CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR'S THESIS

Jiří Fiedler

Synchronized Control of Group of Helicopters Using Direct Communication

Department of Cybernetics

Thesis supervisor: **Dr. Martin Saska**

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne............................ ....

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jiří F i e d l e r

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Synchronizované řízení skupiny helikoptér využívající přímou komunikaci

Pokyny pro vypracování:

Student v rámci své práce implementuje vhodný protokol umožňující komunikaci mezi řídící stanicí a jednotlivými helikoptérami pomocí XBee, což umožní měnit plán pohybu helikoptér za letu a zároveň analyzovat telemetrická data on-line.
Dále bude implementována přímá komunikace mezi helikoptérami, která umožní sdílení telemetrických dat a akčních zásahů získaných řídicí jednotkou na základě palubních senzorů. Student využije této sdílené informace k přesnější relativní stabilizaci letounů.
Chování navržené metody bude v reálných letových podmínkách porovnáno s chováním původního stabilizačního systému, který přímou komunikaci nevyužívá.
Robustnost metody bude ověřena sérií experimentálních letů formace 2-3 helikoptér.

Seznam odborné literatury:

[1] Lee, T. - Leoky, M. – McClamroch, N.H. Geometric tracking control of a quadrotor UAV on SE(3), IEEE Conference on Decision and Control (CDC), 2010.
[2] Krajník, T - Nitsche, M - Faigl, J. - Vaněk, P. - Saska, M. - Duckett, T. - Přeučil, L. - Mejail, M.: A practical multirobot localization system. Journal of Intelligent & Robotic Systems, Vol. 76, Issue 3-4, pp 539-562, 2014.
[3] Saska, M. - Vonásek, V. - Krajník, T. - Přeučil, L. Coordination and navigation of heterogeneous MAV–UGV formations localized by a 'hawk-eye'-like approach under a model predictive control scheme. International Journal of Robotics Research, Vol. 33, Issue 10, pp 1393-1412, July 2014.
[4] Daniel, K. - Wolff, A. - Wietfeld, C., Protocol Design and Delay Analysis for a MUAV-Based Aerial Sensor Swarm, IEEE Wireless Communications and Networking Conference (WCNC), 2010.
[5] Saska, M. - Kasl, Z. - Přeučil, L.: Motion Planning and Control of Formations of Micro Aerial Vehicles. The 19th World Congress of the International Federation of Automatic Control (IFAC), 2014.

Vedoucí bakalářské práce: Ing. Martin Saska, Dr. rer. nat.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 23. 1. 2015

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Jiří F i e d l e r

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Bachelor Project: Synchronized Control of Group of Helicopters Using Direct Communication

Guidelines:

A proper protocol will be implemented for communication between ground station and Micro Unmanned Vehicles (MAVs) with XBee modules. This enables to change plans of motion of MAVs during flight and to analyze telemetry from MAVs online.
Furthermore, a direct communication between MAVs will be implemented for sharing telemetric data and control inputs obtained based on onboard sensors by stabilization board. Student employs this kind of shared information for more precise relative stabilization of MAV groups.
The performance of the designed method will be compared with the original stabilization system, which does not use any direct communication.
The robustness of the method will be verified in various experiments of formation flying with 2-3 MAVs.

Bibliography/Sources:

[1] Lee, T. - Leoky, M. – McClamroch, N.H. Geometric tracking control of a quadrotor UAV on SE(3), IEEE Conference on Decision and Control (CDC), 2010.
[2] Krajník, T - Nitsche, M - Faigl, J. - Vaněk, P. - Saska, M. - Duckett, T. - Přeučil, L. - Mejail, M.: A practical multirobot localization system. Journal of Intelligent & Robotic Systems, Vol. 76, Issue 3-4, pp 539-562, 2014.
[3] Saska, M. - Vonásek, V. - Krajník, T. - Přeučil, L. Coordination and navigation of heterogeneous MAV–UGV formations localized by a 'hawk-eye'-like approach under a model predictive control scheme. International Journal of Robotics Research, Vol. 33, Issue 10, pp 1393-1412, July 2014.
[4] Daniel, K. - Wolff, A. - Wietfeld, C., Protocol Design and Delay Analysis for a MUAV-Based Aerial Sensor Swarm, IEEE Wireless Communications and Networking Conference (WCNC), 2010.
[5] Saska, M. - Kasl, Z. - Přeučil, L.: Motion Planning and Control of Formations of Micro Aerial Vehicles. The 19th World Congress of the International Federation of Automatic Control (IFAC), 2014.

Bachelor Project Supervisor: Ing. Martin Saska, Dr. rer. nat.

Valid until: the end of the summer semester of academic year 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 23, 2015

Acknowledgements

Firstly, I would like to thank my supervisor Dr. Martin Saska for his kindly approach. Furthermore I thank to Tomáš Báča for a lot of technical advice and assistance during experiments. At last, I want to thank my family and girlfriend for the encouragement during my studies.

Abstrakt

Tato práce se zabývá návrhem a implementací protokolu pro komunikaci mezi řídící stanicí a jednotlivými kvadrokoptérami pomocí XBee modulů. Za použití této komunikace je možné řídit kvadrokoptéry pomocí příkazů, měnit plán pohybu kvadrokoptér za letu a zároveň monitorovat telemetrická data. Dále je implementována příma komunikace mezi kvadrokoptérami, která je využita ke sdílení globálního souřadného systému. Počátky souřadných systémů kvadrokoptér ve formaci jsou sjednoceny a pozice kvadrokoptér ve formaci jsou měreny v jednom globálním souřadném systému. Globální souřadný systém je použit ke kompenzaci driftu pozice kvadrokoptér ve formaci. Každá kvadrokoptéra letí svou vlastní trajektorii v globálním souřadném systému nezávisle na ostatních kvadrokoptérách ve formaci, tudíž kvadrokoptéry mohou letět v různých tvarech formace. Tato metoda byla ověřena několika experimenty se skutečnými kvadrokoptérami.

Klíčová slova: XBee, Komunikace, Protokol, MAV, Telemetrie, Relativní stabilizace, Formace

Abstrakt

This thesis deals with design and implementation of protocol for communication between the ground station and micro aerial vehicles (MAVs) using XBee modules. Using this communication, MAV can be controlled by commands, trajectory of MAVs can be changed during flight and telemetry from MAVs can be analyzed online. Furthermore, a direct communication between MAVs is implemented and used for global coordinate system distribution. Origins of the coordinate systems of MAVs in the formation are unified and positions of MAVs in the formation are measured in one global coordinate system. Global coordinate system is used to compensate relative position drift of MAVs in the formation. Each MAV flies its own trajectory in the global coordinate system independently from other MAVs in the formation, therefore various shapes of the formation can be flown. This method is verified by several experiments with real MAVs.

Keywords: XBee, Communication, Protocol, MAV, Telemetry, Relative stabilization, Formation

Contents

List of Figures

1 Introduction

Micro aerial vehicles (MAVs) capable of vertical takeoff are very popular nowadays. Multicopters became low-cost easily accessible platform in past few years. They can be seen in various shapes and sizes. They cost from fifteen to thousands of euro. Smaller and cheaper multicopters are widely used among the hobbyist. Someone use them just as a toy for fun, others do first person view racing competitions with small, agile and really fast quadcopters. Toy quadcopter and basic hobby quadcopter are shown in figure 1. Amazing aerial videos and photos can be made with MAVs too. Aerial photography is usually made by professionals with expensive, large and reliable hexacopters or octocopters. Multicopters are also used in practical tasks. Amazon PrimeAir is one of the example [3]. Amazon use multicopters to deliver packages to the customers. Multicopters are even able to save lives. Ambulance drone is project about hexacopter carrying mobile defibrillator [5]. This MAV is able to carry 4 kg of payload and fly up to 100 km/h. MAVs are used for aerial observation and exploration, especially in places which were hit by natural disasters, which could be dangerous for a human and which are too small for an aircraft with human crew [2].

Figure 1: *An example of multicopters*

Multicopters are great platform for science and research due to its low cost and potential applications. Multicopters require certain level of automation for their purposes. The lowest automatic assistance is on toys and racing multicopters. This assistance enables pilot to control the multicopter and stabilize it in desired angle. Advanced level of automatic assistance is used on multicopters designed for aerial photography. These multicopters use barometer, magnetometer and the global position system (GPS) to stabilize their position and altitude. They support features like return to home, when MAV autonomously flies to the take-off point and lands there. MAVs must be fully autonomous in applications like package delivery or aerial observation [3, 2]. For these purposes controllers must be developed [4, 13].

Single MAV is able to fly autonomously nowadays, Amazon PrimeAir and Ambulance drone are the examples. Next challenge is to create autonomous MAV formation. MAV formation can be used in applications, where carrying heavy load or monitoring large area is needed. MAVs can fly in heterogeneous formation with ground vehicles in order to monitor the environment from high perspective and navigate the formation [26].

Navigation and localization of MAVs are crucial for successful flight in the formation. External camera system such as Vicon in special laboratory room can be used for precise localization in indoor applications [1, 14]. Laboratory conditions are great for developing controllers, but external camera system is not available in scenarios like exploration and rescue. External camera system also limits the range of flight to environment, which is equipped with this expensive system. MAVs using onboard sensors for navigation are not limited in range of flight. GPS can be used for outdoor applications and formation with large spacing [28]. GPS lacks the required precision (up to 1 m) for compact formations of small MAVs, moreover GPS loses reliability in urban and indoor environments. Optical flow sensors can be used in both outdoor and indoor applications [6, 9]. Optical flow sensor is low-cost, lightweight, onboard sensor, which works similar as a typical optical mouse. Optical flow sensor is used in this work.

Swarm robotics is one of the research topics of Multi-robot System group (MRS) at Czech Technical University in Prague. The goal of MRS is to develop distributed autonomous system, which can control MAV groups with height precision. The MAV groups should be independent on the environment and capable of indoor and outdoor flight. It should be fully self-contained without using any external localization system, external cameras or external computation power. MAV should rely purely on the on-board sensors and on-board computation power [10]. MAVs used by MRS are equipped with XBee 2.4 Ghz modules used for wireless communication.

Communication between MAV and ground station (laptop) enables ground station operator to monitor MAV telemetry and progress of the mission. Online mission progress monitoring is one of the key features in applications like aerial observation. Telemetry monitoring helps to avoid mission failure and MAV crashes. Ground station operator is able to identify fault and stop the MAV before accident. It is possible to control the MAV by sending its commands. The whole MAV formation can be controlled at once. When an unexpected situation occurs, the ground station operator is able to land with the whole formation to prevent the damage on MAVs and surrounding environment. MAVs are able to communicate between themselves in formation. MAVs can share telemetry and adjust their trajectory in order to adapt to a new situation [28].

The goal of this thesis is to design and implement a protocol for communication between MAVs and ground station via XBee modules. Using this communication, the ground station operator is able to monitor MAV telemetry online, moreover MAVs can be controlled by commands from the operator. Using these commands, it is possible to dynamically change trajectory of MAVs in order to adapt the shape of the formation to the environment, land at once with the whole MAV formation in case of danger, synchronize time on MAVs with the ground station and change positions of MAVs. The position of the MAV can be set in a new coordinate system. The position change is used for navigation of MAVs by external devices like other MAVs, ground vehicles or dock station. Communication between MAVs is implemented and used to improve relative stabilization of MAVs in formation. All MAVs in formation share one global coordinate system. All positions and trajectory waypoints (section 5) are measured in this system. Global coordinate system is periodically distributed in formation. Periodical distribution compensates position drift caused by noisy optical flow sensor measurement. Each MAV is independent unit and can fly its own trajectory in formation, therefore various shapes of formation can be flown. Designed protocol and it usage is verified by experiments with real MAVs.

All goals of this thesis are fulfilled. Overview of the experimental platform used by MRS is described in section 2. The XBee modules used for communication are described in section 3. The MAV coordinate system and position change are described in section 4. Position change is crucial for navigation of MAVs in the formation. Trajectory waypoints and setpoints calculation are described in section 5. The setpoint calculation is remade in order to support dynamic changes of trajectory waypoints and time measurement. The new approach to flight in formation is described in section 6. This section describes global coordinate system and its distribution. The MAV communication capability is described in section 7. The designed protocol used for communication is described in section 8. This section serves as a reference book for new protocol.

1.1 State of the art

MAVLink Micro Air Vehicle Communication Protocol is protocol widely used among the MAVs. MAVLink is lightweight, header-only message marshalling library for MAVs. It is used for communication between the ground station and MAVs and for inter-communication between the subsystem of the MAV. It can pack C-structure over serial channels with high efficiency. It is used in PX4, Pixhawk, APM and Parrot AR Drone commercial platforms [19].

The protocol is designed to be transferred via serial link, hence MAVLink packets have header consisting of length of the packet, component ID, message ID, checksum and more. The structure of MAVLink packets is shown in figure 2. The minimum packet length is

8 bytes for acknowledgement packets. The maximum packet length is 263 bytes for full payload. MAVLink supports integer and float data types and arrays of these data types. The payload of the packet are MAVLink messages. Each MAVLink message is identified by its ID. ID defines type of the MAVLink message. MAVLink supports up to 256 types, where IDs 150-240 are reserved for extensions (personal messages). Each type has its own defined structure.

