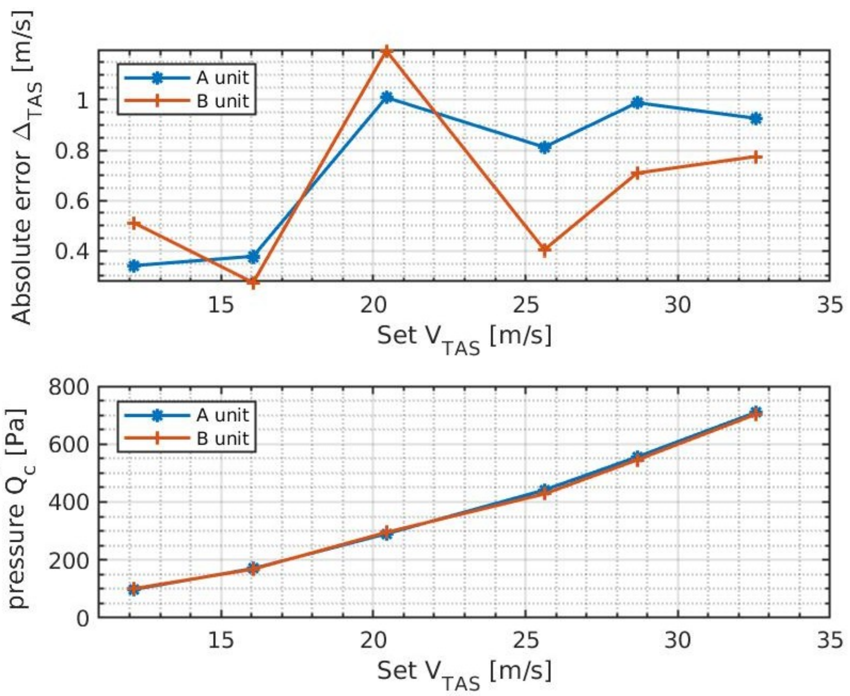


Figure 7.15: True airspeed measurement. The upper figure shows difference between measured and reference V_{TAS}. The lower figure includes values of measured dynamic (differential) pressure Q_c.

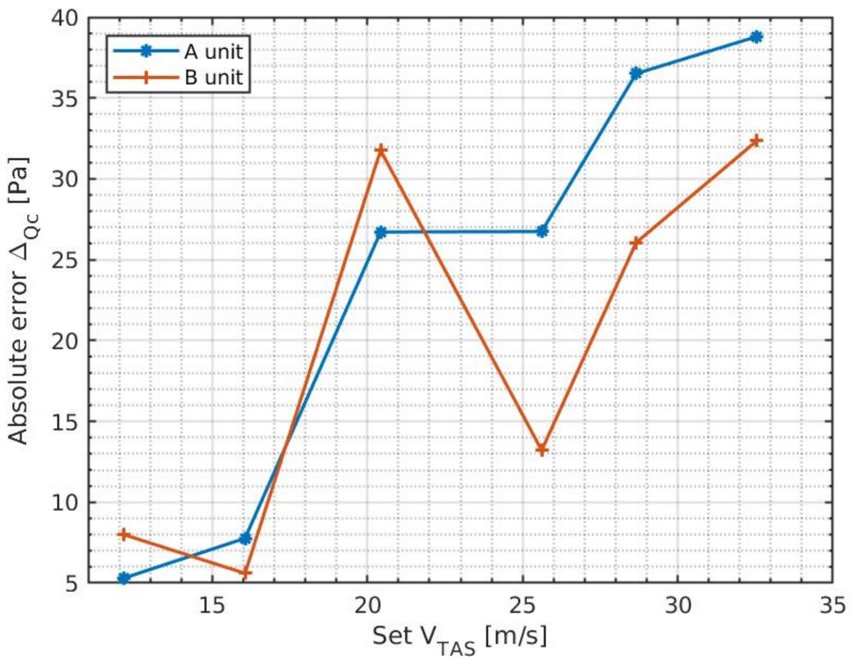


Figure 7.16: True airspeed measurement. Figure shows variations between measured dynamic pressure and pressure values which are related to reference V_{TAS}.

7.4.2 Experimental Measurement of the Aerometric Parameters

The following two measuring methods were experimental measurements of the aerometric parameters in case of changes in angle of sideslip β. In both methods was used only differential pressure sensor and measurement was carried out in four poistions of β: 0°, 10°, 21° and 28.5°, at two airspeed V_{TAS}: $16.05\,\mathrm{m\,s^{-1}}$ and $32.55\,\mathrm{m\,s^{-1}}$.

In the first method, both total pressure ports were connected to the same differential pressure and one pitot-static probe was rotated 38.66° on mounting against direction of airflow [Figure 7.17]. Pitot-static probe with no rotation in starting position ($\beta = 0°$) was connected to positive port on sensor because there is supposed to be measured higher pressure during experiment than in another probe. The measured parameter was differential pressure in each position and airspeed. The goal was to determine relation between angle of sideslip and the mentioned differential pressure on total ports of pitot-static probe in this setup. For measurement were used the same pitot-static probes as in previous measurement [Figure 7.13].

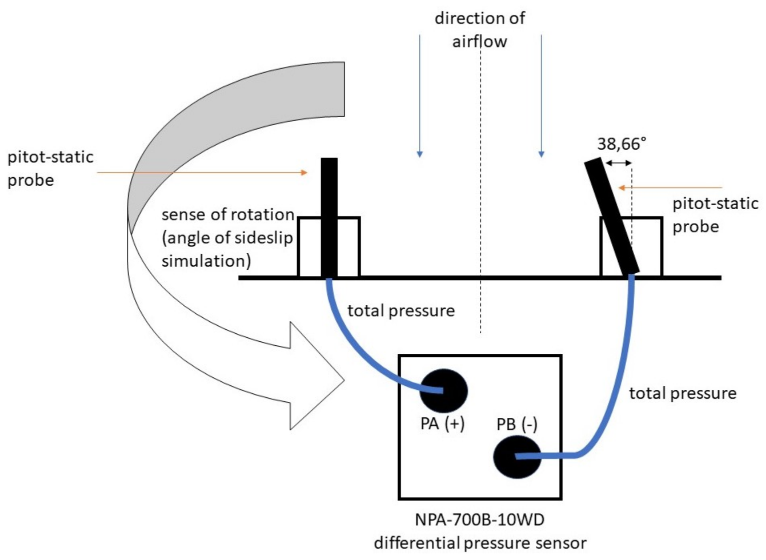


Figure 7.17: Experimental measurement setup - method 1.

Measurement results from the first method are shown in the [Figure 7.18] and [Figure 7.19]. For comparison, the results are expressed in terms of the ratio p_Δ/Q_c. Q_c is value of the dynamic pressure at $\beta = 0°$ at given airspeed and p_Δ is measured value of the differential pressure.

As was mentioned in the previous subsection, the pitot-static probe PT is more sensible for changes in direction of the air stream than probe ST, and in this case it is obvious from the results in terms of the ratio p_Δ/Q_c as was expected.

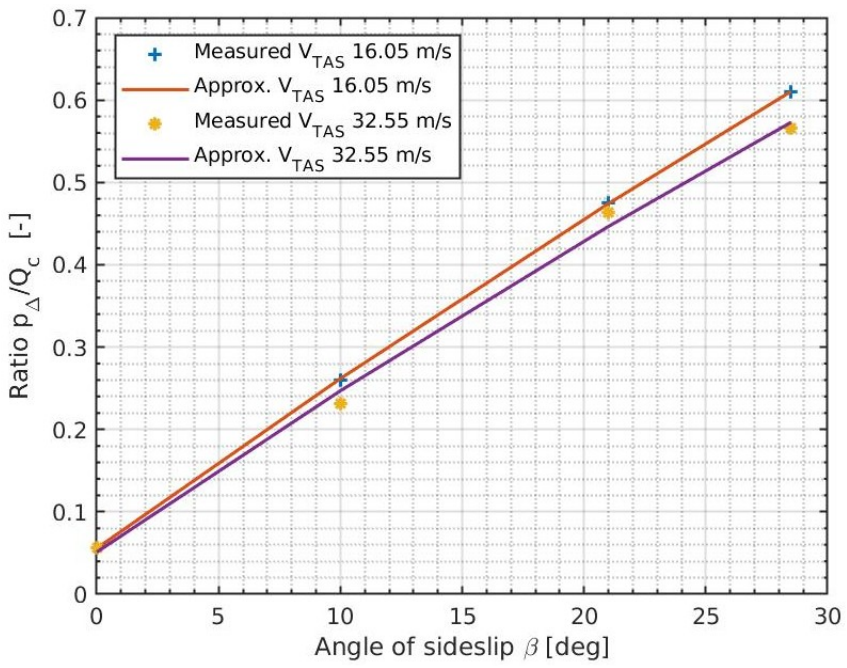


Figure 7.18: Measured only with pitot-static probes PT at two airspeed.

