1918 TALLINNA TEHNIKAÜLIKOOL

INFOTEHNOLOOGIA TEADUSKOND

Automaatikainstituut

Reaalajasüsteemide õppetool

Jaanus Kaugerand

ISP70LT

Autonomous Unmanned Aerial System as a Component in a Tactical System of Systems

Magistritöö

Juhendaja: Jürgo-Sören Preden
reaalajasüsteemid
vanemteadur

Tallinn 2014

AUTORIDEKLARATSIOON

Olen koostanud antud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud. Käesolevat tööd ei ole varem esitatud kaitsmisele kusagil mujal.

Kuupäev:

Autor: Allkiri:

Autonoomne mehitamata lennuk taktikalise süsteemide süsteemi komponendina

Annotatsioon

Käesoleva magistritöö eesmärk on võtta kasutusele autonoomne mehitamata õhu-süsteem (mehitamata lennuk, inglise keeles Unmanned Aerial System), mis teeb koostööd autonoomse maapealse sensorsüsteemide võrgustikuga (inglise keeles Unattended Ground Sensor System), eesmärgiga pakkuda sensorsüsteemi poolt avastatud sündmuste kohta visuaalset infot. Töö käigus tehakse kindlaks nõuded ja piirangud autonoomsele mehitamata õhu-süsteemile, luuakse nimetatud õhu-süsteem ning integreeritakse see maa peal paikneva sensorsüsteemiga. Töö kirjeldab ka lõpliku süsteemiga tehtud katsete tulemusi ning järeldusi.

Lõputöö on kirjutatud inglise keeles ja sisaldab teksti 67 leheküljel, 4 peatükki, 28 joonist ja 6 tabelit.

Autonomous Unmanned Aerial System as a Component in A Tactical System of Systems

Abstract

Autonomous UAS are seldom considered as a component in a larger System of Systems. Current thesis reviews the challenges in integrating a simple mini UAS capable of autonomous flight with an existing System of Systems (SoS) consisting of a ground sensor systems network, with the objective to provide visual information about the events detected by the ground sensor systems. The thesis describes the creation of the UAS, its integration with an SoS and the evaluation of the resulting SoS.

This thesis is written in English and contains 67 pages of text, 4 chapters, 28 figures and 6 tables.

Table of Contents

1 INTRODUCTION....1

1.1 Objective of the thesis....2

1.2 Description of the application scenario....2

1.3 Outline of the thesis....4

2 BACKGROUND FOR CREATING AN AUTONOMOUS UAS....5

2.1 Definition of Autonomous Unmanned Aerial System (UAS)....5

2.2 Legal issues applicable to control of autonomous UAS....6

2.3 Brief overview of NATO STANAG on UAV control....8

2.4 Description of the MAVLink protocol....10

2.5 Aspects of Systems of systems....12

2.5.1 Definition of autonomy....12

2.5.2 Taxonomy of autonomy....13

2.5.3 Context of measuring the autonomy....15

2.5.4 System of Systems definition....15

2.5.4.1 Components of SoS as autonomous and communicative agents....17

2.5.5 Characteristics of Systems of Systems....17

2.5.5.1 Autonomy....18

2.5.5.2 Belonging....19

2.5.5.3 Connectivity....19

2.5.5.4 Diversity....19

2.5.5.5 Emergence....19

2.5.6 Autonomous control on contemporary UAS....20

2.6 Data association....21

2.6.1 Detection and classification....21

2.6.2 Aided classification and situation identification....22

3 CREATING AN AUTONOMOUS UAS....23

3.1 Components of the SoS in the Application Scenario....23

3.1.1 Net Relay Server....24

3.1.2 Ground control station....24

3.1.3 The UGS network....25

3.1.4 UAS platform for ISR SoS....25

3.2 Description of the UAS on-board components....27

3.2.1 Fuselage....27

3.2.2 Servos and motors....28

3.2.3 Batteries....28

3.2.4 On-board remote control receiver....28

3.2.5 Payload....29

3.2.6 Autopilot....29

3.2.6.1 AKPilot IO ports....30

3.2.6.2 The different states of AKPilot....31

3.2.7 Embedded system for UAS autonomous control....32

3.2.8 Modem....33

3.3 Description of software implemented for UAS....35

3.3.1 The requirements for the software....35

3.3.2 UAS software architecture for the high level control module....36

3.3.3 Pilot Control module....37

3.3.4 The Mission Control module....38

3.3.4.1 The mission plan generator....38
3.3.4.2 Uploading a mission plan to AKPilot....40
3.3.4.3 Execution of the mission plan....40
3.3.4.4 UAV camera control....41
3.3.5 Internal and external communication....42

4 TESTING....43
4.1 Laboratory tests....43
4.1.1 Generating a simple plan, uploading it and running the simulation on it....43
4.1.2 Testing the UAS-GCS communication over Net Relay server....43
4.1.3 Testing software components on Gumstix and Raspberry Pi embedded systems…44
4.1.4 Testing the UAS in full configuration in the laboratory....45
4.2 Flight tests....46
4.2.1 Testing UAS without payload....46
4.2.1.1 Lessons learned (10.05.2014; 16.05.2014; 18.05.2014):....46
4.2.2 Testing UAS with payload....47
4.2.2.1 Lessons learned (23.05.2014):....47
4.2.3 Testing UAS in full mission configuration....47
4.2.3.1 Lessons learned (30.05.2014):....48
4.2.3.2 Lessons learned (31.05.2014):....50