Byte Index	Content	Value	Explanation
0	Packet start sign	v1.0: 0xFE (v0.9: 0x55)	Indicates the start of a new packet.
1	Payload length	0 - 255	Indicates length of the following payload.
2	Packet sequence	0 - 255	Each component counts up his send sequence. Allows to detect packet loss
3	System ID	1 - 255	ID of the SENDING system. Allows to differentiate different MAVs on the same network.
4	Component ID	0 - 255	ID of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot.
5	Message ID	0 - 255	ID of the message - the id defines what the payload “means” and how it should be correctly decoded.
6 to (n+6)	Data	(0 - 255) bytes	Data of the message, depends on the message id.
(n+7) to (n+8)	Checksum (low byte, high byte)	ITU X.25/SAE AS-4 hash, **excluding packet start sign, so bytes 1..(n+6)** Note: The checksum also includes MAVLINK_CRC_EXTRA (Number computed from message fields. Protects the packet from decoding a different version of the same packet but with different variables).	

Figure 2: *Structure of MAVLink packet*
Source: http://qgroundcontrol.org/mavlink/start

There are several reasons why we decided not to use MAVLink Micro Air Vehicle Communication Protocol and develop a new protocol. The main reasons are to set the size of transferred packets to minimum and reach the maximum efficiency of XBee modules. Communication with MAVs and ground station is always done via XBee modules using

API frames 3.2 hence the protocol does not need to support features like checksum or packet length, because these are implemented in XBee API frames. MAVLink trajectory waypoint format is designed for GPS navigation and the operation of reading and writing waypoints is complicated. Waypoints are read and written one by one and transfer is confirmed by acknowledge packets. That requires a lot of XBee communication capacity. In the new protocol, all trajectory waypoints are sent in one packet, which increases the payload/header ratio and is frugal to XBee communication capacity. MAVLink protocol has wide range of use. It supports helicopters, multicopters, fixed wings, rockets, submarines and others, hence it offers features which are not needed for our work. The new protocol is designed just for our needs and therefore is more efficient.

2 Experimental platform

2.1 Hardware overview

MAVs used in this thesis are based on L4-ME quadcopter kit designed by MikroKopter company. This platform includes brushless 110 W motors equipped with 10" propellers, speed controllers BL-Ctrl V2.0 and stabilization board FlightCtrl V2.5 ME [15]. The MAV is shown in figure 3. We have created a custom landing gear, propeller guards and LED lights.

Figure 3: *Experimental MAV* [15]

Each MAV is equipped with camera module. The camera module is Caspa™ camera and Overo® computer made by Gumstix. The camera module is used for relative visual localization of MAVs [7, 8]. The camera module is shown in figure 4a. The camera module software is able to detect a circular pattern (further referred as Blob) shown in figure 4b and calculate its 3D relative position to the MAV from its position in the camera frames and knowledge of its absolute dimensions. The camera module is able to calculate the position at the rate from 33 Hz to 60 Hz. This system is further referred as Blob detector [12].

Each MAV is equipped with Pixhawk PX4Flow Smart Camera module extended with Maxbotix HRLV-EZ4 ultrasonic range finder [17]. The PX4Flow is shown in figure 5. Ultrasonic range finder is used for measuring distance from the ground. The distance is measured from 0.3 m to 4 m. The smart camera module is pointing to the ground. It calculates velocity along the two horizontal axes relative to the ground surface from the two consecutive images from the camera. The module includes 3-axis gyroscope that is used for compensating rotation motion of the sensor. Data is send over UART (universal asynchronous receiver-transmitter) using MAVLink protocol.

(a) *Caspa camera with Overo computer* [7, 8]

(b) *Blob*

Figure 4: *Blob detector*

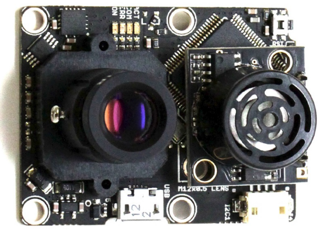


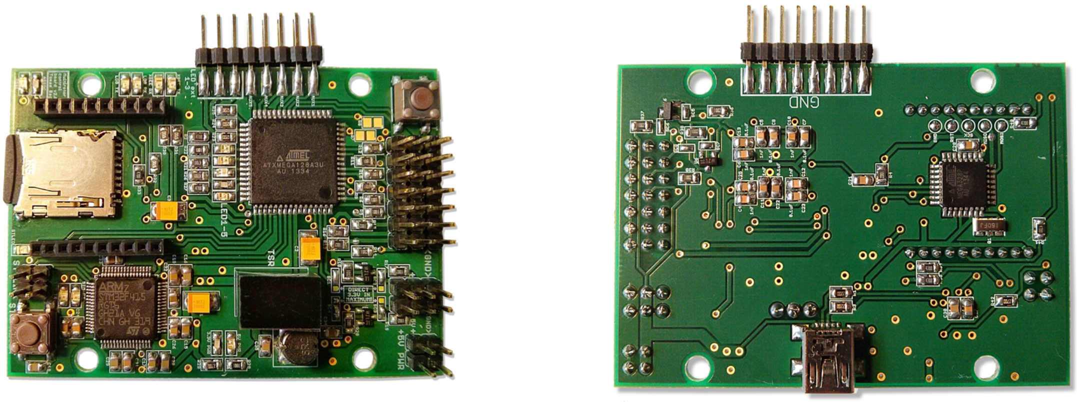
Figure 5: *PX4Flow Smart Camera module* [17]

The brain of the MAVs is a custom control board version 2. The custom control board is shown in figure 6. The custom control board contains two microcontrollers, OpenLog module, XBee socket and micro SD card socket. The board supports connection of external sensors and modules via UART and i^{2}c. The board has square mounting pattern (45 x 45 mm) [4].

The main microcontroller is 8-bit AVR, ATxMega128a3u. This MCU (microcontroller unit) is used to handle communication, compute trajectory and compute altitude, speed and position controllers. Trajectory is described in section 5. Controllers are described in section 2.2. The main microcontroller has 32 MHz clock, 8 kB of SRAM memory, 7 separate UARTs, where 3 of them are used for communication with other onboard modules and 4 UARTs are used for communication with external modules, two i^{2}c lines and PPM (pulse position modulation) input/output for communication with RC receiver and FlightCtrl V2.5 ME.

The second MCU is 32-bit ARM device, STM32F415RGT6. This MCU is used for computing MPC controller and Kalman filter described in thesis [4]. It has 168 MHz clock and 192 kB of RAM. It serves as a coprocessor for xMega. It communicates with xMega via UART.

OpenLog is an open source solution for data logging. It is an SD card logging device connected by UART to the xMega. It logs as many telemetry data as possible and serves as a black box [27]. Using OpenLog, logs with telemetry data are available for post-processing and flight evaluation.



(a) Custom control board top

(b) Custom control board bottom

Figure 6: *Custom control board v.2* [4]

XBee modules are used for communication between MAVs and ground station. XBee is ZigBee module made by Digi [11]. XBee Pro S2B is shown in figure 7. XBee modules are described in section 3.

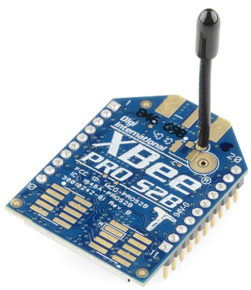


Figure 7: *XBee PRO S2B* [11]

2.2 Software overview

Altitude, velocity, position and MPC controllers are available for MAV control. Altitude, velocity and position controller are computed on xMega MCU. These controllers are closely described in thesis [6]. MPC controller is computed on the ARM MCU. This controller is closely described in thesis [4]. Altitude controller is a PID controller using relative distance from the ground surface from Maxbotix HRLV-EZ4 ultrasonic range finder. This controller is computed simultaneously with the rest of the controllers. Speed controller is a PID controller using relative speed to the ground from Pixhawk PX4Flow. Speed controller is used in autonomous landing task. Position controller is a PID controller using position of the MAV. Position is calculated from the MAV speed. This controller has been replaced by MPC controller on experimental platform, but it is still available. Model predictive controller (MPC) is calculated on ARM MCU on custom control board. MPC controller is robust and reliable controller which is used for trajectory following. Trajectory is described in section 5.

MAVs are able to land autonomously. Trajectory following is stopped and the MAV smoothly lands if landing is turned on. The MAV takes off to the desired altitude and continue with trajectory following if landing is turned off. Trajectory is described in section 5. Speed controller and altitude controller are used for autonomous landing. Autonomous landing is done in several steps. These steps are also called landing states. Landing states are described in table 1. Landing state is switched to stabilization if landing is turned on. Landing state is switched to take off if landing is turned off.

Landing state	**Description**
Stabilization	Speed setpoints are set to zero. Altitude setpoint is set to 35cm. Landing state is switched to *landing* if altitude deflection is lower than 10cm for 0.5 second.
Landing	In *landing* phase, the MAV is 35cm above the ground surface. Throttle output is fluently decreasing and the MAV smoothly lands on the ground surface. Landing state is switched to *on ground* if throttle output reaches altitude controller saturation.
On ground	The MAV is on the ground surface and throttle output is equal to the lower altitude controller saturation.
Take off	Speed setpoints are set to zero. Altitude setpoint is set to desired altitude calculated from trajectory waypoints. Landing state is switch to *flight* if altitude deflection is lower than 10cm for 0.5 second.
Flight	The MAV is in the air and is controlled by active controller.

Table 1: Landing states

Time is measured on MAVs. Time measurement starts from zero when the MAV is turned on because of absence of real time clock module. Time is measured with precision of milliseconds using crystal oscillations and microcontroller interruption. Time can be changed and synchronized with the ground station using MAV communication protocol (MCP). MCP is described in section 8.

3 XBee

XBee is ZigBee module made by Digi. XBee Pro S2B is used in this project. XBee modules are used for communication between MAVs and the ground station. XBee Pro S2B is low-cost, low-power wireless mesh network module, that operates within the 2.4 GHz frequency band designed to operate within the ZigBee protocol. XBee PRO S2B is shown in figure 8. XBee Pro S2B communicates with xMega MCU via UART. XBee Pro S2B has transmission distance up to 90 meters in indoor or urban environment and data throughput is up to 35kbps. XBees are typically used in a low data rate applications like wireless control and monitoring. They are not suited for real-time critical data transfer. Two modes are available for communication with XBee. AT mode is described in section 3.1 and API (application programming interface) mode is described in section 3.2 [11].

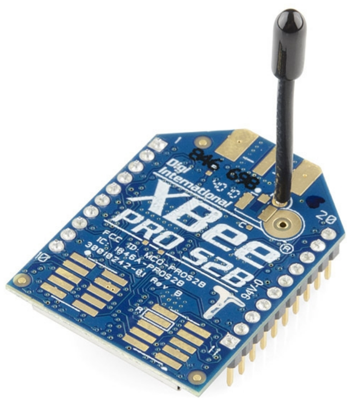


Figure 8: *XBee PRO S2B* [11]

ZigBee is based on an IEEE 802.15.4 standard. ZigBee devices can transmit data over long distances using a mesh network. Each ZigBee module has an unique MAC address. This address is used for identification. ZigBee mesh network is shown in figure 9. Three device types are used in ZigBee mesh network.

ZigBee coordinator is the most capable device in ZigBee mesh network. The coordinator forms the network. Exactly one coordinator is in each network. Coordinator is able to route data from other devices, communicate with all devices in network, allow routers and end devices to join the network and buffer RF data packets for sleeping end device children.

ZigBee router must join a ZigBee mesh network before it can transmit, receive and route data. Router is able to route data from other devices, communicate with all devices in network, allow routers and end devices to join the network and buffer RF data packets for sleeping end device children.

ZigBee end device must join a ZigBee mesh network before it can transmit and receive data. End device is able to communicate just with the parent node (either the coordinator or a router). End device can not route data from other devices and allow devices to join the ZigBee mesh network. Relationship between end device and parent node enables end device to be asleep which increases battery life.

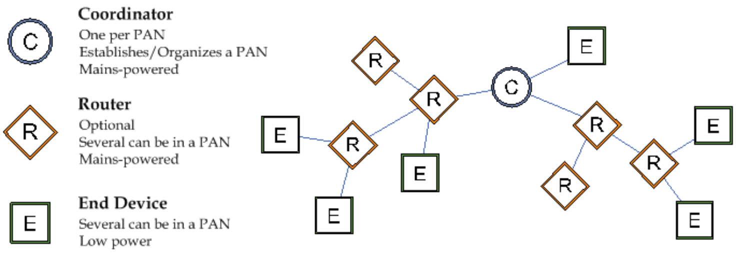


Figure 9: *ZigBee mesh network*
Source: [11]

3.1 AT mode

AT mode is synonym to Transparent mode. Any data sent to the XBee is sent to the remote ZigBee module identified by the destination MAC address in memory. AT mode is useful for very simple networks, where is not necessary to change destination MAC address very often. Command sequence must be sent to XBee module in order to switch it to command mode. Module operating in AT mode can be configured in command mode using AT commands. The biggest disadvantage is that receiving module is not able to distinguish the source of the data.