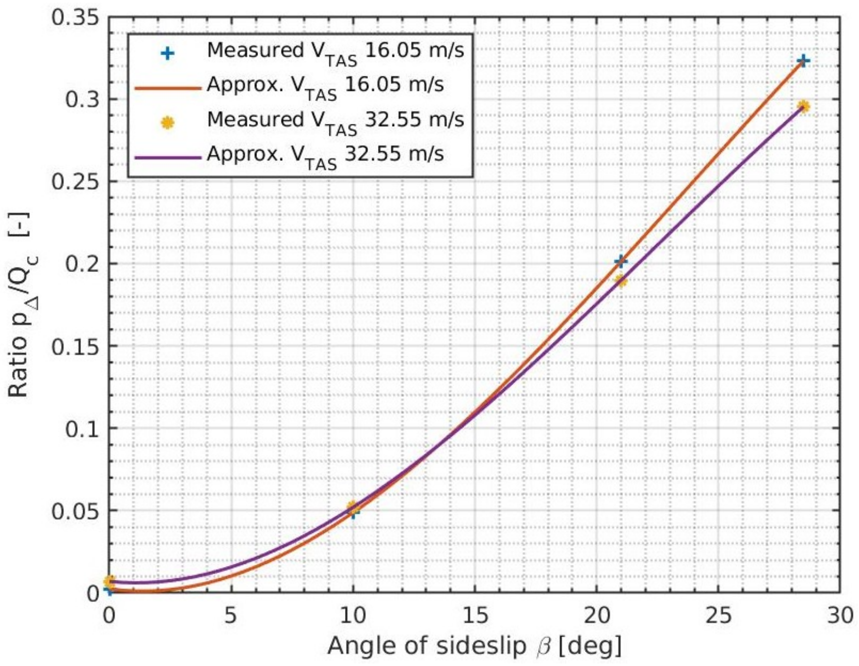


Figure 7.19: Measured only with pitot-static probes ST at two airspeed.

In the second method, static pressure ports were connected to the same differential pressure and only one pair of the pitot-static probes NP was used [Figure 7.20]. On both pitot-static probes were sealed inner static pressure holes. Pitot-static probe PT was not used, because both types of probes have the same position of the static pressure holes and total pressure measurement is not taken into account in this case.

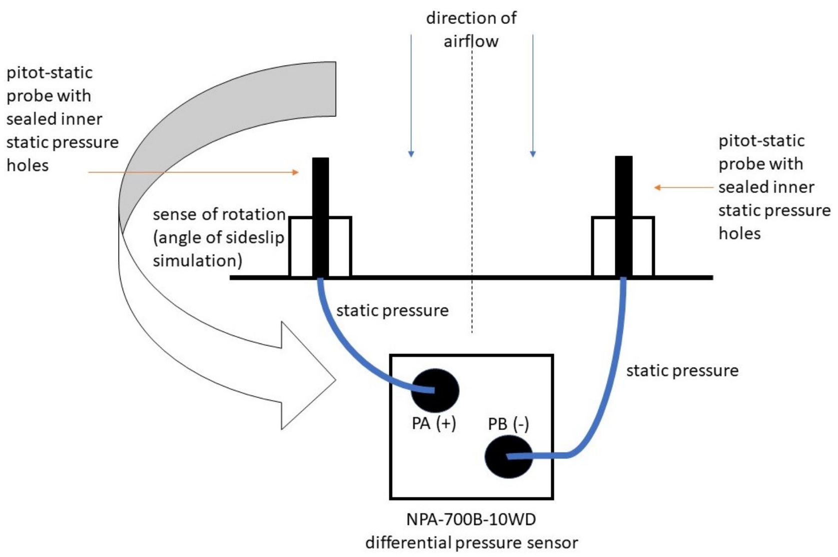

direction of airflow
pitot-static probe with sealed inner static pressure holes
sense of rotation (angle of sideslip simulation)
pitot-static probe with sealed inner static pressure holes
static pressure
static pressure
PA (+)
PB (-)
NPA-700B-10WD differential pressure sensor

Figure 7.20: Experimental measurement setup with ST probes - method 2.

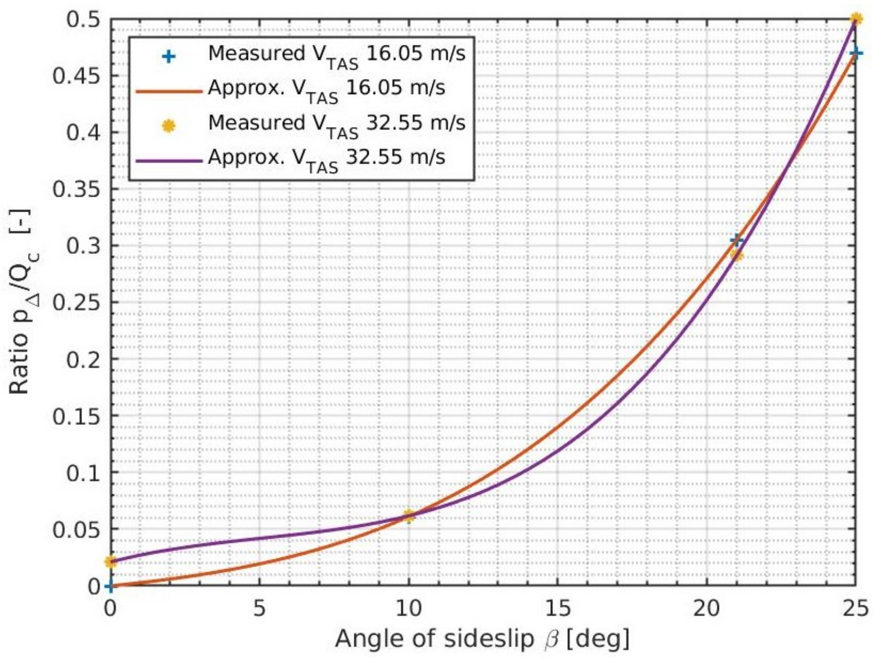


Figure 7.21: Measured only with pitot-static probes *ST*.

Measurement results from the second method are shown in the [Figure 7.21] and they are expressed similarly as in the previous method. In obtained results the relation between angle of sideslip and ratio p_Δ/Q_c, the approximated values for the ratio p_Δ/Q_c in range $\beta = 0°$ to $10°$, can not be assumed as realistic. Therefore the pitot-static probes are designed and manufactured for precise measurement of static pressure in lower changes of the direction of air flow. It means that changes in static pressure measuring should not be significant and smooth in the mentioned range.

7.4.3 Angle of Attack Measurement

The original idea included purchasing the pitot-static probe for measuring aerodynamic angles. However, finally were obtained only standard pitot-static probes, but designed measuring modules can be used with pitot-static probe that is also dedicated for angle of attack measurement. There were not a possibility to measure angle of attack by using any of the previously mentioned methods. There are few scientific papers dedicated to the angle of attack measurement by pitot-static system. In subsection 2.1.3 were described and analysed differential pitot-static probes with hemispherical nose shape for determination aerodynamic angles.

Chapter 8
Conclusion

The main goal of master thesis was design and practical realization of the aerometric system for unmanned aerial vehicle with fixed wings. Proposed and developed system consists of the two measuring modules and main module and is capable of measuring and providing real time aerometric data as barometric altitude, arispeed (TAS, CAS) vertical speed, sideslip. The main unit synchronizes these data with other GNSS parameters provided from connected GNSS module. System was tested using aerodynamic tunnel for simulation of real operation. Measuring modules are small-sized and can be directly mounted into aircraft wing to short connection with pitot-static probe.

For general description of the essential knowledge as aerometric parameters and systems in chapter 1 was used literature with focus on aerodynamics and aircraft systems. Next chapters in details specify the hardware (parts selection, boards design) and software development. Software implementation includes usage of the real time operating system FreeRTOS and advanced debugging techniques. Hence the system includes many communication protocols and buses, one chapter is dedicated to their description, especially to the CAN bus with CANaerospace protocol that is the major interface in system.

Measuring modules with pressure sensors were tested with focus on accuracy and temperature variations. The results were used for calibration. Based on measured values, the differences are negligible and they confirm the right selection of the absolute and differential pressure transducers for application in aerometric system. Test approach using aerodynamic tunnel shows that system is ready for practical usage. Maximum measured absolute error in true airspeed was just $-1.2\,\mathrm{m\,s^{-1}}$ in measured range from $0\,\mathrm{m\,s^{-1}}$ to $36\,\mathrm{m\,s^{-1}}$. After analysis, the Gaussian-weighted moving average was applied during data post-processing to noised absolute pressure data. Further calculation of vertical speed of de-noised data showed the correct rates, which has been simulated.

Another test approach was environmental temperature test using climatic chamber at $-20\,°\mathrm{C}$ and $60\,°\mathrm{C}$. Based on test results, the differential pressure sensor can be used with slight difference ($\Delta V_{TASmax} = 0.27\,\mathrm{m\,s^{-1}}$) outside of specified temperature-compensated range.

Another testing included experimental measuring using two types of pitot-static probes oriented in specific directions with focus on angle of sideslip. Test results determine exact relation between measured differential pressure and angle of sideslip. This approach could be a subject of further research. Using this approach, it is necessary to measure aerometric data at more values of angle of sideslip.

Measuring of angle of attack was not proven after discussion, however it was analysed. The designed system can be used for measuring aerodynamic angles using special differential pitot-static probe.

This prototypical aerometric system can be used for flight tests in real unmanned aircraft after installation and mounting. Another useful usage is for laboratory lesson related to aircraft instrumentation and also for further experimental various measurements in aerodynamic tunnel.

8.1 Future Work and Development

There are possible improvements and possible development regarding the realised system. In the case of board design, it is necessary to redesign measuring modules for moving temperature sensor *DS18B20* to another place, which has no impact on the temperature measurement process. Also, the thermal analysis of the board should be proven. The following design possibility is to remove useless capacitors to save space on board. Regarding board design, the modification of the schematic with the filter circuit can be analysed to remove the AC component from the power supply circuit. Also, analysis of the power input circuit behaviour during a voltage drop or rise is necessary.