5 CONCLUSION....52

RESÜMEE....54

REFERENCES....58

List of Figures

Figure 1: Asymmetric threat detection scenario....3
Figure 2: UAV family tree....5
Figure 3: STANAG for UAV control....8
Figure 4: MAVLink packet structure....10
Figure 5: Mission upload....11
Figure 6: OODA loop....14
Figure 7: Contextual autonomous capability model....15
Figure 8: Structure of a cyber-physical system....16
Figure 9: SOS architecture....23
Figure 10: HappyKillmore Ground Control Station....24
Figure 11: Vaatlusdroon UAS....25
Figure 12: AKPilot....29
Figure 13: AKpilot connections....30
Figure 14: Raspberry Pi....33
Figure 15: UAS mission scenario....35
Figure 16: UAS software architecture....36
Figure 17: Pilot Control module....37
Figure 18: MAVLink mission example....39
Figure 19: Camera control procedure....41
Figure 20: Internal and external communication model....42
Figure 21: UAS testing in laboratory....45
Figure 22: Generated mission plan downloaded from AKPilot....45
Figure 23: Example of UAS camera quality from 100m hight....47
Figure 24: The first testing of the application scenario....48
Figure 25: Target 2, image 3....49
Figure 26: Target 3, image 2....49
Figure 27: Second testing of the application scenario....50
Figure 28: Target 2 image 1....51

List of Tables

Table 1: MAVLink packet structure....10
Table 2: Sheridan's model....13
Table 3: Example of autonomy spectrum and OODA Loop....14
Table 4: Characteristics of a System of Systems....18
Table 5: States/modes of AKPilot....31
Table 6: Gumstix Overo on Pinto-TH expansion board vs Raspberry Pi....32

Appendices

Appendix I: Implemented MAVLink messages in Pilot Control software....57

Glossary of Abbreviations and Symbols

AKPilot	Autopilot, that has its name by its programmer Andrus Kangro
COTS	Commercial off the shelf
CPS	Cyber-physical system
GPS	Global Positioning System
GCS	Ground Control Station
HMI	Human machine interface
ISR	Intelligence, Surveillance and Reconnaissance
ProLab	The Laboratory for Proactive Technologies
LGPL	GNU Lesser General Public License
MAS	Multi Agent System
MAV	Micro air vehicle
MAVLink	Micro air vehicle link
NMEA	National Marine Electronics Associaton
NATO	North Atlantic Treaty Organization
MT	Master thesis
SA	Situational Awareness
STANAG	Standardization Agreement
UAV	Unmanned Aerial Vehicle
UAS	Unmanned Aerial System
UGS	Unattended Ground Sensor

1 Introduction

During recent decades, the development of autonomous unmanned aerial vehicles has been rapid in both military and civilian domain. Today there are already applications where aerial vehicles are working together in teams, but almost no examples can be found where autonomous air vehicles are teamed up with autonomous ground sensors systems to form a larger autonomous System of Systems (SoS). The goal of the current thesis is to implement a tactical autonomous unmanned aerial system (UAS) that collaborates with a tactical autonomous network of ground sensor systems with the purpose of offering visual information about the events detected by the sensor systems. In the context of the thesis, the requirements and constraints for such UAS are explored, and the UAS is designed, implemented and integrated with the network of sensor systems on the ground. Finally the results are evaluated against a realistic application scenario presented in the first chapter, which is a scenario of a perimeter control for a temporary military base in an asymmetric threat environment. The scenario is an adaptation of the scenario from the ongoing European Defence Agency project IN-4-STARS in the Research Laboratory for Proactive Technologies (ProLab) at Tallinn University of Technology.

The UAS developed in the context of the thesis will be used in the IN-4-STARS project.

The basis for UAS is an autonomous tactical reconnaissance unmanned aerial vehicle (UAV), designed and custom built the for current thesis by a team of enthusiasts (“Vaatlusdroon”). The name Vaatlusdroon was chosen when author of the thesis together with Andrus Tamboom participated in entrepreneurship-competition “Ajujaht” in 2011 with the then latest model of teams self-built UAV. The design of the Vaatlusdroon UAS has been inspired by a half-military reconnaissance competition “Põrgupõhja Retk” annually organized by Estonian Defence League. Vaatlusdroon team has been participating in this competition for several years. The UAV together with necessary equipment has been carried in rucksack to the reconnaissance task and successfully used to gain intelligence as described in three competition scenarios. This is also the source of authors interest into autonomous unmanned aerial vehicles.

The thesis explores the requirements, designs the overall architecture, chooses and implements the components in order to add the autonomous communication and mission planning and execution ability to a UAV. During the process, ProLab offered assistance in

both procurement of the necessary components and also implementing the individual software components.

The tactical Intelligence, Surveillance and Reconnaissance (ISR) SoS discussed in current thesis and presented in the related case study is designed and implemented by ProLab, during the actual testing of the UAS the ground sensor systems are emulated.

1.1 Objective of the thesis

The thesis explores and evaluates the requirements for the system components that must be added to an operator controlled UAS to facilitate the integration of the UAS to the SoS and provides the design for the required hardware and software. The main objective is to implement the autonomous UAS according to the operational and technical requirements derived from the application scenario of the IN-4-STARS project, assuming an SoS architecture for the ISR system. Finally the thesis will evaluate the result during the testing of the implemented UAS according to the application scenario.

A sub-objective of the thesis is to identify the limitations and possibilities for establishing data connectivity between the UAS and the UGS. For that purpose one of the ground sensor nodes is installed on board UAS in order to create a possibility to monitor the performance of a wireless ad hoc communication when one of the nodes is moving in and out of the communication range of the UGS.

1.2 Description of the application scenario

The perimeter control scenario for the current thesis foresees that the tactical sensor network is used for force protection and the UAS is utilized to augment the situational picture by providing visual information on detected objects and events. The scenario is described below briefly, and visualized on Figure [1].