3.2 API mode

API operation requires that communication must be done through a structure interface (API Frames). Overview of API frames is shown in figure 10. Sending data to different ZigBee modules using API mode is simple, because destination MAC address is part of the API frame. Module operating in API mode is able to distinguish the source of the received data. API mode is used in this project.

ZigBee transmit request API frame is send to the XBee module via UART in order to send data to another ZigBee device. Structure of the *ZigBee transmit request API frame* is shown in figure 11. Frame type is identifier of the *ZigBee transmit request API frame*. Frame ID is used to identify ACK (acknowledgement) packets. ACK packets are not used in this thesis. 64-bit destination address is MAC address of the XBee to which the data should be delivered. 16-bit destination address is address of ZigBee module in current network. If 16-bit address is set as unknown, XBee module is able to find that address and store it in its memory. XBee module is able to retry sending packet in case of failure. This feature

API Frame Names	API ID
AT Command	0x08
AT Command - Queue Parameter Value	0x09
ZigBee Transmit Request	0x10
Explicit Addressing ZigBee Command Frame	0x11
Remote Command Request	0x17
Create Source Route	0x21
AT Command Response	0x88
Modem Status	0x8A
ZigBee Transmit Status	0x8B
ZigBee Receive Packet (AO=0)	0x90
ZigBee Explicit Rx Indicator (AO=1)	0x91
ZigBee IO Data Sample Rx Indicator	0x92
XBee Sensor Read Indicator (AO=0)	0x94
Node Identification Indicator (AO=0)	0x95
Remote Command Response	0x97
Over-the-Air Firmware Update Status	0xA0
Route Record Indicator	0xA1
Many-to-One Route Request Indicator	0xA3

Figure 10: *API frames*
Source: [11]

can be disabled by setting the option field to value 0x01. Retries should be disabled in real-time applications like online telemetry monitoring. RF data are payload which is sent to target ZigBee device. MAV communication protocol (MCP) is transferred as a payload of ZigBee transmit request API frame. MCP is described in section 8.

ZigBee receive packet API frame is received from the XBee module, when XBee module receives packet. *ZigBee receive packet API frame* is shown in figure 12. Frame type is identifier of the *ZigBee receive packet API frame*. 64-bit and 16-bit addresses are addresses of the source ZigBee module. Received data contain the MCP packet.

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x16	
	Frame-specific Data	Frame Type	3	0x10	
		Frame ID	4	0x01	Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to 0, no response is sent.
		64-bit Destination Address	MSB 5	0x00	Set to the 64-bit address of the destination device. The following addresses are also supported: 0x0000000000000000 - Reserved 64-bit address for the coordinator 0x000000000000FFFF - Broadcast address
			6	0x13	
			7	0xA2	
			8	0x00	
			9	0x40	
			10	0x0A	
			11	0x01	
			LSB 12	0x27	
		16-bit Destination Network Address	MSB 13	0xFF	Set to the 16-bit address of the destination device, if known. Set to 0xFFFE if the address is unknown, or if sending a broadcast.
			LSB 14	0xFE	
		Broadcast Radius	15	0x00	Sets maximum number of hops a broadcast transmission can occur. If set to 0, the broadcast radius will be set to the maximum hops value.
		Options	16	0x00	Bitfield of supported transmission options. Supported values include the following: 0x01 - Disable retries and route repair 0x20 - Enable APS encryption (if EE=1) 0x40 - Use the extended transmission timeout Enabling APS encryption presumes the source and destination have been authenticated. I also decreases the maximum number of RF payload bytes by 4 (below the value reported by NP). The extended transmission timeout is needed when addressing sleeping end devices.It also increases the retry interval between retries to compensate for end device polling.See Chapter 4, Transmission Timeouts, Extended Timeout for a description. Unused bits must be set to 0.
		RF Data	17	0x54	Data that is sent to the destination device
			18	0x78	
			19	0x44	
			20	0x61	
			21	0x74	
			22	0x61	
			23	0x30	
			24	0x41	
	Checksum		25	0x13	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Figure 11: *ZigBee transmit request API frame*
Source: [11]

	Frame Fields		Offset	Example	Description
API Packet	Start Delimiter		0	0x7E	
	Length		MSB 1	0x00	Number of bytes between the length and the checksum
			LSB 2	0x11	
	Frame-specific Data	Frame Type	3	0x90	
		64-bit Source Address	MSB 4	0x00	
			5	0x13	64-bit address of sender. Set to 0xFFFFFFFFFFFFFFFF (unknown 64-bit address) if the sender's 64-bit address is unknown.
			6	0xA2	
			7	0x00	
			8	0x40	
			9	0x52	
			10	0x2B	
			LSB 11	0xAA	
		16-bit Source Network Address	MSB 12	0x7D	16-bit address of sender
			LSB 13	0x84	
		Receive Options	14	0x01	0x01 - Packet Acknowledged 0x02 - Packet was a broadcast packet 0x20 - Packet encrypted with APS encryption 0x40 - Packet was sent from an end device (if known) Note: Option values can be combined. For example, a 0x40 and a 0x01 will show as a 0x41. Other possible values 0x21, 0x22, 0x41, 0x42, 0x60, 0x61, 0x62.
		Received Data	15	0x52	Received RF data
			16	0x78	
			17	0x44	
			18	0x61	
			19	0x74	
			20	0x61	
	Checksum		21	0x0D	0xFF - the 8 bit sum of bytes from offset 3 to this byte.

Figure 12: *ZigBee receive packet API frame*
Source: [11]

4 Coordinate system

It is important to define the coordinate system in which position of the MAV is measured. The quadcopter is set in "X" configuration as shown in figure 13. The z axis is perpendicular to axes x and y and is oriented downwards. Roll, pitch and yaw are angles of rotation of the body around the axes x, y, z. Distances are measured in meters and angles are measured in degrees in Cartesian coordinate system. Elevator position and aileron position terms are used in this thesis for measuring positions in the coordinate system of the MAV. Elevator position is position along the x axis. Aileron position is position along the y axis.

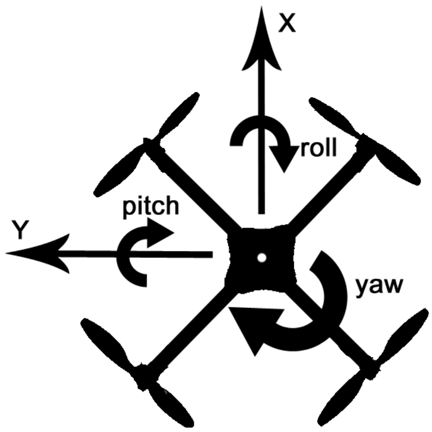


Figure 13:
Coordinate system of the MAV

Each MAV has its own coordinate system. Let's call this system the MAV coordinate system. Origin of the MAV coordinate system is set to the current position of the MAV if the MAV is turned on and if MPC controller set as active. Position of the MAV in the MAV coordinate system is calculated by Kalman filter from speed measurement. Position of the MAV in a new coordinate system can be set using MAV communication protocol (MCP).

4.1 Coordinate system change

Position of the MAV in a new coordinate system can be set using *MCP Position set command packet* (section 8.1.7). New elevator position and aileron position must be set in meters. The new coordinate system in which the new position of the MAV is measured must be Cartesian and axes of the new coordinate system must be parallel with axes of the MAV coordinate system.

If the position of the MAV in the new coordinate system differs from the position of the MAV in the MAV coordinate system, the coordinate systems have different origins. MAV states including its position are estimated by Kalman filter. These states are unchangeable. Because origin of the MAV coordinate system can not be changed, transformation between the MAV coordinate system and the new coordinate system is done. Let's consider position as a vector

$$\mathbf{P} = \begin{pmatrix} elevator\, position \\ aileron\, position \end{pmatrix}. \tag{1}$$

Position of the origin of the MAV coordinate system in the new coordinate system is calculated by equation

$$\mathbf{P_{org}} = \mathbf{P_{new}} - \mathbf{P_{MAV}}, \tag{2}$$

where $\mathbf{P_{new}}$ is position of the MAV in the new coordinate system and $\mathbf{P_{MAV}}$ is position of the MAV in the MAV coordinate system. Schematic picture is shown in figure 14. $\mathbf{C_{new}}$ is origin of the new coordinate system and $\mathbf{C_{MAV}}$ is origin of the MAV coordinate system.

It is possible to transform points from the MAV coordinate system to the new coordinate system and back using the position of the origin of the MAV coordinate system in the new coordinate system. Position transformation is described by equations

$$\mathbf{P}'_{\mathbf{MAV}} = \mathbf{P}'_{\mathbf{new}} - \mathbf{P_{org}}, \tag{3}$$

$$\mathbf{P}'_{\mathbf{new}} = \mathbf{P}'_{\mathbf{MAV}} + \mathbf{P_{org}}, \tag{4}$$

where $\mathbf{P}'_{\mathbf{MAV}}$ is point in the MAV coordinate system, $\mathbf{P}'_{\mathbf{new}}$ is point in the new coordinate system and $\mathbf{P_{org}}$ is the position of the origin of the MAV coordinate system in the new coordinate system.

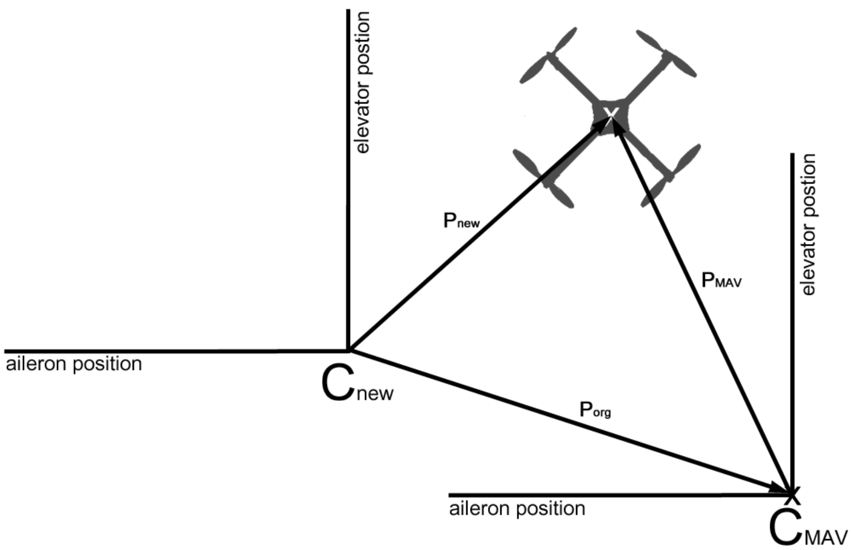


Figure 14: *Position of the MAV in the MAV coordinate system and in the new coordinate system*

Change of the MAV position is important for navigation. Elevator position and aileron position of trajectory waypoints are set in the new coordinate system. Setpoints calculated from the trajectory waypoints are then transformed to the MAV coordinate system using equation (3). Trajectory waypoints are described in section 5. Position of the MAV monitored by the ground station operator and used for global coordinate system distribution is transformed to the new coordinate system using equation (4). Telemetry monitoring is described in section 7.1. Global coordinate system distribution is described in section 6.4. Coordinate system change is verified in experiment in section 9.2.

5 Trajectory

Movement of the MAV is defined by trajectory waypoints. Position setpoints for position PID controller and MPC controller are calculated from the trajectory waypoints. Trajectory waypoint is defined by time, altitude, elevator position and aileron position. Waypoint time is desired time, when the MAV should be on elevator and aileron position in a desired altitude. Time is set in a POSIX time format. Up to ten waypoints can be stored in the MAV memory. These waypoints are designed to be changed dynamically using MAV communication protocol (MCP), hence more waypoints are not needed. Waypoints must be sorted by time for proper functionality of setpoints calculation. MCP is described in section 8.

5.1 Setpoints calculation

Setpoints are calculated from trajectory waypoints. Trajectory is calculated with frequency of 70 Hz. Elevator and aileron position setpoints and altitude setpoint are calculated from current setpoints, current time and the selected waypoint. The selected waypoint is the first in MAV memory. If the current time is equal or grater than the selected waypoint's time, next waypoint in memory is selected. If there are no more waypoints to fly trough, setpoints are set to last waypoint positions and altitude. The calculation is described by equations

$$\Delta t = t - t_w, \tag{5}$$

$$E_s(t + t_s) = E_s(t) + \frac{E_w - E_s(t)}{\Delta t} t_s, \tag{6}$$

$$A_s(t + t_s) = A_s(t) + \frac{A_w - A_s(t)}{\Delta t} t_s, \tag{7}$$

$$At_s(t + t_s) = At_s(t) + \frac{At_w - At_s(t)}{\Delta t} t_s, \tag{8}$$

where t is the current time, t_w is the waypoint time, E_s is elevator position setpoint, A_s is aileron position setpoint, At_s is altitude setpoints, E_w is waypoint elevator position, A_w is waypoint aileron position, At_w is waypoint altitude, t_s is the sampling time of trajectory calculation.