Additionally, the testing focused on electromagnetic compatibility can be carried out, because the individual parts of aircraft avionics should not interfere. Probably there are not any sources of electromagnetic interference with significant impact.

Regarding software development, the One wire protocol can be emulated using the UART bus. For optimal synchronisation process can be used UTC time or clock signal provided by GNSS module - *pulse per second.* Also, the Gaussian-weighted moving average can be implemented as real-time functionality for denoising absolute pressure reading.

Another possibility is real-time temperature correction of pressure readings based on temperature sensor measurement. However, environmental testing at more temperature points is necessary for temperature compensation.

The system can also calculate and provide the calibrated airspeed, but it is necessary to measure accuracy in further laboratory tests with a reference device.

The graphical user interface (GUI) can be developed for further optimise the testing procedures. That GUI can provide communication (e.g. via UART bus) with the test module or with the CAN bus transceiver.

Bibliography

[1] *EASA Part-66 Module M13 B2 Study book - Aircraft Structures and Systems*. Aircraft Technical Book Co., 2016, ISBN 9781941144183, 650 p.

[2] ALTHEN SENSORS & CONTROLS: Air Data Probes. [Online; accessed 2021-10-19].
URL: `https://www.althensensors.com/measurement-systems/pressure-temperature-measurement-systems/aeroprobe/5584/air-data-probes/`

[3] Altium: Altium Designer. [Online; accessed 2022-03-20].
URL: `https://www.altium.com/`

[4] Amazon Web Services: The FreeRTOS™ Reference Manual API Functions and Configuration Options. [Online; accessed 2022-03-20].
URL: `https://www.freertos.org/fr-content-src/uploads/2018/07/FreeRTOS_Reference_Manual_V10.0.0.pdf`

[5] Amphenol Advanced Sensors: *NPA Surface-Mount Pressure Sensor Series*. 2 2019, [Online; accessed 2022-03-13].
URL: `https://f.hubspotusercontent40.net/hubfs/9035299/Documents/AAS-920-477J-NovaSensor-NPA-SurfaceMnt-013019-web.pdf`

[6] Amphenol Advanced Sensors: *NPA Series | Pressure Sensors, Application Guide*. 8 2021, [Online; accessed 2022-03-13].
URL: `https://f.hubspotusercontent40.net/hubfs/9035299/Documents/AAS-910-289F-030315-web.pdf`

[7] anon.: [Online; accessed 2021-10-19].
URL: `https://inairaviation.com/wp-content/uploads/2016/09/adcs.png`

[8] anon.: [Online; accessed 2022-03-20].
URL: `https://static6.arrow.com/aropdfconversion/arrowimages/a69398e8c3c8034f445ce2ab16cf54a67048157c/npa2barbedports.jpg`

[9] anon.: [Online; accessed 2022-03-20].
URL: `https://sigma.octopart.com/146303909/image/Molex-51382-0500.jpg`

[10] anon.: [Online; accessed 2022-03-20].
URL: `https://sigma.octopart.com/146307033/image/Molex-55935-0410.jpg`

[11] ARM: Cortex-M4. [Online; accessed 2022-03-20].
URL: `https://developer.arm.com/Processors/Cortex-M4`

[12] ARM: GNU Toolchain. [Online; accessed 2022-03-20].
URL: `https://developer.arm.com/Tools%20and%20Software/GNU%20Toolchain`

[13] Avionics Anonymous: UAVCAN Air Data Computer. [Online; accessed 2021-10-19].
URL: `https://www.avionicsanonymous.com/product-page/microadc`

[14] Barry, R.: Mastering the FreeRTOS™ Real Time Kernel A Hands-On Tutorial Guide. [Online; accessed 2022-03-20].
URL: `https://www.freertos.org/fr-content-src/uploads/2018/07/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf`

[15] Bennett, C.; Lawson, N.; Gautrey, J.; aj.: CFD simulation of flow around angle of attack and sideslip angle vanes on a BAe Jetstream 3102 – Part 1. *Aerospace Science and Technology*, volume 68, 2017: p. 561–576, ISSN 1270-9638, doi:https://doi.org/10.1016/j.ast.2017.03.015.
URL: `https://www.sciencedirect.com/science/article/pii/S1270963816305831`

[16] BOSCH: CAN Specification, Version 2.0. [Online; accessed 2022-03-20].
URL: `http://esd.cs.ucr.edu/webres/can20.pdf`

[17] Collins Aerospace: Wing-mounted angle-of-attack probe. [Online; accessed 2021-10-20].
URL: `https://www.aeroexpo.online/prod/collins-aerospace-utc-aerospace-systems/product-170790-62971.html`

[18] Collinson, R. P. G.: *Introduction to Avionics Systems, Second Edition*. Springer US, 2003, ISBN 978-1-4419-7466-2, 492 p., doi:10.1007/978-1-4419-7466-2.
URL: `https://www.springer.com/gp/book/9781402072789`

[19] Crossbow Technology: MNAV100CA User´s Manual, Revision F. 6 2007, [Online; accessed 2022-03-20].
URL: `https://www.yumpu.com/en/document/read/18713372/mnav100ca-users-manual-crossbow-technology`

[20] Druck: Modular Pressure Controllers. [Online; accessed 2022-03-20].
URL: https://www.bakerhughes.com/druck/test-and-calibration-instrumentation/pressure-controllers-pace

[21] Dynon Avionics: AOA / PITOT PROBES. [Online; accessed 2021-10-19].
URL: https://dynonavionics.com/aoa-pitot-probes.php

[22] European Organisation for the Safety of Air Navigation EUROCONTROL: REVISION OF ATMOSPHERE MODEL IN BADA AIRCRAFT PERFORMANCE MODEL, EEC Technical Report No. 2010-001, Project: BADA. 2010, [Online; accessed 2021-10-20].
URL: https://www.eurocontrol.int/sites/default/files/library/001_Revision_of_BADA_atmosphere_model.pdf

[23] Faculty of Mechanical Engineering, Czech Technical University in Prague: Department of Fluid Dynamics and Thermodynamics, Devices. [Online; accessed 2022-03-06].
URL: https://www.fs.cvut.cz/en-ustavy/en-sekce-ustav-mechaniky-tekutin-a-termodynamiky/en-ustav-mechaniky-tekutin-a-termodynamiky-12112/en-odborna-cinnost-12112/en-vybaveni-12112/en-pristroje-12112/

[24] FrSky Electronic: ASS-70/ASS-100 Air Speed Senor Instruction Manual. [Online; accessed 2022-03-20].
URL: https://www.frsky-rc.com/wp-content/uploads/2017/07/Manual/ASS-70%20ASS-100%E8%AF%B4%E6%98%8E%E4%B9%A6.pdf

[25] Hastie, T.; Tibshirani, R.; Friedman, J.: *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*. Springer New York, NY, second edition, 2008, ISBN 978-0-387-84858-7, doi:https://doi.org/10.1007/978-0-387-84858-7.

[26] Honeywell: MicroPressure MPR Series long port highres photo. [Online; accessed 2022-03-20].
URL: https://s7d1.scene7.com/is/image/Honeywell65/sps-siot-mpr-series-long-port-highres-photo-ciid-172518

[27] Honeywell: *TruStability® Board Mount Pressure Sensors, HSC Series—High Accuracy, Compensated/Amplified, ±1.6 mbar to ±10 bar | ±160 Pa to ±1 MPa | ±0.5 inH2O to ±150 psi, Digital or Analog Output*. Sensing and Control Honeywell, Golden Valley, Minnesota, USA, 8 2014, [Online; accessed 2022-03-27].
URL: https://prod-edam.honeywell.com/content/dam/honeywell-edam/sps/siot/en-us/products/sensors/pressure-sensors/board-mount-pressure-sensors/trustability-hsc-series/documents/sps-siot-trustability-hsc-series-high-accuracy-board-mount-pressure-sensors-50099148-a-en-ciid-151133.pdf

[28] Honeywell: *MPR SERIES, MicroPressure Board Mount Pressure Sensors Compact, High Accuracy, Compensated/Amplified*. Honeywell Advanced Sensing Technologies, Richardson, Texas, 5 2021, [Online; accessed 2022-03-13].
URL: `https://prod-edam.honeywell.com/content/dam/honeywell-edam/sps/siot/en-us/products/sensors/pressure-sensors/board-mount-pressure-sensors/micropressure-mpr-series/documents/sps-siot-mpr-series-datasheet-32332628-ciid-172626.pdf`

[29] International Organization for Standardization: ISO 11898-1:2015 Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling. [Online; accessed 2022-03-20].
URL: `https://www.iso.org/standard/63648.html`

[30] JLCPCB: JLCPCB. [Online; accessed 2022-03-20].
URL: `https://jlcpcb.com/`

[31] JSBSIM: Aerodynamic frame, defining the aerodynamic angles α_B and β. [Online; accessed 2021-10-19].
URL: `https://jsbsim-team.github.io/jsbsim-reference-manual/assets/img/ac_aero_axes.svg`