The scenario describes a complex system for identifying and mitigating the asymmetric threat to a temporary military base perimeter. The components in such a system are Unattended Ground Sensor (UGS), UAS and possible data fusion nodes. The data fusion aspects are not discussed in the thesis as they fall outside the scope of the current work. A UGS is essentially an autonomous computer based sensing system, equipped with a power supply, a set of sensors, and typically a wireless communication interface [1]. In the context of current thesis the collection of these systems is considered a tactical and autonom-

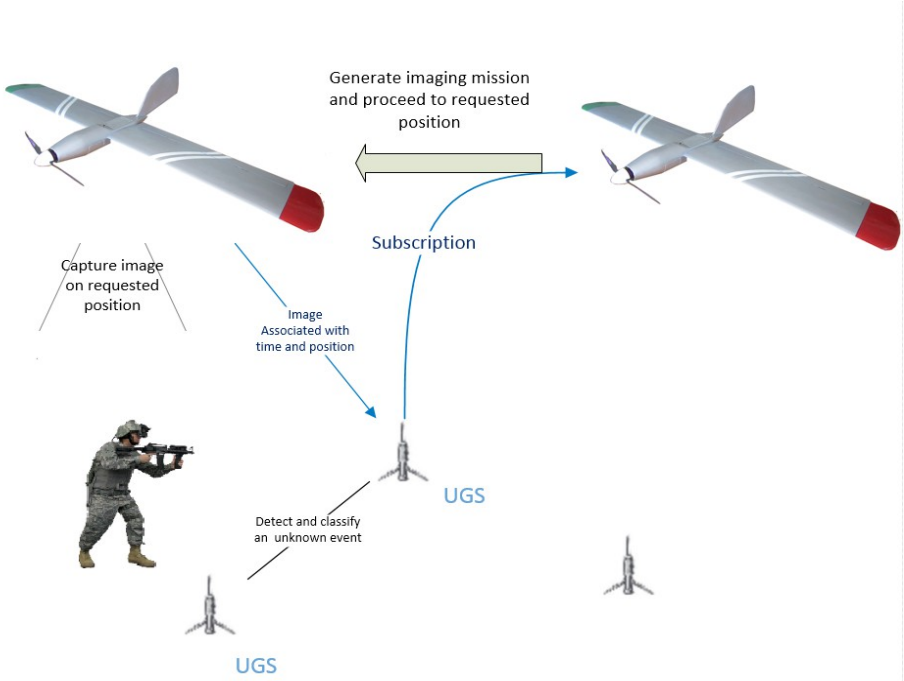

Figure 1: Asymmetric threat detection scenario

ous System of Systems (SoS). The SoS concept is explained later in chapter 2.5.

The Scenario description is the following: the UAS is launched and configured to remain in a standby mode (i.e. hold a circling pattern) in a specific area. 10 minutes after the UAS has been launched and has assumed the standby position, a mobile object of interest (e.g. a vehicle of the red force) is detected by UGS.

The UGS establishes contact to UAS and communicates the instructions (i.e. coordinates and type of sensor to use) from where the images are required to enhance the situational picture (i.e. where object of interest is located).

The UAS will then relocate to the new position, take pictures/video of the area of interest and transmit the pictures to the data requester (UGS), that will combine the data with the fused sensor data from UGS and communicate the data to the (human) data consumer. In case many information requests are received, the requests are handled by order of priority and order of arrival.

1.3 Outline of the thesis

The thesis is divided into 4 major chapters. Chapter one gives the introduction, outline and objectives of the thesis. Chapter 2 gives background for creating a UAS, chapter 3 describes the details of implementation and chapter 4 evaluates the results by conducting both laboratory and flight testing.

Chapter 2 starts off by introducing the definition of the Unmanned Aerial Vehicle. Section 2.2 defines the legal requirements relevant for the UAS operation. Section 2.3 explains NATO allied nations standardization of UAS control. Section 2.4 explains the communication protocol used for Commercial Off The Shelf (COTS) micro air vehicles. Section 2.5 discusses the definition of System of Systems, autonomy and also the measuring of autonomy in contemporary military unmanned aircraft. Section 2.6 introduces the concept of data association. Thesis describes the work performed by the author in:

1. Chapter 3.2 where the overall design of the UAS is developed, including the on board components:
   - The author identified the additional components to transform the UAV to a component of an SoS.(choice of payload, the embedded computer and communication interface.
   - The author also provided input to the design and assembly of the rest of the components (legacy components).
2. In chapter 3.3 the software implemented on UAS is described:
   - The author implemented the mission planning software and the software for uploading the mission to the autopilot, sending the mission related commands to the autopilot, and parsing the messages sent by autopilot.
   - The author also provided input on overall design of the SoS software.
3. In chapter 4 the testing procedures and results of the UAS are described:
   - The author planned and conducted the laboratory tests described in chapter 4.
   - The flight tests were planned and carried out in cooperation with the author, Team “Vaatlusdroon” and ProLab staff. The author managed the negotiations about the test scenarios and the testing schedule.

2 Background for creating an autonomous UAS

2.1 Definition of Autonomous Unmanned Aerial System (UAS)

Unmanned Aerial System (UAS), also called a drone, is a powered and guided aircraft with no human pilot on board. Sakamoto [2] gives a good overview of the UAV definition, see Figure [2]. It brings out the clear difference between a conventional aircraft and a UAV. Current thesis uses the term UAS instead of UAV in order to emphasize the complexity of the on-board systems of autonomous unmanned aircraft considered in the current thesis – while a regular UAV can be an aircraft with no autonomous operation capabilities, the UAS discussed in the current thesis is capable of autonomous operation, task evaluation and execution. A UAS can be controlled either remotely by a human operator on ground using a communication link, or autonomously by an on-board computer.