6 Flight in Formation

The main goal of this project is to design and implement a system for MAVs flying together in desired formation. Shape of the formation can be changed dynamically. MAVs rely purely on the on-board sensors and computation power.

6.1 Previous work

Previous work on control of MAVs in formation is described in thesis [6]. No communication between the MAVs neither between the MAV and the ground station was employed. The whole trajectory was programmed in the custom control board. The leader-follower strategy was used in the formation. The leader MAV was following trajectory using PX4Flow sensor data. The follower MAV was following the leader MAV using data from Blob detector. The follower MAV had a constant position setpoint. The position of the follower MAV was determined by its relative position to the Blob attached to the leader MAV. The Blob was the origin of the follower MAV coordinate system. This approach has specific limitations. Just one shape of the formation is available. The follower MAV follows the leader MAV with a certain lag. Oscillations of the leader MAV has negative impact on performance of the follower MAV.

6.2 New approach

In the new approach presented in this thesis, each MAV flies its own trajectory independently of the other MAVs, hence various shapes of the formation can be flown. Each MAV has its own coordinate system, which complicates navigation of MAVs in the formation and trajectory waypoints calculation. Global coordinate system is introduced in order to simplify navigation of MAVs in the formation. Origins of MAVs coordinate systems are changed and united. Positions of MAVs can be measured absolutely in the global coordinate system. Blob detector and XBee are used to distribute global coordinate system between MAVs in the formation.

This approach enables possibility of flying in various shapes of the formation, because each MAV is an independent unit, which flies its own trajectory in the global coordinate system independently of the other MAVs. Moreover imperfect trajectory following of other MAVs in the formation can not impair performance of the MAV, because trajectory of the MAV in the global coordinate system is not affected by other MAVs. The new approach is verified by experiment in section 9.4.

6.3 Global coordinate system

Each MAV has its own coordinate system. One of the coordinate systems has to be chosen as the global coordinate system in order to unify the MAVs coordinate systems. Positions of the MAVs are measured in the global coordinate system and trajectory way-points are set in the global coordinate system. Change of the coordinate system of the MAV is described in section 4.1.

Origins of MAVs coordinate systems drift relatively to the ground, because position of the MAV is not measured directly, but is calculated from speed, which is noisy. The position drift is measured in experiment in section 9.1. If this drift would not be compensated, after a while MAVs would not be in a place in the formation in which they supposed to be and can crash with other MAVs. The MAV formation drifts as one unit if global coordinate system exists and MAVs do not crash with each other. Drift of the formation can be removed if the formation is heterogeneous and other vehicle with drift-less position measurement participate, for example vehicle using SLAM for its localization.

MAVs are able to distribute the global coordinate system and compensate position drift using MAV communication protocol (MCP) and Blob detector. MCP is described in section 8.

6.4 Global coordinate system distribution

Two MAVs participate on the global coordinate system distribution. The coordinate system of the first MAV (master MAV) is considered as the global coordinate system, the coordinate system of the second MAV (slave MAV) is changed. The master MAV must know the address of the slave MAV. The address is set using MCP *Position slave set command packet* (section 8.1.5). The master MAV has known position in the global coordinate system. It is possible to measure relative distance between the master MAV and the slave MAV using the Blob detector. Let's consider position as a vector

$$\mathbf{P} = \begin{pmatrix} elevator\, position \\ aileron\, position \end{pmatrix}. \tag{9}$$

Position of the slave MAV in the global coordinate system is calculated by equation

$$\mathbf{P_{slave}} = \mathbf{P_{master}} + \mathbf{P_{relative}}, \tag{10}$$

where $\mathbf{P_{slave}}$ is position of the slave MAV in the global coordinate system, $\mathbf{P_{master}}$ is position of the master MAV in the global coordinate system and $\mathbf{P_{relative}}$ is relative distance between the master MAV and the slave MAV.

Calculated position of the slave MAV in the global coordinate system is sent to the slave MAV using MCP *Position set command packet* (section 8.1.7). Position of the slave MAV is then set in the global coordinate system as described in section 4.1.

The position of slave MAV in global coordinate system is set with error caused by XBee packet transfer delay. This error can be calculated by equation

$$\Delta p = t_d * v, \tag{11}$$

where Δp is position error, t_d is XBee packet transfer delay and v is speed of the slave MAV. According to section 3.3 in paper [18], XBee packet transfer delay is lower than 0.075 second. Speed of the MAV is limited to 0.4 m/s. According to the equation (11), error is lower than 0.03 m for both x and y axes.

Global coordinate system can be distributed through the whole formation using position master-slave distribution method. MAVs just need to be linked in pairs and have blob in sight. An example of MAVs configuration for global coordinate system distribution in the formation is shown in figure 15.

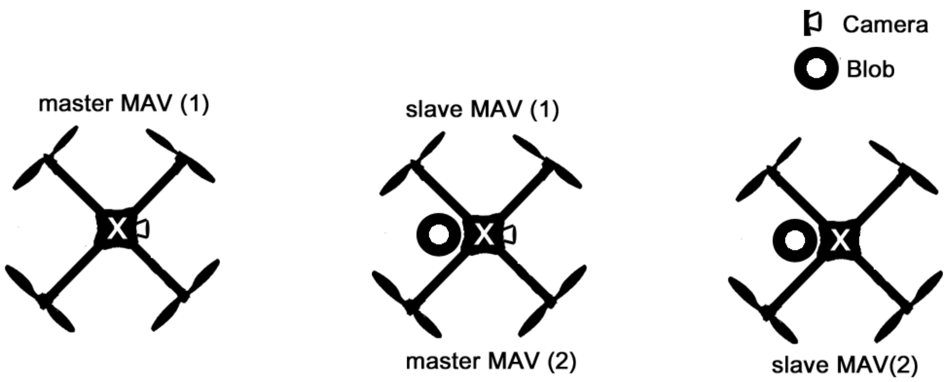


Figure 15: *An example of coordinate system distribution in the formation*

The global coordinate system must be distributed periodically in order to compensate position drift. Periodically distributed global coordinate system also corrects position error caused by MAVs different yaw angle in the formation. The global coordinate system distribution is made every two seconds. This period has been found sufficient for position drift correction and is frugal to XBee channel transfer capacity. Performance of two MAVs with and without periodically distributed coordinate system is tested in experiment in section 9.3.

6.5 Preparation of the flight in the formation

Several steps must be followed to perform flight of MAVs in the formation. For safety reasons, each MAV is controlled by human pilot who can take control over the MAV in case of danger. The steps for performing the flight of MAVs in the formation may be summarized as follows:

1. Place MAVs on the ground in the desired starting shape of the formation. All MAVs in the formation must be oriented the same way. All MAVs in the formation must be at positions in which Blobs are visible due to proper function of main coordinate system distribution.

2. Turn the MAVs on

3. Turn on position PID controller or MPC controller. Turn on landing

4. Turn main coordinate system distribution on. Upload starting position of the trajectories to the MAVs

5. Check if positions and setpoints of the MAVs are correct. Pilots of the MAVs set throttle on RC controller to middle position

6. Turn off landing. All MAVs hover in the air on a starting place of their trajectory

7. Upload trajectories to the MAVs. MAVs start to fly automatically

8. Change and upload new trajectories dynamically when needed

7 MAV Communication

MAV is able to communicate with the ground station and other MAVs. Communication is done via XBee Pro S2B using MAV communication protocol (MCP). Each XBee has specific MAC address. Each MAV is identified by MAC address of XBee, which is mounted on it. MAVs are equipped with router XBee. The ground station is equipped with coordinator XBee. This distribution of XBees must be abide in order to proper functionality of telemetry monitoring. The ground station operator is able to monitor MAV states and telemetry. The MAV can be controlled by MPC commands. XBee is described in section 3. MCP is described in section 8.

Autonomous machines have to be under human control especially in phase of development. In case of failure, the ground station operator is able to stop the whole formation at once. On-line telemetry monitoring allows to prevent accidents and can save health and money.

Communication between MAVs is used to increase precision of relative stabilization of the MAVs in the formation. XBee is not suited for critical real-time data transfer, because of its transfer delay and low transfer capacity. Therefore, real-time sharing of control outputs from controllers is not appropriate. Moreover the MAV is unstable system with disturbances. Controllers compensate these disturbances and stabilize the MAV. Disturbances can not be predicted and differ on every MAV. Hence, it is not possible to obtain useful information from control outputs, which can be used for improving relative stabilization of the MAVs. Communication between MAVs is used to share positions of the MAVs in the global coordinate system distribution. Global coordinate system distribution is described in section 6.4.

7.1 Telemetry monitoring

Each MAV is able to send telemetry to the ground station. Telemetry is send periodically with frequency of 20 Hz. Specific telemetry data sent by the MAV can be chosen by ground station operator using *MCP Telemetry to coordinator command packet* (section 8.1.1). Telemetry is received as a *MCP telemetry packet* (section 8.2). Telemetry monitoring is used to observe progress of the MAV mission and check if the mission continues as expected. Failure of sensor or a bug in code can be revealed by the ground station operator and accidents can be prevented. Telemetry data which can be monitored are shown in table 2.

Telemetry	**Description**
Altitude	Ground distance obtained from *PX4Flow*
Estimated altitude	Filtered altitude
Altitude speed	Derivation of estimated altitude
Elevator speed	Speed obtained from *PX4Flow*
Aileron speed	Speed obtained from *PX4Flow*
Estimated elevator speed	Elevator speed filtered by Kalman filter
Estimated aileron speed	Aileron speed filtered by Kalman filter
Elevator position	Position estimated by Kalman filter
Aileron position	Position estimated by Kalman filter
Elevator acceleration	Acceleration estimated by Kalman filter
Aileron acceleration	Acceleration estimated by Kalman filter
Elevator acceleration error	Acceleration error estimated by Kalman filter
Aileron acceleration error	Acceleration error estimated by Kalman filter
Elevator acceleration input	Acceleration input estimated by Kalman filter
Aileron acceleration input	Acceleration input estimated by Kalman filter
Altitude controller output	Output value of altitude controller
Elevator controller output	Output value of currently active controller
Aileron controller output	Output value of currently active controller
Altitude setpoint	Desired altitude
Elevator position setpoint	Desired elevator position
Aileron position setpoint	Desired aileron position
Valid Blob	If data from Blob detector are valid
Blob elevator	Blob elevator distance
Blob aileron	Blob aileron distance
Blob altitude	Blob altitude distance
Pitch angle	Pitch angle of the MAV
Roll angle	Roll angle of the MAV
Output throttle	Value of throttle output from control board
Output elevator	Value of elevator output from control board
Output aileron	Value of aileron output from control board
Output rudder	Value of rudder output from control board
Elevator shift	Deflection between MAV and new coordinate system
Aileron shift	Deflection between MAV and new coordinate system

Table 2: MAV telemetry

7.2 MAV control

MAVs can be controlled by MCP commands. The ground station operator is able to sent those commands and manage the MAV. MPC commands are described in section 8.1. The main advantage is possibility to control several MAVs at once. This ability is helpful if MAVs fly in the formation and flight plan is changed and if the MAVs in the formation need to land. MAV control is shown in table 3.

MAV control	Description
Telemetry to Coordinator	Specific telemetry data which are sent to the ground station for monitoring can be chosen by *MPC telemetry to coordinator command* (section 8.1.1). Telemetry monitoring is described in section 7.1
Landing	Autonomous landing can turned on and off by *MCP landing request command* (section 8.1.2). Autonomous landing is described in section 2.2.
Controllers	Active controller can be switched by *MCP controllers request command* (section 8.1.3). Controllers are described in section 2.2.
Trajectory set	Movement of the MAV is controlled by trajectory waypoints. Trajectory waypoints can be set by *MCP trajectory set command* (section 8.1.4). Trajectory is described in section 5.
Position set	Position of the MAV can be set in a new coordinate system using *MPC position set request command* (section 8.1.7). Position of MAV must be set in meters. Position change is described in section 4.1.
Position slave set	The MAV is able to distribute its coordinate system to another MAV. Address of the MAV, whose coordinate system should be changed, can be set by *MPC position slave set request command* (section 8.1.5). Coordinate system distribution is described in section 6.4.
Time set	Ground station operator is able to set time to the MAV. Time must be synchronized with ground station every time when the MAV is turned on because of absence of real time clock module on MAV. Time on the MAV can be set by *MCP time request command* (section 8.1.6). Time measurement on the MAV is described in section 2.2.

Table 3: MAV control

7.3 MAV states

The ground station operator is able to monitor MAV states. Landing state and controllers state are automatically sent to the ground station when changed. All states can be obtained by *MCP status request command* (section 8.1). States are received as *MCP reports* (section 8.3). MAV states are described in table4.