[32] Le Vie, L.: *Review of Research on Angle-of-Attack Indicator Effectiveness*. NASA Langley Research Center, Hampton, Virginia, 2014, [Online; accessed 2021-10-20].
URL: `https://www.researchgate.net/profile/Lisa-Le-Vie/publication/277331189_Review_of_Research_on_Angle-of-Attack_Indicator_Effectiveness/links/5568a89508aec22683032c03/Review-of-Research-on-Angle-of-Attack-Indicator-Effectiveness.pdf`

[33] Maxim Integrated: DS18B20, Programmable Resolution 1-Wire Digital Thermometer. [Online; accessed 2022-03-20].
URL: `https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf`

[34] MAXIN INTEGRATED: Guide to 1-Wire Communication, TUTORIALS 1796. [Online; accessed 2022-03-20].
URL: `https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1796.html`

[35] MICROCHIP: *High-Speed CAN Transceiver, DS20001667G*. Microchip Technology Inc., 2016, [Online; accessed 2022-03-27].
URL: `https://ww1.microchip.com/downloads/en/devicedoc/20001667g.pdf`

[36] Molex: 513820500, 2.00mm Pitch MicroClasp Wire-to-Board Receptacle Housing, Positive Lock, Single Row, 5 Circuits, White. [Online; accessed 2022-03-20].
URL: `https://www.molex.com/webdocs/datasheets/pdf/en-us/0513820500_CRIMP_HOUSINGS.pdf`

[37] Molex: 559350510, 2.00mm Pitch MicroClasp Wire-to-Board Header, Single Row, Right Angle, 5 Circuits, with PCB Locator. [Online; accessed 2022-03-20].
URL: `https://www.molex.com/webdocs/datasheets/pdf/en-us/0559350530_PCB_HEADERS.pdf`

[38] Mor, I.; Seabridge, A.; Jukes, M.: *Civil Avionics Systems, Second Edition*. John Wiley & Sons, Ltd, 2013, ISBN 9781118536735, 612 p.

[39] National, D.; Electronics, M.; Nmea, A.: National Marine Electronics Association. *History*, , n. June, 1983: p. 1–4.
URL: `https://www.nmea.org/`

[40] NATIONAL ADVISORY COMMITTEE FOR AERONAUTICS, Gracey, William: *TECHNICAL NOTE 4351, SUMMARY OF METHODS OF MEASURING ANGLE OF ATTACK ON AIRCRAFT*. NACA, Washington, 8 1958, [Online; accessed 2022-03-20].
URL: `http://www.windcraft.fi/aoa/doc/NACATN4351.pdf`

[41] NXP: I2C-bus specification and user manual, UM10204. [Online; accessed 2022-03-20].
URL: `https://www.nxp.com/docs/en/user-guide/UM10204.pdf`

[42] PERCEPIO AB: Percepio Tracealyzer. [Online; accessed 2022-03-20].
URL: `https://percepio.com/tracealyzer/`

[43] Popowski, S.; Dąbrowski, W.: Measurement and estimation of the angle of attack and the angle of sideslip. 03 2015, doi:10.3846/16487788.2015.1015293.

[44] ROHM semiconductor: 0.5A Variable Output Industrial LDO Regulator. [Online; accessed 2022-03-20].
URL: `https://fscdn.rohm.com/en/products/databook/datasheet/ic/power/linear_regulator/bdxxga5mefj-lb-e.pdf`

[45] Sankaralingam, L.; Ramprasadh, C.: A comprehensive survey on the methods of angle of attack measurement and estimation in UAVs. *Chinese Journal of Aeronautics*, volume 33, n. 3, 2020: p. 749–770, ISSN 1000-9361, doi:https://doi.org/10.1016/j.cja.2019.11.003, [Online; accessed 2021-10-20].
URL: `https://www.sciencedirect.com/science/article/pii/S1000936119304078`

[46] STMicroelectronics: ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32. [Online; accessed 2022-03-20].
URL: `https://www.st.com/en/development-tools/st-link-v2.html`

[47] STMicroelectronics: *Reference manual, STM32F446xx advanced Arm-based 32-bit MCUs, RM0390 Rev 6*. STMicroelectronics, 2021, [Online; accessed 2022-03-27].
URL: `https://www.st.com/resource/en/reference_manual/rm0390-stm32f446xx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf`

[48] STMicroelectronics: *STM32F446xC/E, Datasheet - production data, DS10693 Rev 10*. STMicroelectronics, 2021, [Online; accessed 2022-03-27].
URL: `https://www.st.com/resource/en/datasheet/stm32f446mc.pdf`

[49] STMicroelectronics: *User manual, Description of STM32F4 HAL and low-layer drivers, UM1725 Rev 7*. STMicroelectronics, 2021, [Online; accessed 2022-03-27].
URL: `https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics.pdf`

[50] STMicroelectronics: *User manual, STM32CubeIDE user guide, UM2609 Rev 5*. STMicroelectronics, 2021, [Online; accessed 2022-03-27].
URL: `https://www.st.com/resource/en/user_manual/dm00629856-stm32cubeide-user-guide-stmicroelectronics.pdf`

[51] Stock Flight Systems: *CAN Aerospace, Interface specification for airborne CAN applications V 1.7*. 12 2006, [Online; accessed 2022-03-13].
URL: `https://www.stockflightsystems.com/tl_files/downloads/canaerospace/canas_17.pdf`

[52] TE Connectivity: *MS4525DO SPECIFICATIONS*. Measurement Specialties, Inc., a TE Connectivity company., 2015, [Online; accessed 2022-03-27].
URL: `https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FMS4525DO%7FB2%7Fpdf%7FEnglish%7FENG_DS_MS4525DO_B2.pdf`

[53] TEXAS INSTRUMENTS: Introduction to the Controller Area Network (CAN). [Online; accessed 2022-03-20].
URL: `https://www.ti.com/lit/an/sloa101b/sloa101b.pdf`

[54] TEXAS Instruments: TPS736xx Cap-Free, NMOS, 400-mA Low-Dropout Regulator with Reverse Current Protection. [Online; accessed 2022-03-20].
URL: `https://www.ti.com/lit/ds/symlink/tps736.pdf`

[55] UBLOX: NEO-M8 u-blox M8 concurrent GNSS modules, UBX-15031086 - R11. [Online; accessed 2022-03-20].
URL: https://content.u-blox.com/sites/default/files/NEO-M8-FW3_DataSheet_UBX-15031086.pdf

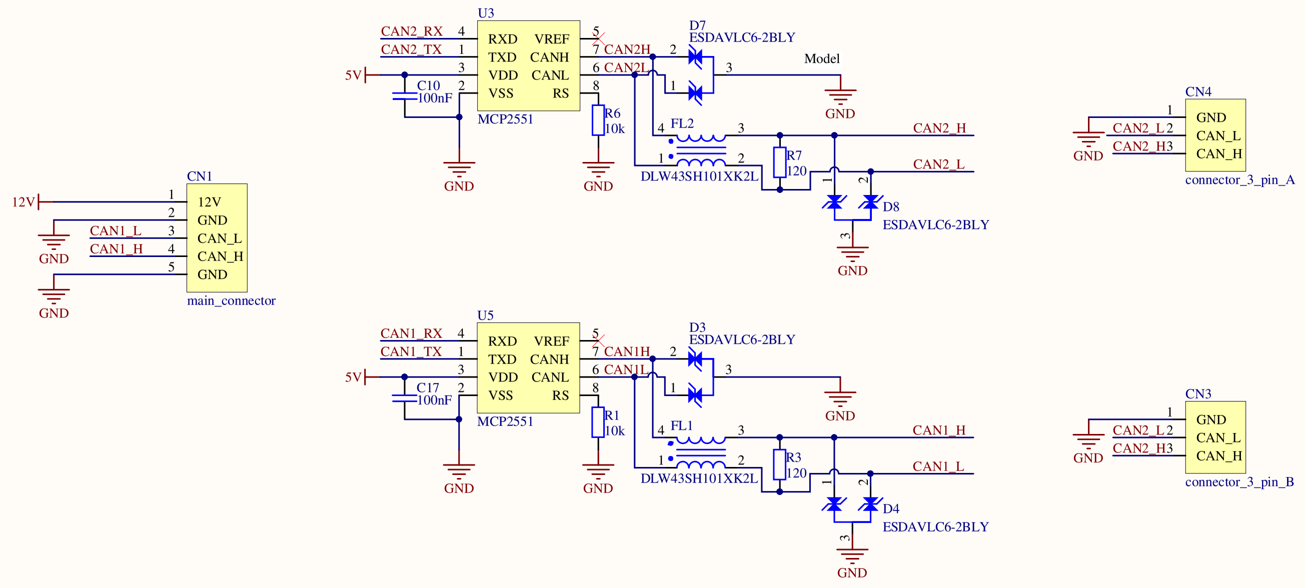
[56] UBLOX: u-blox 8 / u-blox M8 Receiver description, Including protocol specification, UBX-13003221 - R26. [Online; accessed 2022-03-20].
URL: https://content.u-blox.com/sites/default/files/products/documents/u-blox8-M8_ReceiverDescrProtSpec_UBX-13003221.pdf

[57] Woolf, P. and et al.: *Pressure Sensors*. University of Michigan, 2021, [Online; accessed 2021-10-20].
URL: https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Book%3A_Chemical_Process_Dynamics_and_Controls_(Woolf)/03%3A_Sensors_and_Actuators/3.03%3A_Pressure_Sensors