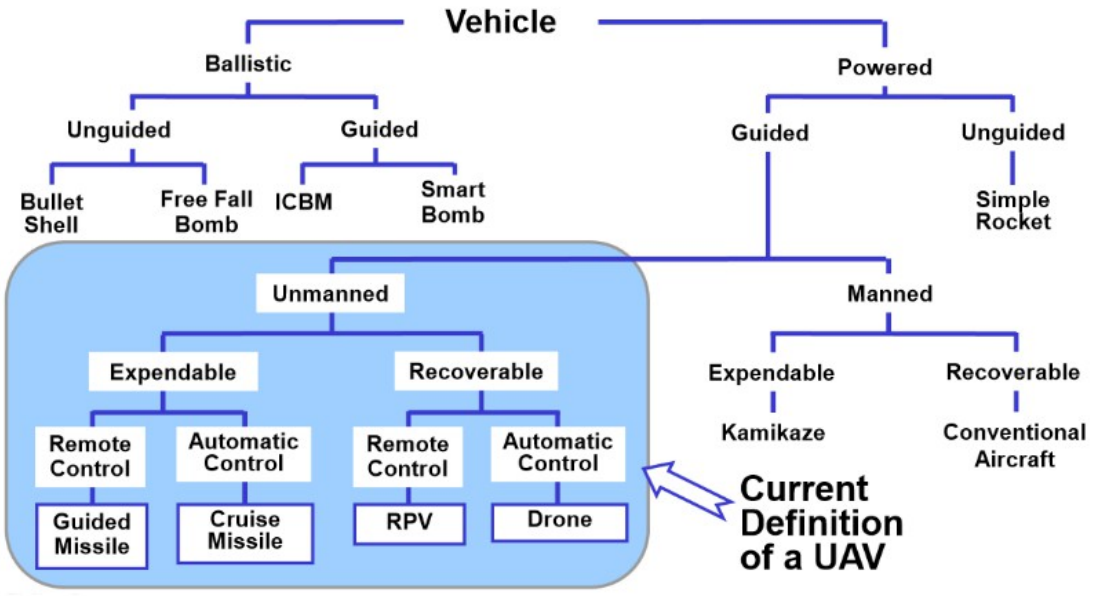

Figure 2: UAV family tree [2]

(RPV – Remotely piloted vehicle, ICBM – Inter Continental Ballistic Missile)

The idea of an autonomous pilotless aircraft is not a new concept, attempts to achieve this date back even to the middle of the 20th century. Today there are working examples of different applications for UAS in both civilian and military sector. In civilian practice, the most common tasks are various surveillance, inspection and mapping tasks, in military the main usage of UAS is for intelligence, surveillance and reconnaissance (ISR) tasks, but

also carrying out armed attacks on enemy.

There are several advantages inherent to designing an aircraft with no human pilot on board. Many tasks are too difficult or not suitable for humans, these tasks are called as dull, dirty and dangerous (D3). Operating unmanned aircraft involves less risk to human lives and operational costs can be smaller. The design of aircraft does not need to consider the space for pilots, instead the freed space could be used for other purposes, e.g. more complex payload, fuel, electronic equipment or propulsion. The aircraft could be smaller as it does not need to carry pilots on board, this also lowers costs for propulsion systems and can make aircraft more fuel-efficient. The aircraft could even be disposable and be used for missions which involve risk of loss of airframe (i.e. monitoring natural disasters), which in case of a manned aircraft would mean a risk to human lives.

The unmanned aircraft also has its downsides. More often than not the sensor information gathered is far too extensive for on-board processing and needs to be fed back to Ground Control Station (GCS) where information can be displayed to human operators who can make the decisions which in turn need to be communicated back to aircraft. This sets either limits on aircraft operating range or/and requires extra infrastructure for relaying the communication between GCS and the aircraft. Another disadvantage of unmanned aircraft is high dependence on GPS satellite system for positioning and timing.

2.2 Legal issues applicable to control of autonomous UAS

The purpose of the legislative regulations in the domain of unmanned vehicles is to ensure safety. The legislation concerning autonomous or remote controlled UAV is not implemented in Estonia at the moment. A draft act has been prepared [3], and author of current thesis together with team Vaatlusdroon and Estonian UAV community has contributed to the draft with own proposals [4] during February of 2013. The resulting rules are expected to be the following:

the UAV-s are classified in 3 categories as:

- IA, takeoff weight below 5kg (proposed by Vaatlusdroon team)
- I, takeoff weight below 25kg
- II, takeoff weight below 150kg
- III, takeoff weight above 150kg

For current thesis the relevant category is IA. The IA category UAS can be operated

without coordination with Air Traffic Control (ATC) below a flight height of 150m. The UAS has to stay within visibility range and within 1000m from the takeoff point. In case longer distance is needed the UAV would according to the draft act automatically advance to category II. Category II UAVs must already be registered in the Aviation Office and Aviation permit is needed. Also the remote operator must have taken the piloting exam. The draft act does not currently have provisions for the case then UAV is completely autonomous. One of the proposals in [4] was that in case of category IA autonomous plane should be allowed to travel longer distances than 1km and then only ATC would have to be contacted and presented with a flight plan and the approval of the flight plan could be given if not in conflict with other traffic. Another proposal was to have less restricted rules for scientific and research activities.

In case the flight of unmanned aircraft is in close proximity to airfields or air traffic control areas the ATC coordination is always required.

In any case if unmanned aircraft is outside the visual line of site the operator must monitor the UAS constantly. It is not fully clear how the term monitoring should be interpreted, does the UAS have to stay within visual sight or is it sufficient to monitor the UAS over the GCS. While with the GCS the operator is able to monitor the movement of UAS on a geographical map, there is also an alternative for UAS control, which is the First Person View (FPV), where the operator monitors the UAS flight through special goggles where the video view from the forward camera on board UAS is displayed.

The legal aspects may be easier to implement in military domain as in principle the military traditional top-down command and control structure ensures that there is always someone in command and thus responsible. There are mainly ethical and moral discussions about armed UAVs as it is generally accepted that the decision to commit a weapon against a target is made by a human. It is technically quite feasible to give this decision to an unmanned system, but considerations of collateral risk, and rules of engagement, make this is generally inappropriate.