MAV States	Description
Telemetry to Coordinator state	Telemetry to Coordinator state informs whether chosen telemetry type is send to ground station. Telemetry to Coordinator state is obtained from *MCP telemetry to coordinator report* (section 8.3.1). Telemetry monitoring is described in section 7.1.
Landing state	Ground station operator is able to monitor current landing state. Landing state is obtained from *MCP landing report* (section 8.3.2). Landing states are states described in section 2.2.
Controllers state	Controllers state informs which controller is active. Controllers state is obtained from *MCP controllers report* (section 8.3.3). Controllers are described in section 2.2.
Trajectory waypoints state	Trajectory waypoints state informs about trajectory waypoints in the MAV memory. It describes theirs desired altitude, elevator, aileron and time. Trajectory waypoints are obtained from *MCP trajectory set report* (section 8.3.4). Trajectory is described in section 5.
Position slave state	Ground station operator is able to monitor current slave MAV address using position slave state. Position slave state is obtained from *MCP position slave set report* (section 8.3.5). Position slave is described in section 6.4.
Time state	Time state informs about MAV time. Time is obtained from *MCP time report* (section8.3.6). MAV time measurement is described in section 2.2.

Table 4: MAV States

7.4 Messages

It is possible to send text messages between devices equipped with XBee. Text messages can be used for debugging. Text messages are simple string messages up to 256 characters long. Sending messages is described in section 8.4.

8 MAV communication protocol (MCP)

The new protocol designed for MAV communication called MCP is implemented. MCP provides packets for MAV control and monitoring. MCP is used for communication between MAVs and between the MAV and the ground station. The ground station operator is able to monitor MAV telemetry. The MAV can be controlled by commands. MCP is designed for communication via XBee modules (section 3). API frames must be used for communication with XBee. MCP packets are transferred as a payload of the *ZigBee Transmit Request API frame* and *ZigBee Receive Packet API frame*. MCP does not need to support packet checksum, because checksum is calculated in XBee API frames. Length of MCP command is calculated from length of XBee API frames.

Several packet types are available. Packet types have tree-like architecture. Packets are divided into packet categories by the value of the first byte (category identifier). Packet categories are shown in table 5.

Packet category	Identifier	Description
Command	0x63 (c)	Choose telemetry, which should be monitored
Telemetry	0x74 (t)	Enable and disable landing
Report	0x72 (r)	Turn on and off controllers
Message	0x6D (m)	Add way-points to trajectory

Table 5: Packet categories

8.1 Command Packets

Command packets are designed for MAV control. *Command packets* start with the command category identifier (0x63 - c). Command type identifier is the second byte of the packet. Command types are shown in table 6. *Report packets* can be obtained by *status request commands*. Each command type have its own *status request command*. Each *status request command* has Get Status identifier (0xFF) on the third byte of the packet.

Command type	Identifier	Description
Telemetry to coordinator	0x01	Choose telemetry, which should be monitored
Landing	0x02	Enable and disable landing
Controllers	0x03	Turn on and off controllers
Trajectory set	0x04	Add way-points to trajectory
Position slave set	0x05	Set the slave MAV in coordinate system distribution
Time	0x06	Set time
Position set	0x07	Set new position of the MAV

Table 6: Command types

8.1.1 Telemetry to Coordinator

Telemetry to coordinator command is used to set which telemetry types are send to the ground station for online monitoring. Each telemetry type has specific identifier. MCP supports up to 256 different telemetry types. Identifiers are shown in table 33. *Telemetry to coordinator command packet* consists of four bytes. The structure of the *telemetry to coordinator command packet* is shown in table 7.

Packet Fields	Byte	Value	Description
Packet category	1	0x63	Command category
Command type	2	0x01	Telemetry to coordinator type
On/Off	3	0x01/0x00	Turn sending on or off
Telemetry identifier	4	0x00-0xFF	Identifier of telemetry data

Table 7: Telemetry to coordinator command packet

Telemetry to coordinator status request command is used to obtain *telemetry to coordinator report packet* (section 8.3.1). The *Telemetry to coordinator status request command packet* consists of four bytes. The structure of the *telemetry to coordinator status request command packet* is shown in table 8.

Packet Fields	Byte	Value	Description
Packet category	1	0x63	Command category
Command type	2	0x01	Telemetry to coordinator type
Get Status identifier	3	0xFF	
Telemetry identifier	4	0x00-0xFF	Identifier of telemetry data

Table 8: Telemetry to coordinator status request command packet

8.1.2 Landing

Landing request command is used to turn on and off autonomous landing. Autonomous landing is described in section 2.2. *Landing request command packet* consists of three bytes. The structure of the *landing request command packet* is shown in table 9.

Packet Fields	Byte	Value	Description
Packet category	1	0x63	Command category
Command type	2	0x02	Landing type
On/Off	3	0x01/0x00	Turn landing on or off

Table 9: Landing request command packet

Landing status request command is used to obtain *landing report packet* (section 8.3.2). *Landing status request command packet* consists of three bytes. The structure of the *landing status request command packet* is shown in table 10.

Packet Fields	Byte	Value	Description
Packet category	1	0x63	Command category
Command type	2	0x02	Landing type
Get Status identifier	3	0xFF	

Table 10: Landing status request command packet

8.1.3 Controllers

Controllers request command is used to switch between active controller. Controllers are described in section 2.2. Each of controller has its own identifier. Controller identifiers are shown in table 32. MCP supports up to 255 controllers. *Controllers request command packet* consists of three bytes. The structure of the *controllers request command packet* is shown in table 11.

Packet Fields	Byte	Value	Description
Packet category	1	0x63	Command category
Command type	2	0x03	Controllers type
Controller identifier	3	0x00-0xFE	Identifier of desired controller

Table 11: Controllers request command packet

Controllers status request command is used to obtain *controllers report packet* (section 8.3.3). *Controllers status request command packet* consists of three bytes. The structure of the *controllers status request command packet* is shown in table 12.

Packet Fields	Byte	Value	Description
Packet category	1	0x63	Command category
Command type	2	0x03	Controllers type
Get Status identifier	3	0xFF	

Table 12: Controllers status request command packet

8.1.4 Trajectory set

Trajectory set request command is used to set trajectory waypoints. Trajectory waypoints are described in section 5. To set more than one waypoint, repeat time, elevator position, aileron position and altitude in packet multiple times. More waypoints are transferred in one packet in order to increase payload/header ratio of the XBee API frames. MCP supports up to 255 waypoints in one packet. *Trajectory set request command packet* consists of $3 + 16k$ bytes, where k is number of trajectory waypoints. The structure of the *trajectory set request command packet* is shown in table 13.

Packet Fields	Byte	Value	Description
Packet category	1	0x63	Command category
Command type	2	0x04	Telemetry set type
Size	3	0x00-0xFE	Number of trajectory waypoints in packet
Time	4-7	uint32	Unsigned 4-byte integer in binary form
Elevator position	8-11	float	4-byte float in binary form
Aileron positon	12-15	float	4-byte float in binary form
Altitude	16-19	float	4-byte float in binary form
Time	20-23	uint32	Unsigned 4-byte integer in binary form
Elevator position	24-27	float	4-byte float in binary form
Aileron positon	28-31	float	4-byte float in binary form
Altitude	32-35	float	4-byte float in binary form
...	...	...	...

Table 13: Trajectory set request command packet

Trajectory set status request command is used to obtain *trajectory set report packet* (section 8.3.4). *Trajectory set status request command packet* consists of three bytes. The structure of the *trajectory set status request command packet* is shown in table (sec. 14).

Packet Fields	Byte	Value	Description
Packet category	1	0x63	Command category
Command type	2	0x04	Telemetry set type
Get Status identifier	3	0xFF	

Table 14: Trajectory set status request command packet

8.1.5 Position slave set

Position slave set request command is used to set the slave MAV address for coordinate system distribution. Coordinate system distribution is described in section 6.4. *Position slave set request command packet* consists of ten bytes. The structure of the *position slave* *set request command packet* is shown in table 15.

Packet Fields	Byte	Value	Description
Packet category	1	0x63	Command category
Command type	2	0x05	Position slave set type
Slave address	3-10	0xXXXXXXXXXXXXXXXX	8-byte slave MAV address

Table 15: Position slave set request command packet

Position slave set status request command is used to obtain *position slave set report packet* (section 8.3.5). *Position slave set status request command packet* consists of three bytes. The structure of the *position slave set status request command* is shown in table 16.

Packet Fields	Byte	Value	Description
Packet category	1	0x63	Command category
Command type	2	0x05	Position slave set type
Get Status identifier	3	0xFF	

Table 16: Position slave set status request command packet

8.1.6 Time

Time request command is used to set time on the MAV. Time is set in seconds in POSIX format. Time measurement is described in section 2.2. *Time request command packet* consists of six bytes. The structure of the *time request command packet* is shown in table 17.

Packet Fields	Byte	Value	Description
Packet category	1	0x63	Command category
Command type	2	0x06	Time type
Current time	3-6	uint32	Unsigned 4-byte integer in binary form

Table 17: Time request command packet

Time status request command is used to obtain *time report packet* (section 8.3.6). *Time status request command packet* consists of three bytes. The structure of the *time status request command packet* is described in table 18.

Packet Fields	Byte	Value	Description
Packet category	1	0x63	Command category
Command type	2	0x06	Time type
Get Status identifier	3	0xFF	

Table 18: Time status request command packet

8.1.7 Position set

Position set request command is used to set position of the MAV in the new coordinate system. Setting position of the MAV in the new coordinate system is described in section 4.1. *Position set request command packet* consists of ten bytes. The structure of the *position set request command packet* is shown in table 19.

Packet Fields	Byte	Value	Description
Packet category	1	0x63	Command category
Command type	2	0x07	Position set type
New elevator position	3-6	float	4-byte float in binary form
New aileron positon	7-10	float	4-byte float in binary form

Table 19: Position set request command packet

Position set type does not have status request command, because position of the MAV is one of the telemetry data.

8.2 Telemetry packets

Telemetry packets are used to transfer telemetry from MAV to ground station. These packets are sent by MAV periodically and consume most of the XBee transfer capacity. Telemetry monitoring is described in section 7.1. *Telemetry packets* start with telemetry category identifier (0x74 - t). *Telemetry packet* consist of $1 + 5k$ bytes, where k is number of transferred telemetry data. The structure of the *telemetry packet* is shown in table 20. All monitored telemetry data are transferred in one packet in order to increase payload/header ratio of the XBee API frames. Telemetry identifier and telemetry data are repeated in packet. Telemetry identifiers are shown in table 33.

Packet Fields	Byte	Value	Description
Packet category	1	0x74	Telemetry category
Telemetry identifier	2	0x00-0xFF	Identifier of telemetry data
Telemetry data	3-6	float	4-byte float in binary form
Telemetry identifier	7	0x00-0xFF	Identifier of telemetry data
Telemetry data	8-11	float	4-byte float in binary form
...	...	...	...

Table 20: Telemetry packet

8.3 Report packets

Report packets are used to transfer MAV states. MAV states are described in section 7.3. *Report packets* start with report category identifier (0x72 - r). Report type identifier is the second byte of the packet. The report type identifiers correspond with command type identifiers (tab. 6). The report types are shown in table 21.

Report type	Identifier	Description
Telemetry to Coordinator	0x01	Monitored telemetry types
Landing	0x02	Current landing state
Controllers	0x03	Currently active controller
Trajectory set	0x04	Trajectory way-points
Position slave set	0x05	Current slave address
Time	0x06	Current MAV time

Table 21: Report types

8.3.1 Telemetry to Coordinator

Telemetry to coordinator report is used to check whether chosen telemetry type is send to the ground station. *Telemetry to coordinator report packet* consists of four bytes. The structure of the *telemetry to coordinator report packet* is shown in table 22. Each telemetry type has specific identifier. Identifiers are shown in table 33.

Packet Fields	Byte	Value	Description
Packet category	1	0x72	Report category
Report type	2	0x01	Telemetry to coordinator type
On/Off	3	0x00/0x01	If telemetry data are send
Telemetry type	4	0x00-0xFF	Identifier of telemetry data

Table 22: Telemetry to coordinator report packet

8.3.2 Landing

Landing report is used to monitor current landing state. Landing states are described in section 2.2. Each landing state has specific identifier. Landing state identifiers are shown in table 31. *Landing report packet* consists of three bytes. The structure of the *landing report packet* is shown in table 23.

Packet Fields	Byte	Value	Description
Packet category	1	0x72	Report category
Report type	2	0x02	Landing type
Landing state identifier	3	0x00-0x04	Current landing state

Table 23: Landing report packet

8.3.3 Controllers

Controllers report is used to check which controller is currently active. Controllers are described in section 2.2. *Controllers report packet* consists of three bytes. The structure of the *controllers report packet* is shown in table 24. Each controller has specific identifier. Controller identifiers are shown in table 32.