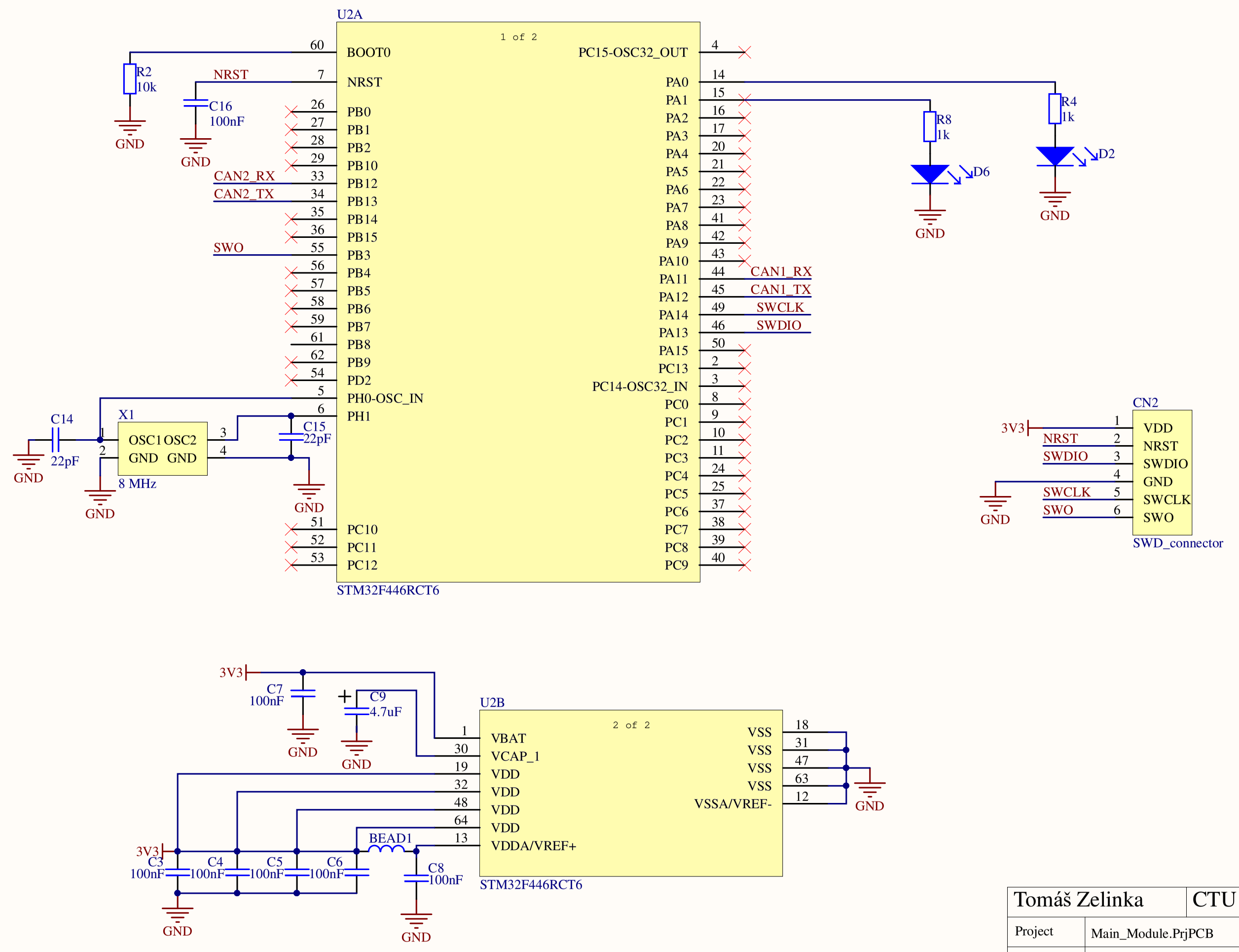
[58] Yeo, D.; Henderson, J.; Atkins, E.: An Aerodynamic Data System for Small Hovering Fixed-Wing UAS. 08 2009, doi:10.2514/6.2009-5756.

Appendix A
PCB Design

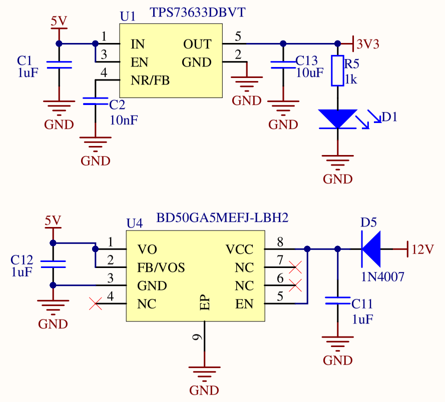
A.1 Board Schematic of Main Module



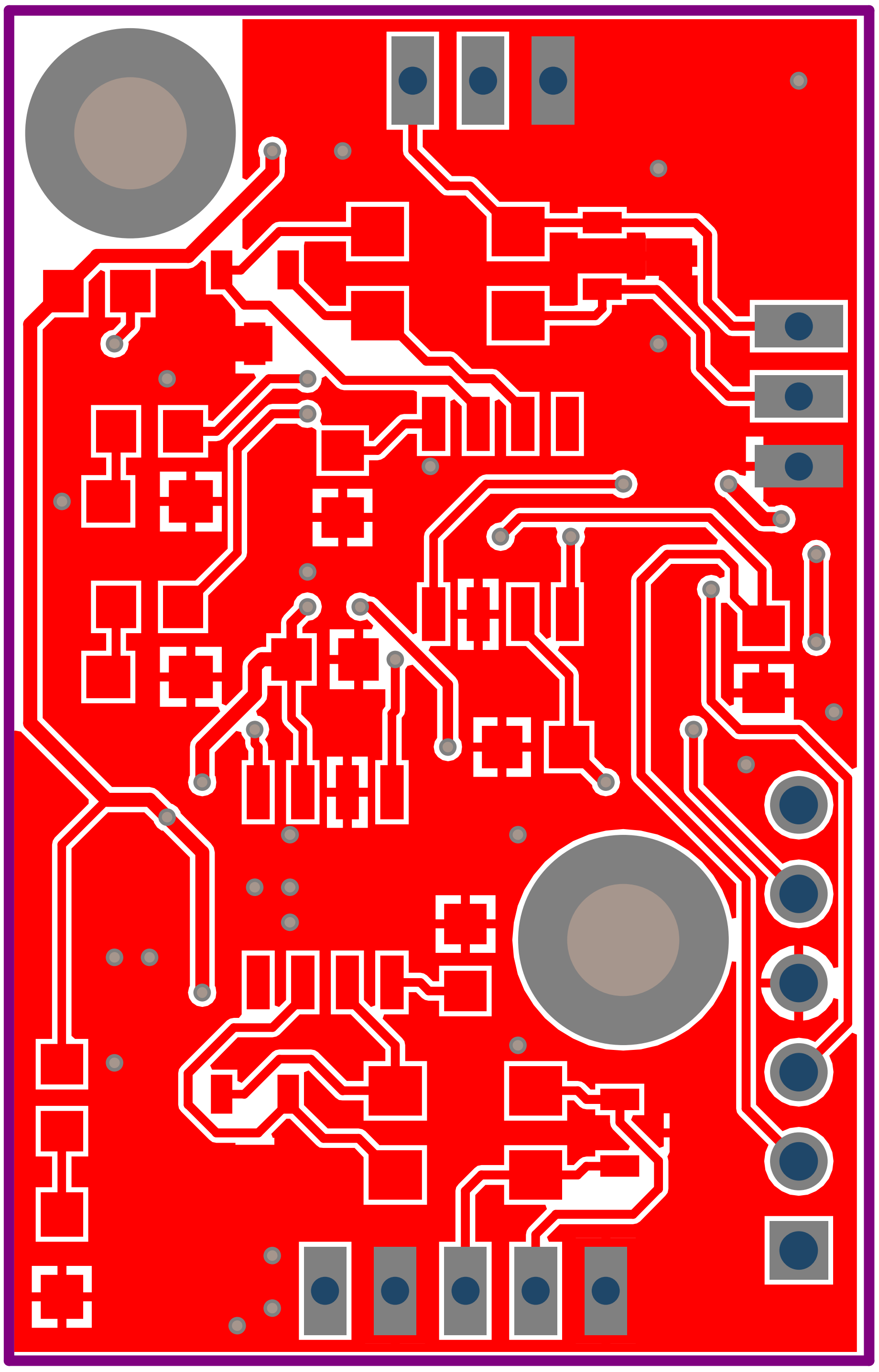
Tomáš Zelinka		CTU FEE
Project	Main_Module.PrjPCB	
Sheet name	can_bus.SchDoc	
Date	6.5.2022	Rev. A