Richard M. O'Meara, presented in his "Intersection" at a robotics law conference "We Robot 2012" that from various humanitarian law treaties there are five general rules regarding the conduct of warfare [5].

1. A general prohibition on the employment of weapons of a nature to cause superfluous injury or unnecessary suffering,

2. military necessity,
3. proportionality,
4. discrimination, and
5. command responsibility.

There rules must be observed when designing military UAS solutions, as when developing new technology used as a weapon in warfare, each country has a responsibility to determine if it is in accordance with international law [6].

2.3 Brief overview of NATO STANAG on UAV control

As the current thesis discusses application of a UAS for military purposes as part of a larger system, a review of the relevant standards for connecting UAS with other systems is appropriate. NATO uses standardization agreements called STANAGs to provide common military or technical procedures and to define processes, procedures, terms, and conditions for common military or technical procedures or equipment for NATO members (or nations cooperating with NATO). STANAGs also form the basis for technical interoperability between a wide variety of communication and information (CIS) systems essential for NATO and Allied operations [7].

The UAV control procedures are standardized in STANAG 4586. It specifies the generic architecture of a UAV control system, a good overview of standards used in relation to UAV operation is given by Terryl Bandzul [8], see Figure [3].

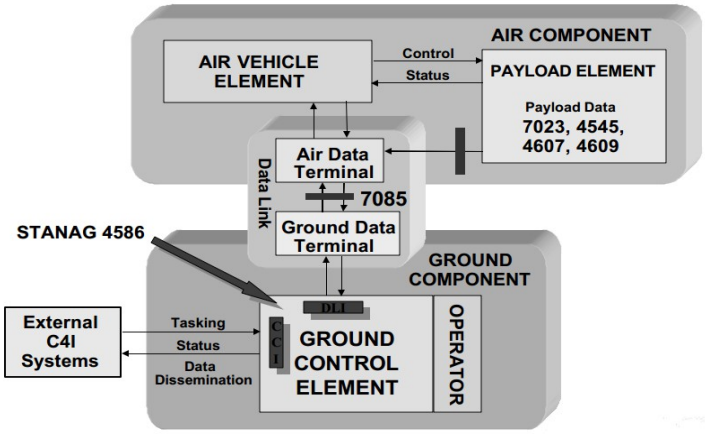

Figure 3: STANAG for UAV control [8]

The main goal of STANAG 4586 is to achieve UAV Systems Interoperability through mainly standardising the two interfaces:

1. Data Link Interface (DLI)
2. Command and Control Interface (CCI)

The DLI provides for standard messages and formats to enable communication between a variety of air vehicles and NATO standardised control stations. The CCI specifies the data requirements that should be adopted for communication between the UAV control systems and all C4I end users through a common, standard interface [9].

Another relevant NATO document for UAV is STANAG 7085 which defines a Common Data Link (CDL) for ISR systems. This STANAG provides the interoperability standards for 3 classes of data links used for imagery data transmission: analogue links, point-to-point digital links, and broadcast digital links. The STANAG is structured in a way it provides a number of options for the specific data link configuration, such as simplex or duplex operation, data rate, carrier frequency, channel multiplexing, interleaving, encryption, and many others that must be matched prior to passing data from transmitter to receiver [10].

The other STANAGs shown on the Figure [3] are 7023, 4545, 4607 and 4609 which provide standards for the data interfaces for digital sensors as payload.

The technological standards trends in STANAG development currently lean towards bandwidth reduction, toward network-centric CONOPS, service-oriented architecture, increasing system autonomy (Autonomy messages added to the STANAG 4586, operators are to become system supervisors in future) and toward collaboration among systems [8].

2.4 Description of the MAVLink protocol

The MAVLink protocol is designed as a header-only message library and was first released in early 2009 by Lorenz Meier under LGPL [11]. The protocol is designed for use between micro air vehicle autopilots and Ground Control Stations (GCS).

MAVLink message management library defines the messages for:

- vehicle attitude / telemetry
- GPS messages
- mission related commands and reports
- messages to read and write parameters (e.g. PID gains)

In current thesis the MAVLink protocol is relevant as it is used for controlling the auto-pilot (listening to GPS messages, autopilot reports and vehicle attitude, sending commands to autopilot).

MAVLink frame is 8 – 263 bytes long and its packet structure is depicted on Figure [4].

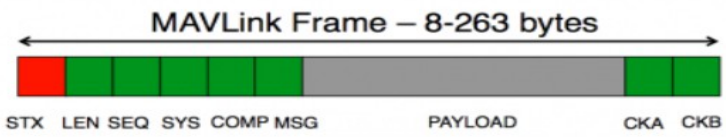

Figure 4: MAVLink packet structure [12]

For closer explanation of each byte segemt see Table [1].

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	ITU X.25/SAE AS-4 hash, excluding packet start sign, so bytes 1.. (n+6)	

Table 1: MAVLink packet structure [12]

The current version in use with most today's micro air vehicles is MAVLink version 1.0. The communication protocol implemented on board AKPilot, the control unit on the UAS developed in the thesis, is MAVLink version 0.9. Most of the Ground Control Stations available have implicit support for both versions.

An example of waypoint list request from Micro Air Vehicle (MAV) autopilot can be seen of Figure [5].