Packet Fields	Byte	Value	Description
Packet category	1	0x72	Report category
Report type	2	0x03	Controllers type
Controller identifier	3	0x00-0xFE	Currently active controller

Table 24: Controllers report packet

8.3.4 Trajectory set

Trajectory set report is used to monitor current trajectory waypoints. *Trajectory set report packet* consists of $3 + 16k$ bytes, where k is number of trajectory waypoints. The structure of the *trajectory set report packet* is shown in table 25. All trajectory waypoints are in one packet in order to increase payload/header ration of the XBee frames.

Packet Fields	Byte	Value	Description
Packet category	1	0x72	Report category
Command type	2	0x04	Telemetry set type
Size	3	0x00-0xFE	Number of trajectory waypoints in packet
Time	4-7	uint32	Unsigned 4-byte integer in binary form
Elevator position	8-11	float	4-byte float in binary form
Aileron positon	12-15	float	4-byte float in binary form
Altitude	16-19	float	4-byte float in binary form
Time	20-23	uint32	Unsigned 4-byte integer in binary form
Elevator position	24-27	float	4-byte float in binary form
Aileron positon	28-31	float	4-byte float in binary form
Altitude	32-35	float	4-byte float in binary form
...	...	...	...

Table 25: Trajectory set report packet

8.3.5 Position slave set

Position slave set report is used to monitor current address of the slave MAV. *Position slave set report packet* consists of ten bytes. The structure of the *position slave set report packet* is shown in table 26.

Packet Fields	Byte	Value	Description
Packet category	1	0x72	Report category
Command type	2	0x05	Position slave set type
Current slave address	3-10	0xXXXXXXXXXXXXXXXX	8-byte slave MAV address

Table 26: Position slave set report packet

8.3.6 Time

Time report is used to monitor time on the MAV. *Time report packet* consists of six bytes. The structure of the *time report packet* is shown in table 27.

Packet Fields	Byte	Value	Description
Packet category	1	0x72	Report category
Command type	2	0x06	Time type
MAV time	3-6	uint32	Unsigned 4-byte integer in binary form

Table 27: Time report packet

8.4 Message Packets

Messages are described in section 7.4. *Message packets* are used to send string messages. *Message packets* start message with the message category identifier (0x6D - m). Size of the message packet is $1 + k$, where k is length of the message. Chars in message are coded in 8-bit ascii. Example of a *message packet* is shown in table 28.

Packet Fields	Byte	Value	Description
Packet category	1	0x6D	Message category
Char 1	2	0x48	H
Char 2	3	0x65	e
Char 3	4	0x6C	l
Char 4	5	0x6C	l
Char 5	6	0x6F	o
Char 6	7	0x20	space
Char 7	8	0x77	w
Char 8	9	0x6F	o
Char 9	2	0x72	r
Char 10	3	0x6C	l
Char 11	2	0x64	d

Table 28: Example of a message packet

9 Experiments

9.1 Telemetry monitoring

Telemetry monitoring feature is verified by measuring and logging MAV position drift. The MAV is equipped with Blob detector measuring position of the Blob as shown in figure 16. The Blob is on fixed position. The MAV is regulated to stay on its position. Elevator position and Blob elevator telemetry data are monitored. Telemetry monitoring is described in section 7.1. Position of the MAV estimated by Kalman filter is compared to the position of the MAV calculated from relative distance between the MAV and the Blob. Position drift of the MAV caused by noisy speed measurement is shown in figure 17. This drift is about 3,5 centimeters per second.

Figure 16: *Configuration of the MAV for the experiment*

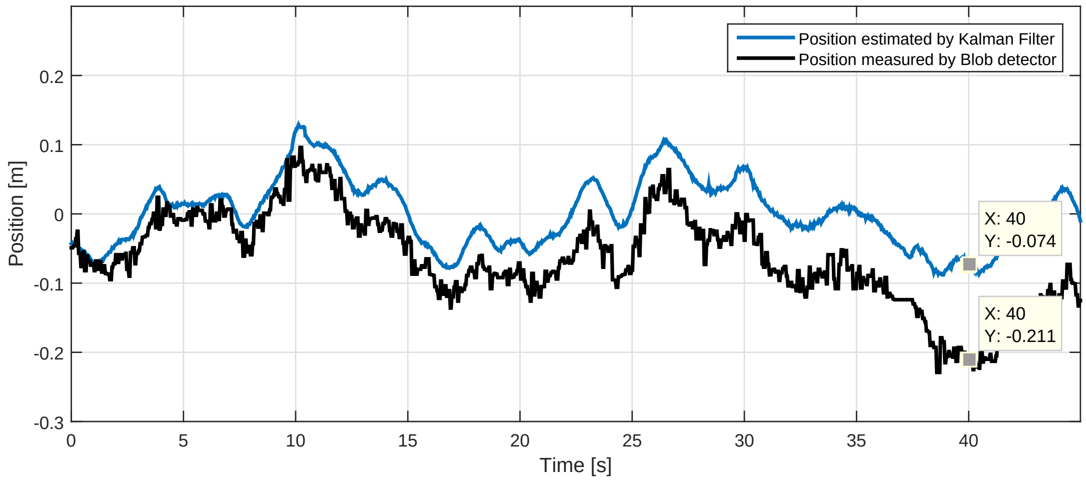


Figure 17: *Position of the MAV according to the Kalman filter and Blob detector*

9.2 Position change

Coordinate system change described in section 4.1 is verified by this experiment with one MAV. The MAV is hovering on a place using MPC controller. Elevator and aileron position setpoints are set to zero. Position of the MAV in the new coordinate system is set in 6th second and in 15th second. Position of the origin of the MAV coordinate system in the new coordinate system is shown in figure 18. This position is used to transform position of the MAV and the setpoint from the MAV coordinate system to the new coordinate system and back. In the MAV coordinate system, position of the MAV is not changed, the setpoint is changed. The position of the MAV and the setpoint in the MAV coordinate system are shown in figure 19a. Position of the MAV is transformed from the MAV coordinate system to the new coordinate system for monitoring. Position of the MAV and the setpoint in the new coordinate system are shown in figure 19b.

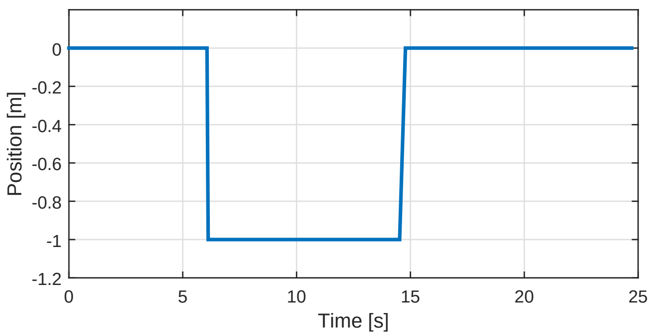
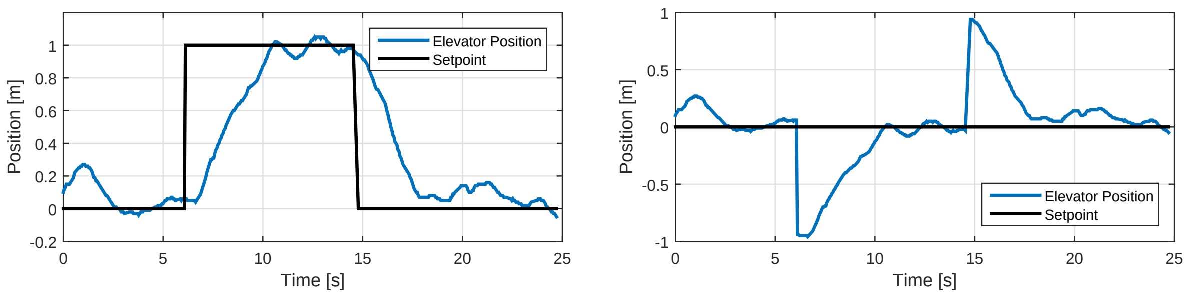


Figure 18: *Position of the origin of the MAV coordinate system in the new coordinate system*



(a) *Position in the MAV coordinate system* (b) *Position in the new coordinate system*

Figure 19: *Position of the MAV*

9.3 Position drift compensation

Coordinate system distribution and drift compensation described in section 6.4 are verified by this experiment. Two MAVs participate in the experiment. The MAVs are hovering on stable positions. Master MAV has elevator and aileron position setpoints set to zero. Slave MAV has elevator position setpoint set to two meters and aileron position setpoint to zero. The master MAV is equipped with Blob detector. The slave MAV is equipped with Blob. Configuration of the MAVs in this experiment is shown in figure 20. Photos from the experiment are shown in figure 24.

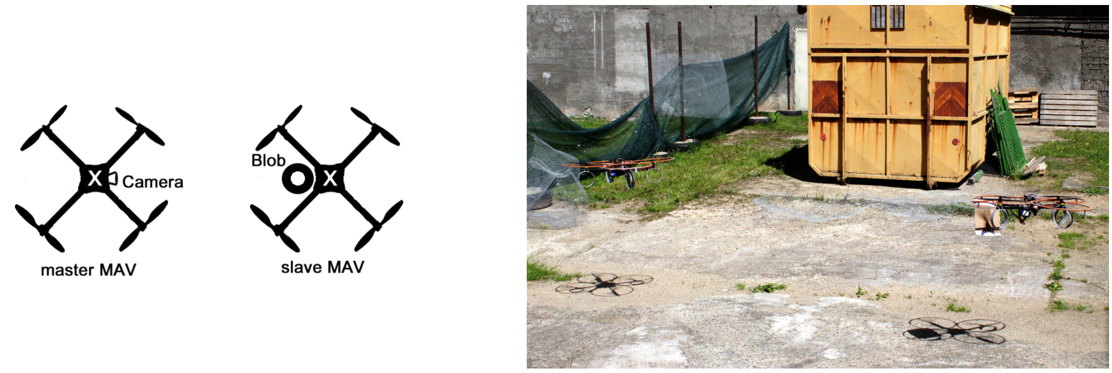
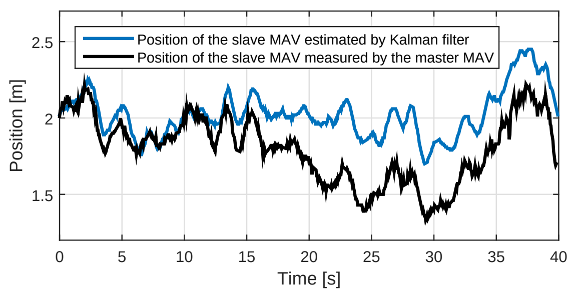
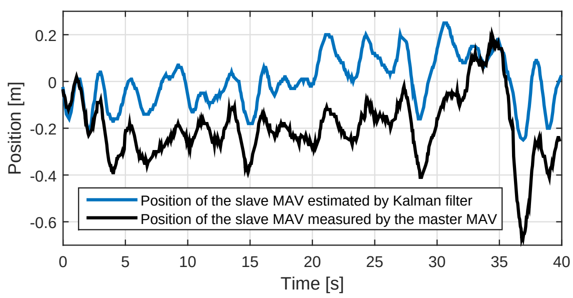


Figure 20: *Configuration of the MAVs for the experiment*

Global coordinate system is set to the slave MAV just once, and therefore position drift is not compensated. Position drift caused by noisy speed measurement and different yaw angles of the MAVs in the formation is shown in figure 21. Position of the slave MAV estimated by Kalman filter is compared to the position of the slave MAV calculated from the position of the master MAV estimated by Kalman filter and relative distance between the MAVs computed by the Blob detector.



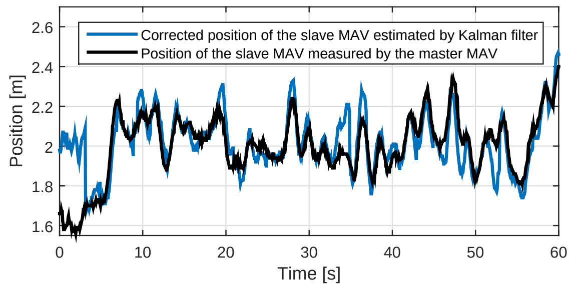
(a) *Elevator position of the slave MAV*



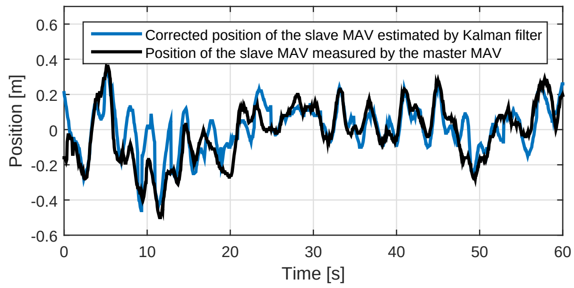
(b) *Aileron position of the slave MAV*

Figure 21: *Position of the slave MAV*

Global coordinate system must be distributed periodically in order to compensate position drift caused by noisy speed measurement and different yaw angles of the MAVs in the formation. Global coordinate system is distributed every two seconds. Position of the slave MAV is shown in figures 22 and 31. Position of the slave MAV estimated by Kalman filter is changed and corrected, therefore the relative position drift between the slave MAV and the master MAV is compensated. Drift correction shown in figure 23 is position of the origin of the slave MAV coordinate system in the global coordinate system.