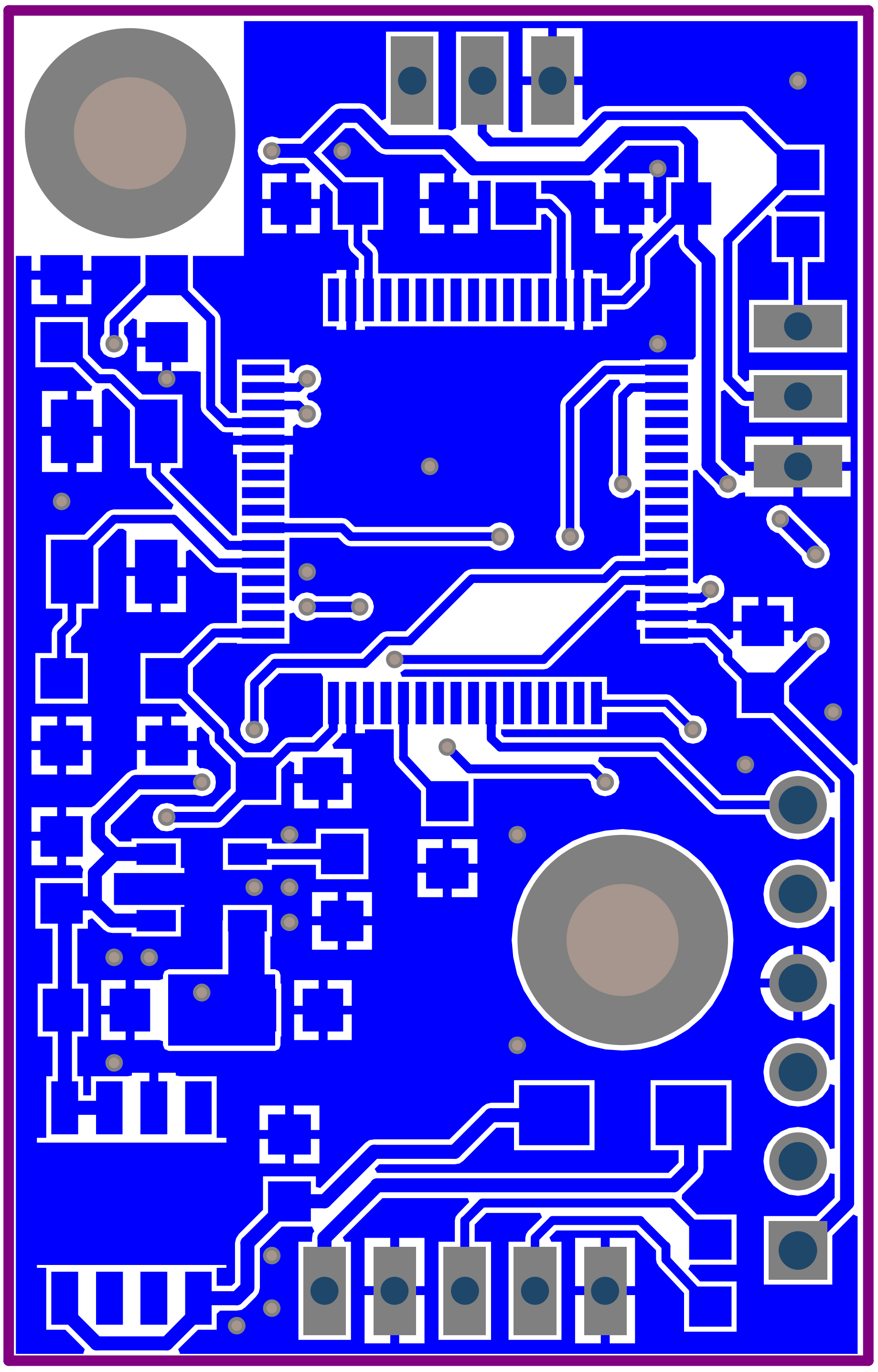


Tomáš Zelinka	CTU FEE
Project	Main_Module.PrjPCB
Sheet name	mcu.SchDoc
Date	6.5.2022	Rev. A

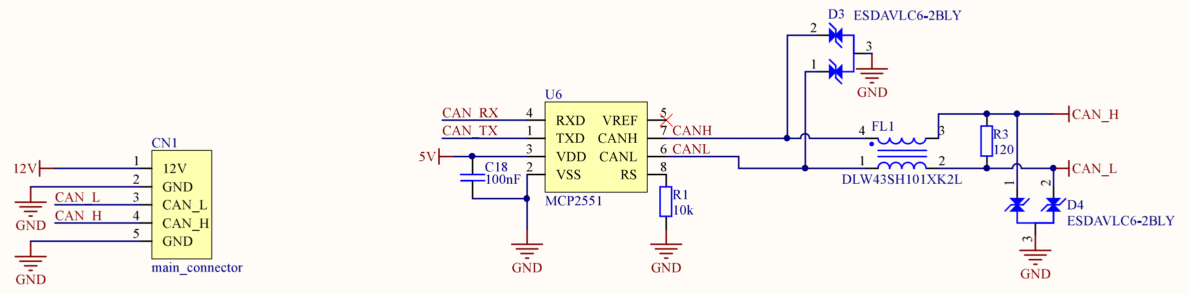


Tomáš Zelinka		CTU FEE
Project	Main_Module.PrjPCB	
Sheet name	power_supply.SchDoc	
Date	6.5.2022	Rev. A

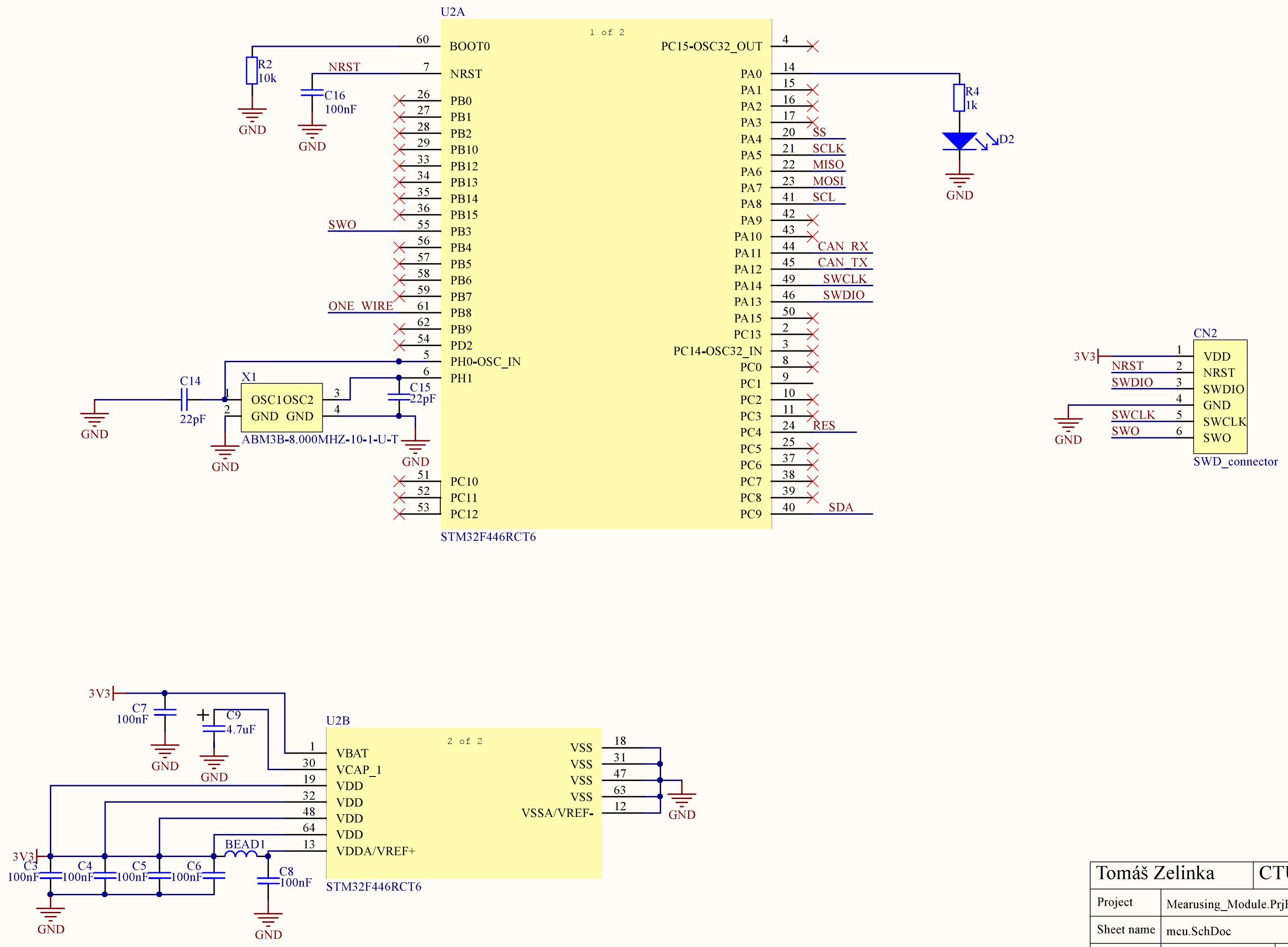




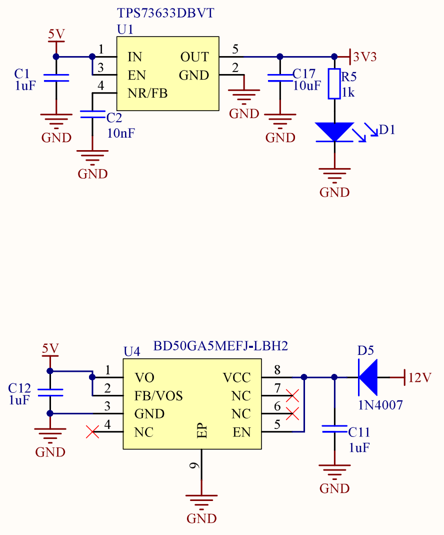
A.2 Board Schematics of Measuring Module