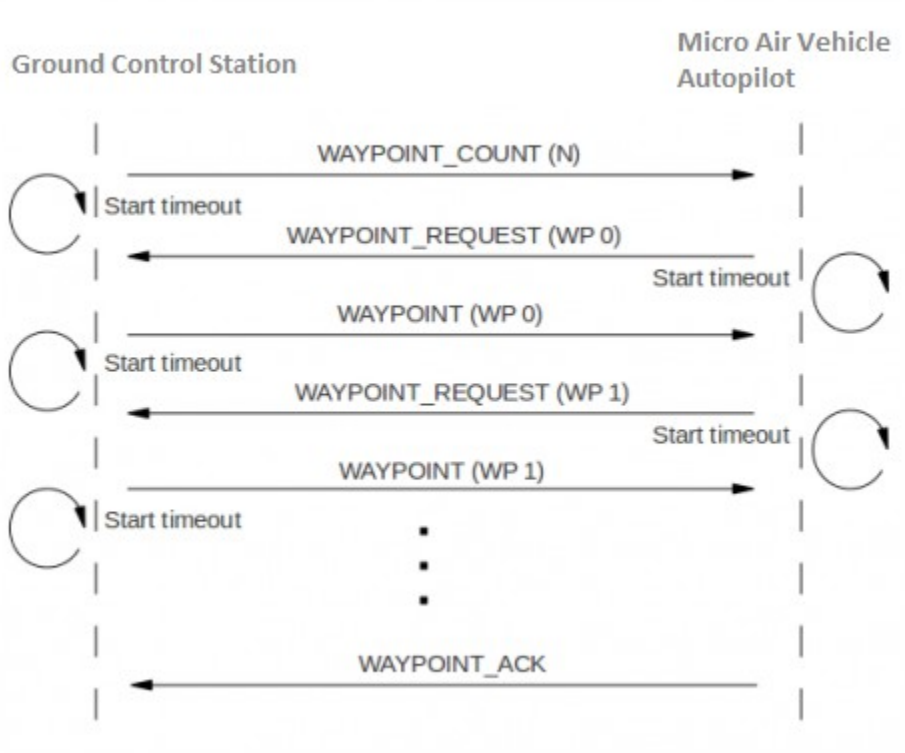

Figure 5: Mission upload [12]

In order to interpret and compose the MAVLink messages a python PyMavlink module the PyMavlink dialect "V09 Common" is used. This is as its name also indicates a dialect with the most common functions needed for a generic autopilot. The MAVLink procedures used in current thesis are explained in chapter 3.3.

2.5 Aspects of Systems of systems

In order to understand the context and requirements for the SoS aspects of the UAS, a short review of the SoS aspect is presented below.

The System of Systems (SoS) is, as its name implies, a system which consists of systems. The definition of SoS which will be explored in chapter 2.5.4, provides a more abstract view of a system under design and helps to concentrate more on behaviours and interactions of the constituent systems and to relate them to the goal of the emergent system. This definition of is closely related to the theory of autonomous agents and multi agent systems (MAS). The current chapter provides an overview of necessary definitions and in addition to SoS explores more thoroughly the notion of autonomy and how autonomy is measured.

2.5.1 Definition of autonomy

The word comes from the Greek words autos (self) and nomos (governance). US National Institute of Standards and Technology defines autonomy for unmanned system as: "An unmanned systems own ability of sensing, perceiving, decision-making, and acting/executing, to achieve its goals as assigned by its human operator(s) through designated HRI" [13] or simply "its own capability to achieve its mission goals" [14]. There are numerous more examples but the substance remains same. An artificial entity can be considered autonomous when it can function independently over extended period of time, perceive its surrounding environment, comprehend at least some part of the environment, decide by itself what is the best action in order to achieve its goal, and act in an optimal way on this decision. The definition of autonomy must not be mixed with the definition of automation, although they can be heavily overlapping. The purpose of automation as well as of autonomy is to lessen the workload of humans, via the capacity to operate without outside intervention [15]. But the definition of automation does not include the decision making process. Especially the decisions over whether or not to and when to provide service that is subscribed by other components or decisions to initiate interactions with other nodes. An automated machine executes preprogrammed actions in a certain fixed sequence, though the complexity of those preprogrammed actions may be very high. Another differentiating factor is that an automated entity operates mostly in a bounded static environment while autonomous entity operates in open dynamic environment. Taking decisions in an uncertain environment is

not a trivial task, the process requires hierarchical buildup of the situational awareness that is derived from perceiving and comprehending of the environment. Endsley (1988) gives the following definition for the situational awareness: “the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future” [16]. The situation awareness does not have to be an individual trait, in an SoS designed according to multi agent system it is achieved collectively by exchanging situational parameters in order to gain a situational awareness of a team [17].

2.5.2 Taxonomy of autonomy

With the recent development in the field of UAVs there have been several attempts to create a taxonomy for levels of autonomy. The most known studies in this field are Sheridan's model and OODA loop.

The Sheridan's model [18] describes machine's independence of human as a continuum from the entity being completely controlled by a human, through the entity being completely autonomous and not requiring input or approval of its actions from a human before taking actions, see Table [2].

Computer:
1. offers no assistance, human must do it all
2. offers a complete set of action alternatives
3. narrows the set of alternatives down to a few
4. suggests one alternative
5. executes the chosen alternative if the human approves
6. allows the human a restricted time to veto before automatic execution of the selected alternative
7. executes automatically, then necessarily informs the human
8. informs human after execution only if he asks
9. informs human after execution if it, the computer, decides to
10. decides everything and acts autonomously, ignoring the human.

Table 2: Sheridan's model [18]

Another way to measure autonomy is to use an OODA model developed by military

strategist John Boyde in the 70s, see Figure [6].