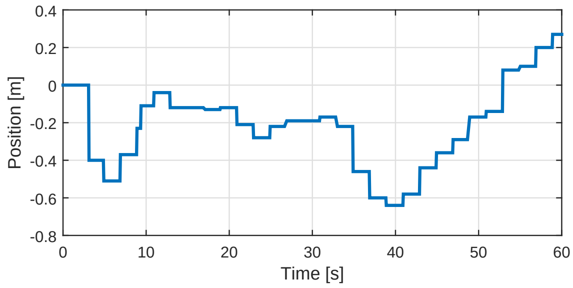


(a) *Elevator position of the slave MAV*

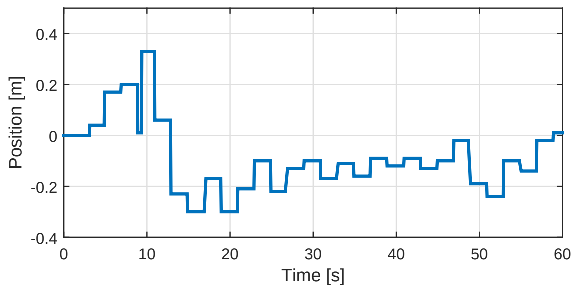


(b) *Aileron position of the slave MAV*

Figure 22: *Position of the slave MAV*



(a) *Elevator correction*



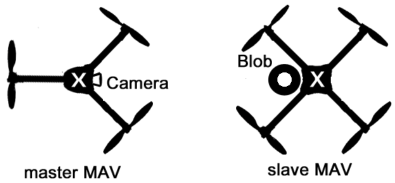
(b) *Aileron correction*

Figure 23: *Correction of the slave MAV position*

Figure 24: *Position drift compensation experiment*

9.4 Flight in the formation

Flight in the formation described in section 6 is verified by this experiment. Master and slave MAVs participate in the experiment. The master MAV is equipped with Blob detector. The slave MAV is equipped with Blob. Configuration of the MAVs is shown in figure 25. Because of technical issues with Gumstix Blob detector, the master MAV is replaced by tricopter platform. This MAV is equipped with Blob detector based on Raspberry Pi [16].



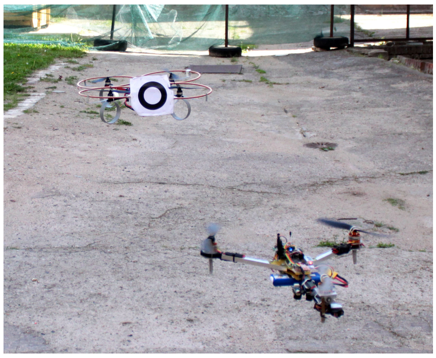


Figure 25: *Configuration of the MAVs for the experiment*

The master MAV distributes the global coordinate system to the slave MAV every two seconds, hence compensate the relative drift between the MAVs. Each MAV flights its own trajectory independently of the other MAVs in the global coordinate system, hence different shapes of the formation is flown. Shape of desired trajectories of the MAVs is shown in figure 26. Desired trajectory and real trajectory of the MAVs are shown in figure 32. The progress of trajectory following in the formation can be seen in photos from the experiment in figure 27. MPC regulator is used for trajectory following. Trajectory is described in section 5.

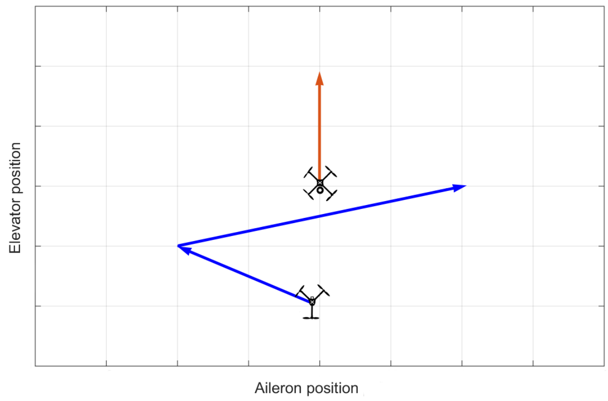


Figure 26: *Shape of the trajectories*

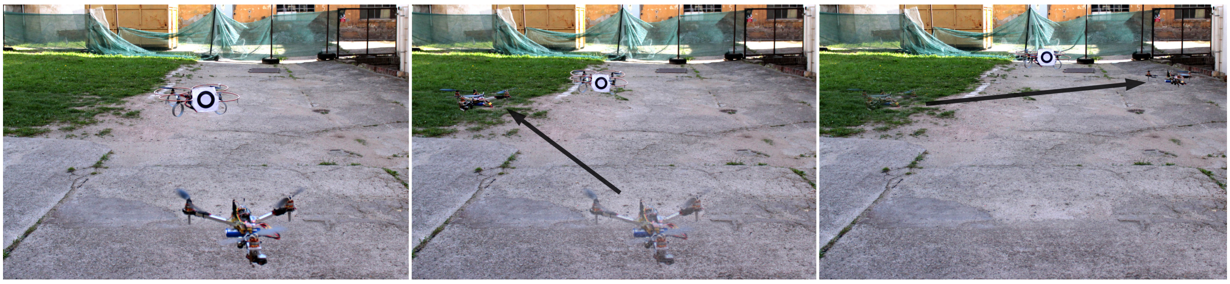
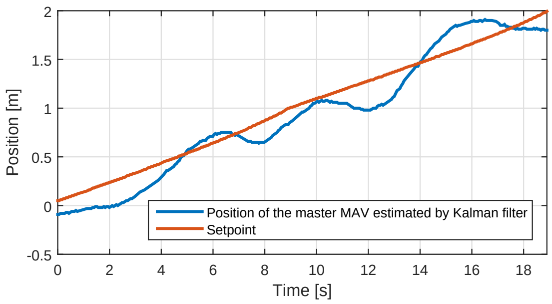
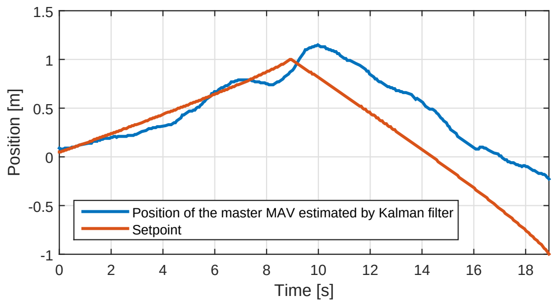


Figure 27: *Flight in the formation*

The progress of trajectory following of the master MAV is shown in figure 28. The progress of trajectory following of the slave MAV is shown in figure 29. Drift correction shown in figure 30 is the position of the origin of the slave MAV coordinate system in the global coordinate system.

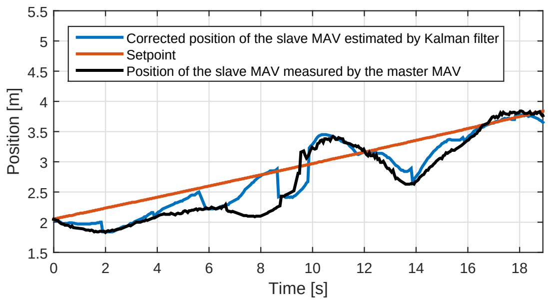


(a) *Elevator position*

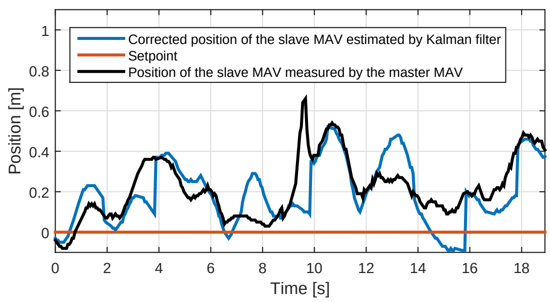


(b) *Aileron position*

Figure 28: *Position of the master MAV*

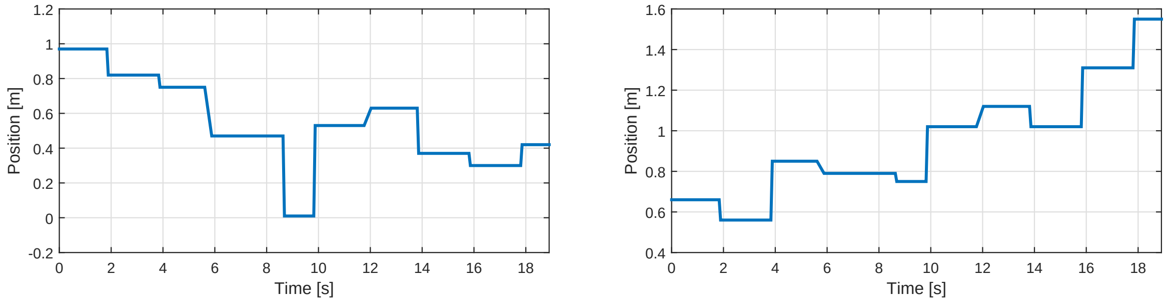


(a) *Elevator position*



(b) *Aileron position*

Figure 29: *Position of the slave MAV*



(a) *Elevator correction* (b) *Aileron correction*

Figure 30: *Correction of the slave MAV position*

Video from the experiment is on attached CD and can be seen on youtube (`www.youtube.com/watch?v=76oemQtmwoM`).

10 Conclusion

All goals of this thesis are fulfilled. A new protocol for MAV communication called MCP is implemented. MAVs are able to communicate with each other and with the ground station using the MCP. The ground station operator is able to monitor the MAV telemetry online. MAV can be controlled by command packets. MAV movement can be dynamically changed by uploading new trajectry waypoints. The ground station operator is able to land with the whole group of MAVs at once in case of danger. Time on MAVs can be synchronized with the ground station. Position of the MAV can be set in a new coordinate system. The direct communication between MAVs is used for more precise relative stabilization of the MAVs in the formation. One of MAVs coordinate systems is chosen as the global coordinate system. The global coordinate system is distributed in the formation using the direct communication between MAVs, therefore coordinate systems of all MAVs in the formation are unified. The position drift of the MAVs in the formation caused by noisy speed measurement is compensated if the global coordinate system is distributed periodically. MAVs positions and trajectory waypoints are measured in the global coordinate system. Each MAV is independent unit which flies its own trajectory in the global coordinate system independently from the other MAVs, hence various shapes of formation can be flown. System presented in this work is verified by several indoor and outdoor experiments with real MAVs.

The system is ready to be used for experimental verification of other developed algorithms of MAV control and multi-robot systems. Relevant information and results in the field of MAV control and multi-robot systems in general, which were achieved by other members of Multi-robot Systems group, can be found in [20, 21, 22, 23, 24, 25].

References

[1] Vicon. http://www.vicon.com/.

[2] Mohamed Abdelkader, Mohammad Shaqura, Christian G Claudel, and Wail Gueaieb. A uav based system for real time flash flood monitoring in desert environments using lagrangian microsensors. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 25–34. IEEE, 2013.

[3] Amazon. Amazon prime air. http://www.amazon.com/b?node=8037720011.

[4] Tomáš Báča. Model predictive control of micro aerial vehicle using onboard micro-controller. Master's thesis, Czech technical university in Prague, Faculty of Electrical Engineering, 2015.

[5] TU Delft. Ambulance drone. http://www.tudelft.nl/en/current/latest-news/article/detail/ambulance-drone-tu-delft-vergroot-overlevingskans-bij-hartstilstand-drastisch.

[6] Václav Endrych. Control and stabilization of an unmanned helicopter following a dynamic trajectory. Master's thesis, Czech technical university in Prague, Faculty of Electrical Engineering, 2014.

[7] Gumstix. Gumstix caspa vl. https://store.gumstix.com/index.php/products/260/.

[8] Gumstix. Gumstix overo. https://store.gumstix.com/index.php/category/33/.

[9] Cheng Hui, Chen Yousheng, and Wong Wing Shing. Trajectory tracking and formation flight of autonomous uavs in gps-denied environments using onboard sensing. In *Guidance, Navigation and Control Conference (CGNCC), 2014 IEEE Chinese*, pages 2639–2645. IEEE, 2014.

[10] Czech Technical University in Prague. Multi-robot systems group. http://mrs.felk.cvut.cz/research/micro-aerial-vehicles.

[11] Digi International Inc. *XBee®/XBee-PRO® ZB RF Modules*, 2014.

[12] Tomas Krajnik, Matias Nitsche, Jan Faigl, Petr Vanek, Martin Saska, Libor Preucil, Tom Duckett, and Marta Mejail. A practical multirobot localization system. *Journal of Intelligent & Robotic Systems*, 76(3-4):539–562, 2014.

[13] Taeyoung Lee, Melvin Leoky, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on se (3). In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 5420–5425. IEEE, 2010.