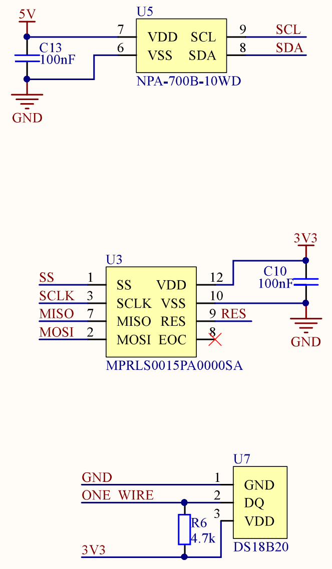
Tomáš Zelinka	CTU FEE	
Project	Mearusing_Module.PrjPCB	
Sheet name	can_bus.SchDoc	
Date	7.5.2022	Rev. B



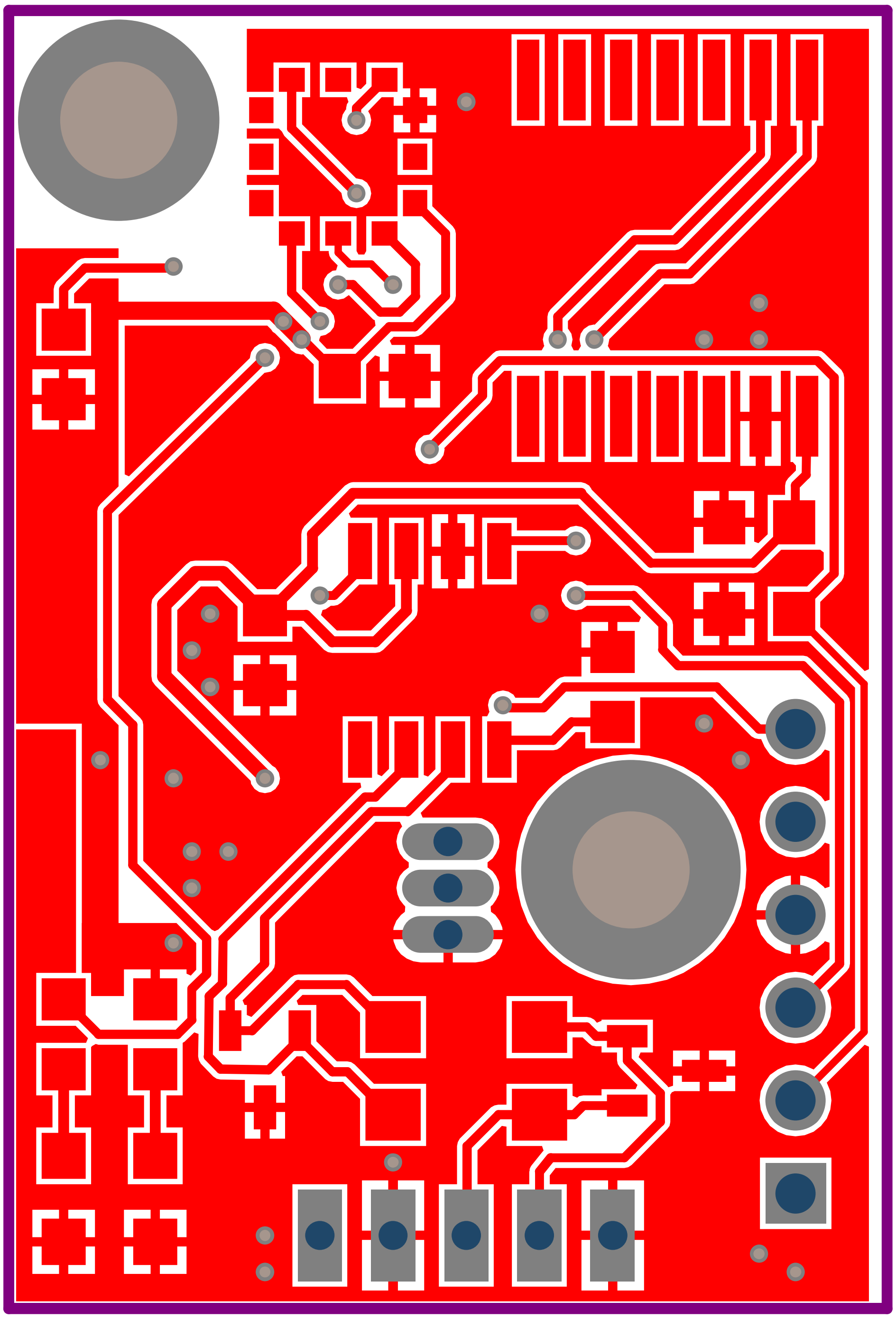
Tomáš Zelinka	CTU FEE	
Project	Mearusing_Module.PrjPCB	
Sheet name	mcu.SchDoc	
Date	7.5.2022	Rev. B

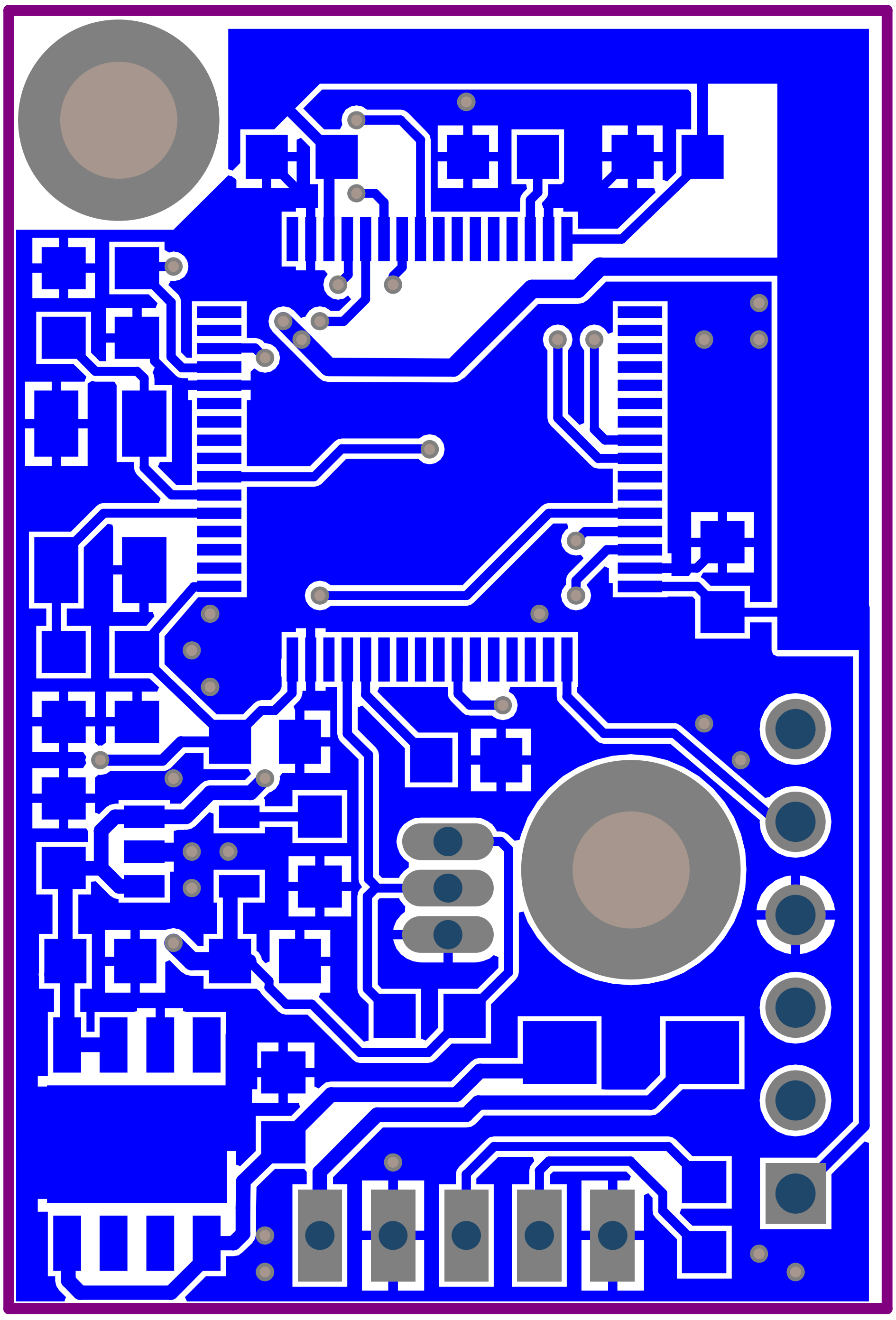


Tomáš Zelinka	CTU FEE	
Project	Mearusing_Module.PrjPCB	
Sheet name	power_supply.SchDoc	
Date	7.5.2022	Rev. B