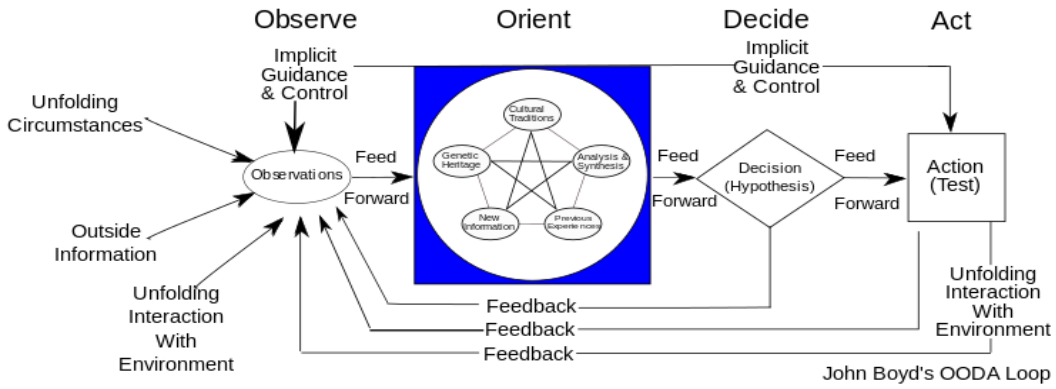

Figure 6: OODA loop [19]

The OODA loop was originally developed in order to formalize the decision cycle for gaining an advantage in military operations [20]. It has four dimensions: Observe, Orient, Decide and Act. According to Boyde the decision cycle is a recurring loop over the those four dimension repeated hundreds of times during a mission. A fighter aircraft pilot would Observe the situation around him, Orient himself about the situations, Decide upon the next action and Act on the decision taken. The key to success here is the completion of the single loop of OODA faster than the opponent.

Today the OODA model has also proven very useful in describing how machine systems like UAVs operate, make decisions and interact with the world [20]. Each dimension of OODA loop can further be assessed in a Sheridan model like scale. This way the problem of measuring the machine independence from human can be divided into smaller tasks. An example of such taxonomy is shown on Table [3].

Level	Observe	Orient	Decide	Act
0	Flight control sensing and on board camera	Telemetry data; remote pilot commands.	None. Off-board pilot.	Control by remote pilot.
5	Local sensors to detect external targets, fused with off-board data.	Group action diagnosis and resource management.	On-board trajectory planning; optimize for current & predicted conditions; collision avoidance.	Group accomplishment of tactical plan as externally assigned; air collision avoidance.
10	Cognizant of all within the battlespace.	Coordinates as necessary.	Capable of total independence.	Requires little guidance of any sort.

Table 3: Example of autonomy spectrum and OODA Loop [20]

There can be found also a lot of references to definitions like human being “In the Loop”, “On the Loop” or human completely taken “Out of the Loop” [10]. These are robust generalizations of human either assisting machine in some or every step of OODA loop or human only observing and only intervening in extreme cases or machine observing, orienting, deciding and acting completely autonomously.

2.5.3 Context of measuring the autonomy

According to the levels of autonomy explained in previous chapters, it is difficult if not impossible to compare two autonomous vehicles. For this purpose the context of mission and environmental complexity must be used. Between 2003 and 2008 the US National Institute of Standards (NIST) has published several works on Autonomy Levels For Unmanned Systems (ALFUS). The main goal for this taxonomy is to evaluate and document the requirements and capabilities of unmanned vehicles. In a nutshell, the levels of autonomy are measured in the following three dimensions: human independence, mission complexity and environmental complexity [21], as illustrated on Figure [7].

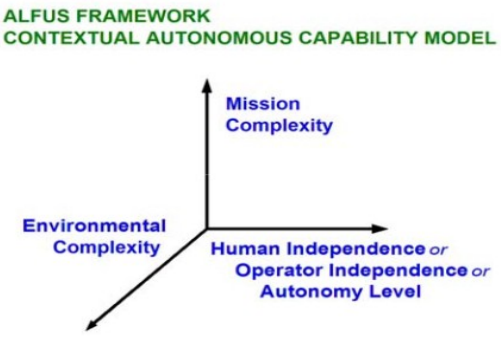

Figure 7: Contextual autonomous capability model [21]

While the Mission Complexity and Environmental Complexity dimensions provide the contextual background, the autonomy level itself is on the third axis. In order to evaluate autonomy level on autonomy axis the ALFUS is also using Sheridan's model.

2.5.4 System of Systems definition

With technological development, systems have grown larger and more complex. New systems can have several components which are systems themselves and where interactions

among the system components and with the environment are more important than the internal design of the constituent subsystems. Furthermore, the environment in which the complex systems are carrying out their tasks, is often dynamic, open and unpredictable. Components in such a complex system may not have fixed roles and internal structure does not necessarily have to be static. These aspects bring in a new level of complexity and have resulted in a new concept – a System of Systems.

The study of System of Systems is not older than a couple of decades and has its origins from large corporations operating in a field of aerospace where complex systems like spacecraft, military fighter aircraft or commercial aircraft are being developed. Today the examples of systems that can be designed according to those principles can be found in industry, power distribution networks, and also from fields of study of smart houses or even smart cities consisting of fully autonomous entities, which interact with each other in order to create more efficient infrastructure and work together in order to achieve a superior goal, that each part of the SoS would not be capable of achieving by itself.

As the SoS discussed in the context of the current thesis interacts with the physical world, the resulting systems can be considered to be Cyber-physical systems (CPS). A CPS is is an integration of computation with physical processes, in Figure [8] there is depicted an example structure of a cyber-physical system [22].

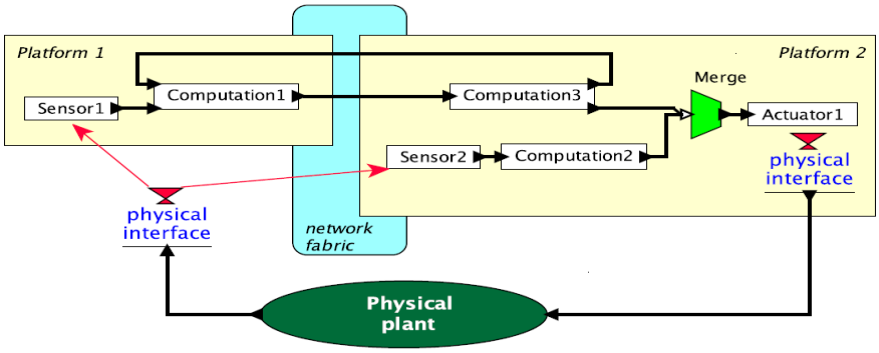

Figure 8: Structure of a cyber-physical system [22]

A CPS is a system which consists of several actors (computational platforms), it may have networks connecting the actors and by definition it interacts with the environment (physical processes).