[14] Sergei Lupashin, Markus Hehn, Mark W Mueller, Angela P Schoellig, Michael Sherback, and Raffaello D'Andrea. A platform for aerial robotics research and demonstration: The flying machine arena. *Mechatronics*, 24(1):41–54, 2014.

[15] MikroKopter. Mk basicset l4-me. https://www.mikrocontroller.com/.

[16] Raspberry Pi. https://www.raspberrypi.org/.

[17] Pixhawk. Px4flow smart camera. https://pixhawk.org/modules/px4flow.

[18] Rajeev Piyare and Seong-ro Lee. Performance analysis of xbee zb module based wireless sensor networks. *International Journal of Scientific & Engineering Research*, 4(4):1615–1621, 2013.

[19] QGroundControl. Mavlink micro air vehicle communication protocol. http://qgroundcontrol.org/mavlink/start.

[20] M. Saska, J. Chudoba, L. Preucil, J. Thomas, G. Loianno, A. Tresnak, V. Vonasek, and V. Kumar. Autonomous Deployment of Swarms of Micro-Aerial Vehicles in Cooperative Surveillance. In *Proceedings of 2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, volume 1, pages 584–595, Danvers, 2014. IEEE Computer society.

[21] M. Saska, Z. Kasl, and L. Preucil. Motion Planning and Control of Formations of Micro Aerial Vehicles. In *Proceedings of The 19th World Congress of the International Federation of Automatic Control*, pages 1228–1233, Pretoria, 2014. IFAC.

[22] M. Saska, T. Krajnik, J. Faigl, V. Vonasek, and L. Preucil. Low Cost MAV Platform AR-Drone in Experimental Verifications of Methods for Vision Based Autonomous Navigation. In *Proceedings of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 4808–4809, Piscataway, 2012. IEEE.

[23] M. Saska, T. Krajnik, V. Vonasek, Z. Kasl, V. Spurny, and L. Preucil. Fault-Tolerant Formation Driving Mechanism Designed for Heterogeneous MAVs-UGVs Groups. *Journal of Intelligent and Robotic Systems*, 73(1-4):603–622, January 2014.

[24] M. Saska, T. Krajnik, V. Vonasek, P. Vanek, and L. Preucil. Navigation, Localization and Stabilization of Formations of Unmanned Aerial and Ground Vehicles. In *Proceedings of 2013 International Conference on Unmanned Aircraft Systems*, pages 831–840, New York, 2013. Springer.

[25] M. Saska, J. Vakula, and L. Preucil. Swarms of Micro Aerial Vehicles Stabilized Under a Visual Relative Localization. In *ICRA2014: Proceedings of 2014 IEEE International Conference on Robotics and Automation*, pages 3570–3575, Piscataway, 2014. IEEE.

[26] M. Saska, V. Vonasek, T. Krajnik, and L. Preucil. Coordination and Navigation of Heterogeneous MAV-UGV Formations Localized by a hawk-eye-like Approach Under a Model Predictive Control Scheme. *International Journal of Robotics Research*, 33(10):1393–1412, September 2014.

[27] SparkFun. Openlog. https://github.com/sparkfun/OpenLog.

[28] Gábor Vásárhelyi, Cs Virágh, Gergő Somorjai, Norbert Tarcai, T Szorenyi, Tamás Nepusz, and Tamás Vicsek. Outdoor flocking and formation flight with autonomous aerial robots. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3866–3873. IEEE, 2014.

Appendix A Protocol Identifiers

Packet category	Identifier
Command	0x63 (c)
Telemetry	0x74 (t)
Report	0x72 (r)
Message	0x6D (m)
Warning	0x77 (w)

Table 29: Packet category identifiers

Command type	Identifier
Telemetry to coordinator	0x01
Landing	0x02
Controllers	0x03
Trajectory set	0x04
Position slave set	0x05
Time	0x06
Position set	0x07

Table 30: Command type identifiers

Landing state	Identifier
On ground	0x00
Landing	0x01
Stabilization	0x02
Take off	0x03
Flight	0x04

Table 31: Landing state indetifiers

Controller	Identifier
Manual control	0x01
Speed controller	0x02
Position controller	0x03
MPC controller	0x04

Table 32: Controllers identifiers

Telemetry	Identifier
Estimated altitude	0x00
Altitude	0x01
Elevator speed	0x02
Aileron speed	0x03
Estimated elevator speed	0x04
Estimated aileron speed	0x05
Elevator position	0x06
Aileron position	0x07
Altitude controller output	0x08
Altitude speed	0x09
Aileron controller output	0x0A
Elevator controller output	0x0B
Altitude setpoint	0x0C
Elevator position setpoint	0x0D
Aileron position setpoint	0x0E
Elevator acceleration	0x0F
Aileron acceleration	0x10
Valid Blob	0x11
Output throttle	0x12
Output elevator	0x13
Output aileron	0x14
Output rudder	0x15
Blob elevator	0x16
Blob aileron	0x17
Blob altitude	0x18
Pitch angle	0x19
Roll angle	0x1A
Elevator shift	0x1B
Aileron shift	0x1C
Elevator acceleration input	0x1D
Elevator acceleration error	0x1E
Aileron acceleration input	0x1F
Aileron acceleration error	0x20

Table 33: Telemetry identifiers

Appendix B Experimental figures

This appendix is described in section 9.

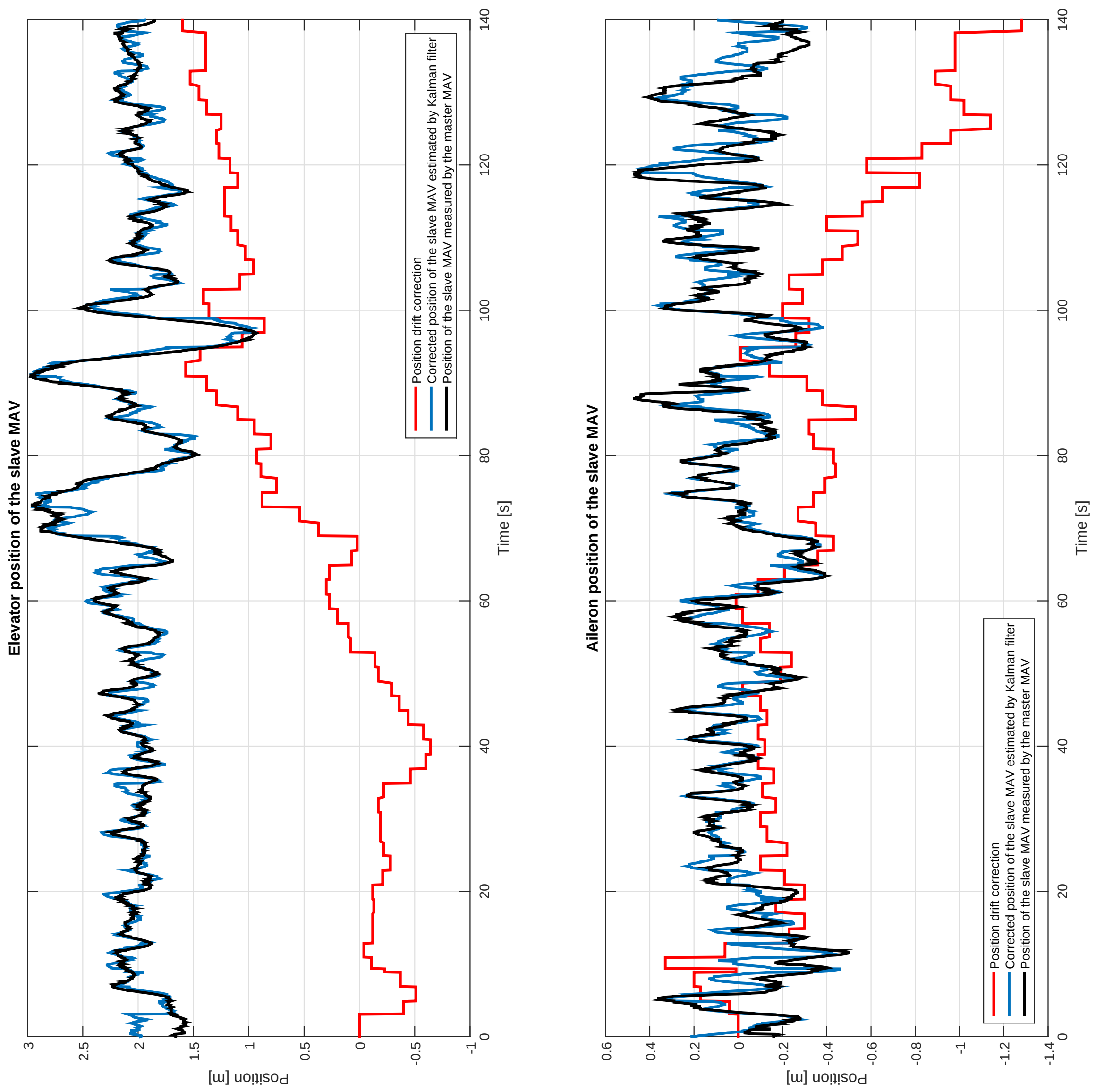


Figure 31: *Position of the slave MAV*

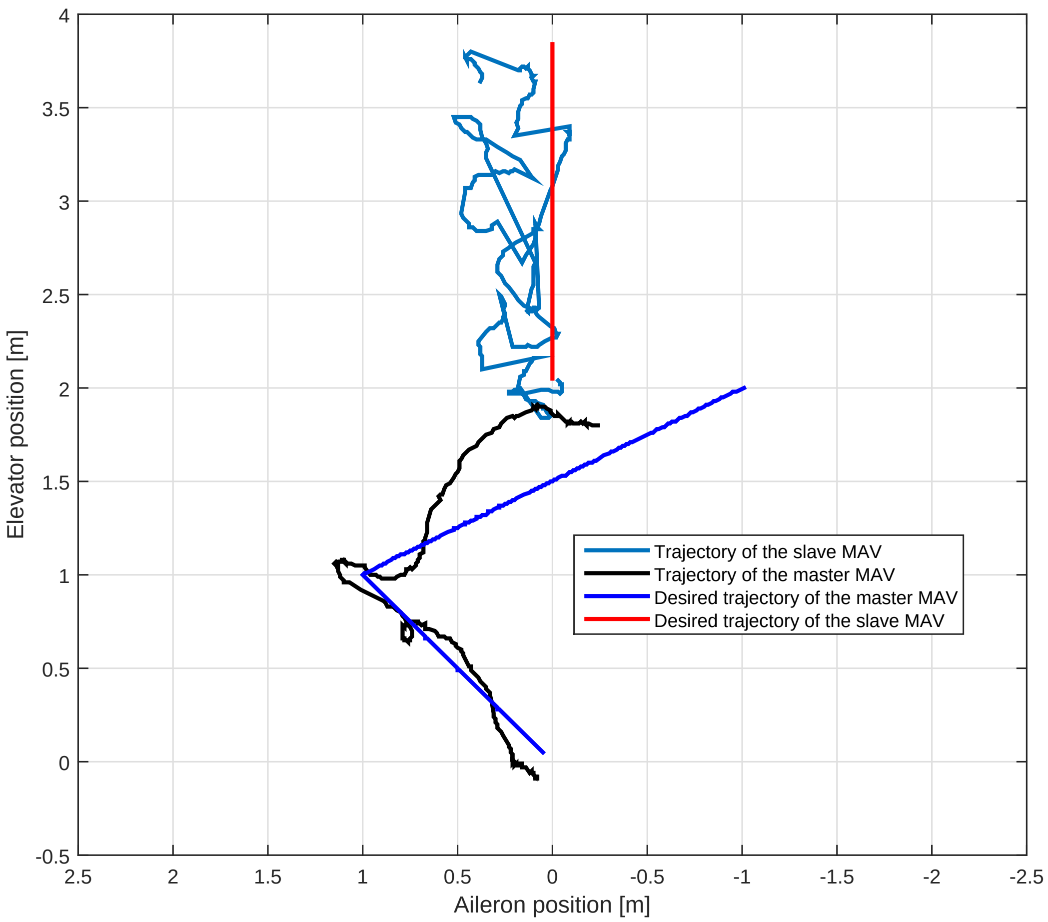


Figure 32: *Trajectory following*

Appendix C CD Content

In Table 34 are listed names of all root directories on CD.

Directory name	**Description**
thesis	Bachelor's thesis in pdf format.
thesis_sources	latex source codes
video	Video from the flight in the formation experiment

Table 34: CD Content

Appendix D   List of abbreviations

In Table 35 are listed abbreviations used in this thesis.

Abbreviation	Meaning
API	application programming interface
MAV	micro aerial vehicle
MCP	MAV communication protocol
MPC	model predictive control
UART	universal asynchronous receiver-transmitter
MCU	microcontroller unit
PPM	pulse position modulation
ACK	acknowledgement
GPS	global position system
SLAM	simultaneous localization and mapping

Table 35: Lists of abbreviations