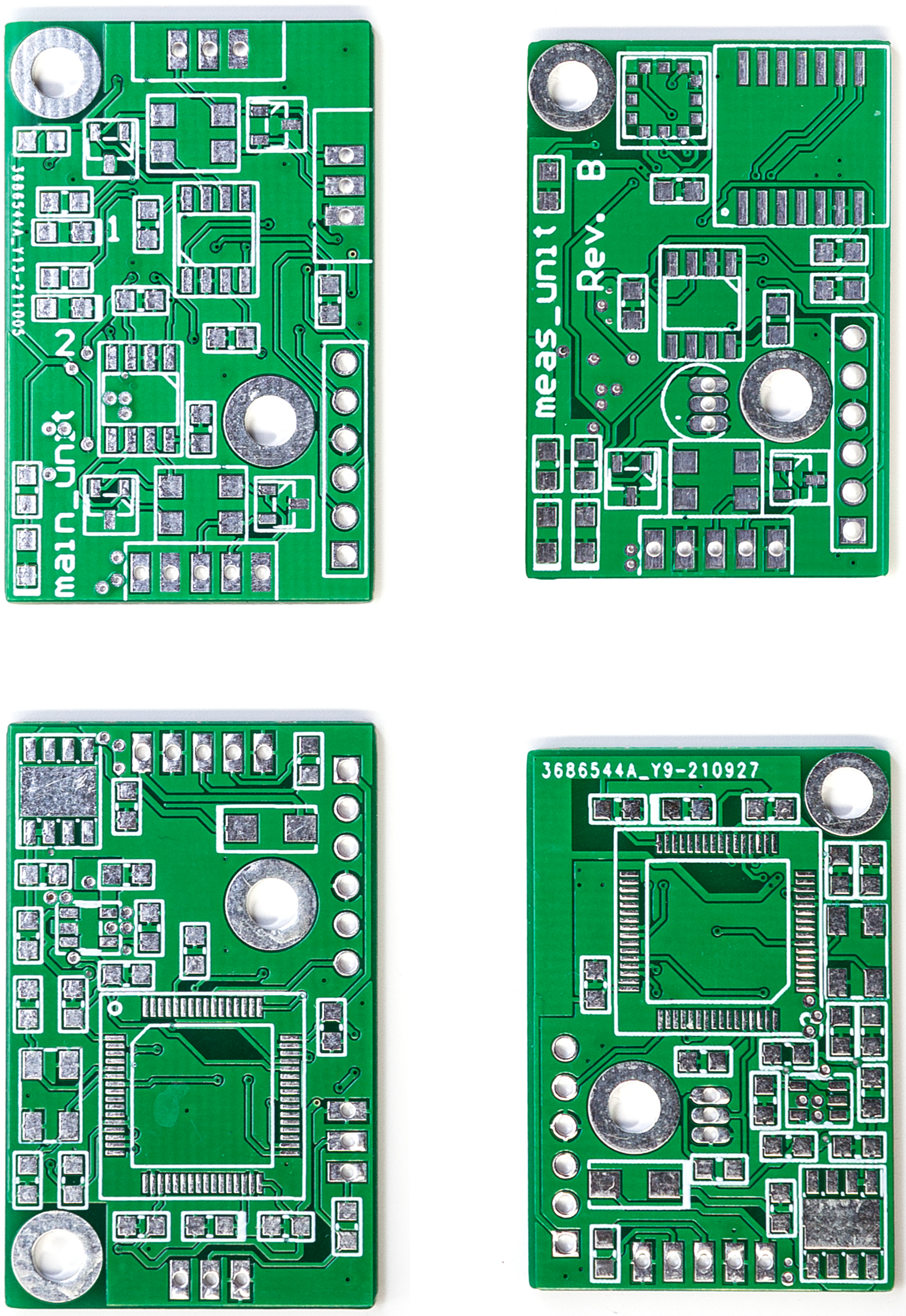


Tomáš Zelinka		CTU FEE
Project	Mearusing_Module.PrjPCB	
Sheet name	sensors.SchDoc	
Date	7.5.2022	Rev. B





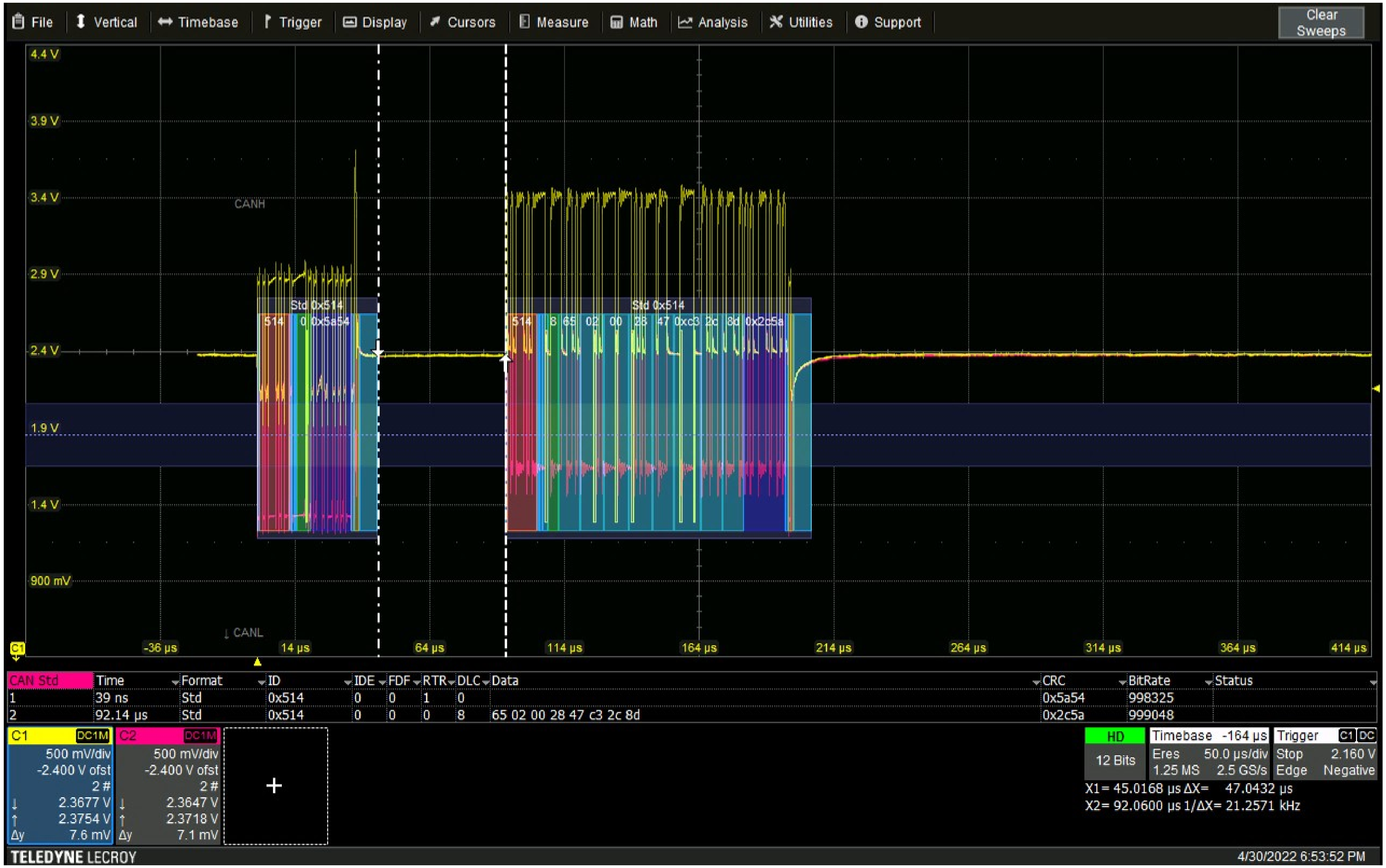
A.3 Manufactured Printed Circuit Boards


main_unit
1
2
3686544A_Y13-211005
meas_unit
Rev. B
3686544A_Y9-210927

Appendix B

CAN BUS Transmission

B.1 Decoded CAN Bus Transmission



CAN Std	Time	Format	ID	IDE	FDF	RTR	DLC	Data	CRC	BitRate	Status
1	39 ns	Std	0x514	0	0	1	0		0x5a54	998325	
2	92.14 µs	Std	0x514	0	0	0	8	65 02 00 28 47 c3 2c 8d	0x2c5a	999048	

C1 DC1M	C2 DC1M
500 mV/div	500 mV/div
-2.400 V ofst	-2.400 V ofst
2 #	2 #
↓ 2.3677 V	↓ 2.3647 V
↑ 2.3754 V	↑ 2.3718 V
Δy 7.6 mV	Δy 7.1 mV

```
HD        Timebase  -164 µs   Trigger   C1 DC
12 Bits   Eres   50.0 µs/div  Stop     2.160 V
          1.25 MS   2.5 GS/s  Edge    Negative
X1= 45.0168 µs ΔX=   47.0432 µs
X2= 92.0600 µs 1/ΔX= 21.2571 kHz
```

TELEDYNE LECROY

4/30/2022 6:53:52 PM

B.2 CAN Identifiers

Parameter	CAN identifier	Data format
Absolute pressure (A module)	1300	FLOAT
Absolute pressure (B module)	1301	FLOAT
Differential pressure (A module)	1302	FLOAT
Differential pressure (B module)	1303	FLOAT
Temperature (A module)	1304	FLOAT
Temperature (B module)	1305	FLOAT
Standard altitude (A module)	1306	FLOAT
Standard altitude (B module)	1307	FLOAT
True airspeed (A module)	1308	FLOAT
True airspeed (B module)	1309	FLOAT
Calibrated airspeed (A module)	1310	FLOAT
Calibrated airspeed (B module)	1311	FLOAT
Vertical speed (A module)	1312	FLOAT
Vertical speed (B module)	1313	FLOAT
UTC	1200	CHAR4
GPS aircraft height above ellipsoid	1038	FLOAT
GPS aircraft longitude	1036	DOUBLEL/DOUBLEH
GPS aircraft latitude	1037	DOUBLEL/DOUBLEH
Queue request	1499	n/a

Table B.1: Implemented CAN identifiers.