One example of the complexity of the system which could be considered as a System of Systems is a Boeing 787 Dreamliner. An interesting case is under media attention since December 2012, the latest update was in March 2014 when it was still unsolved. The Boe-

ing engineers are having trouble finding what causes the battery system failure. The system which fails may not be of big importance, the battery system itself is very robust system and works in other systems, but now it is put into a very complex airplane which involves several autonomous components, and the result is intermittent failures. Prof. Steven Eppinger, a researcher from MIT, comments on this: “In complex systems, you can have thousands of interconnections of parts, which will create unanticipated and unintended interactions. It is likely that one or more of those interactions is causing this failure in the Dreamliner’s battery system” [23].

2.5.4.1 Components of SoS as autonomous and communicative agents

In computer science the term agent is an abstraction that allows to concentrate on interactions and behaviours rather than internal mechanisms. Stan Franklin and Art Graesser give following definition: “An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future” [24]. In other words the agent is a computational system that is autonomous, is able to perceive dynamic environment and have a goal oriented reactive or proactive behaviour according to its comprehension and computational capabilities. The agent may be capable of optimizing or adopting its actions and may be able to learn from previous experience. Furthermore, in the context of current thesis, it must be emphasized that agents are considered to have capability to exchange information among each other - they are communicative. This ability can also be considered as implicitly required and existing if the surrounding environment contains other agents with the same agenda or higher goal, and they need to work together in order to achieve that goal. Of course this sets more complex technical requirements for the physical implementations of systems (components in an SoS) that must act similarly to such autonomous and communicative agents.

2.5.5 Characteristics of Systems of Systems

Despite of the various definitions of the concept of an SoS, it can still often be difficult to differentiate between when a system is a System or SoS. One of the more comprehensive overviews on the definition of SoS comes from the paper published by Broadman and Sauser in 2006. They defined five characteristics [25] that describe the SoS: autonomy,

connectivity, belonging, diversity and emergence. In following every one of those characteristics shall be explained in more detailed and compared against a monolithic system.

	System	System of Systems
Autonomy	Parts give up autonomy to system.	Goal driven autonomy by parts.
Belonging	Parts are akin to family members; they are organic part of the System.	Constituent systems have similar goals with the SoS and choose to belong on a cost/benefits basis;
Connectivity	High connectivity hidden in elements, and minimum connectivity among major subsystems.	Dynamically supplied by constituent systems, possibly via a net-centric architecture.
Diversity	Minimized by modular hierarchy; parts' diversity encapsulated to protect simplicity into the next level of hierarchy.	Increased diversity in SoS capability achieved by released autonomy, committed belonging, and open connectivity.
Emergence	Foreseen, both good and bad behaviour, and designed in or tested out as appropriate.	Enhanced deliberately by creating an emergence capability climate.

Table 4: Characteristics of a System of Systems [25]

2.5.5.1 Autonomy

The principal difference between autonomy in a monolithic system and an SoS is illustrated on Table [4]. In a monolithic system the components of system are not autonomous, they are being controlled by the system and the system cannot fullfill its purpose without its parts. In a System of Systems each constituent system exercises autonomy in order to achieve the overall goal of an SoS.

When looking for examples in the scale of autonomy characteristic, we could in one end have system like a UAS that for some reason cannot complete its mission without its link to ground control station (and operators), while on the other end of the scale we could have components of a sensor network designed according to contemporary understanding of the principles of Systems of Systems and multi-agent networks. In latter case the total system would be scalable, consisting of autonomous components, and robust. Autonomous UAS as a component in such system would continue with its mission and contribution to achieving the goal of SoS autonomously using its best abilities even if the connectivity to other components of SoS is intermittently lost.

2.5.5.2 Belonging

According to Broadman's and Sauser's characteristics, the belonging shows the willingness of a system to pay the costs of offering some service or to collaborate with other systems in return to adding a new value to the systems own purpose and role and to enhance the possibility of achieving both the system's own goal and the goal of a System of Systems. In other words the belonging indicates also if a system or a device is an organic part of a larger system, or chooses to be part of a System of Systems because it is beneficial. The choice of collaboration obviously assumes also a certain degree of autonomy.

2.5.5.3 Connectivity

Connections and interactions among the components of SoS are considered as important as the components themselves. In systems engineering it has become a rule that if the connections between components are huge, they are hidden away, encapsulated. This cannot be the case in SoS as the constituent components are autonomous and decide largely by themselves when an interaction is initiated or which connections must be created among the components, see Table [4].

2.5.5.4 Diversity

Diversity may come in many forms, it may manifest itself in various forms of interactions, behaviours, perception and comprehension etc. Any task may require certain amount of diverse skills. In case of SoS, it is not difficult to see that more diversity among its components results in an ability to solve more complex tasks. The diversity also enhances the SoS odds for survival in complex environment and also possibility for emergent behaviour.

2.5.5.5 Emergence

High level of autonomy, loose belonging and connectivity, and high level of diversity in a System-of-Systems leads to synergism, adaptivity and emergent behaviour and the other way around if synergism and emergence is desired, high level of autonomy, loose belonging, loose connectivity and high level of diversity is required. In such a System-of-Systems the emergent behaviour is unpredictable, but is highly welcomed as it facilitates the ability to cope with the unexpected nature of the environment. This is also that something that distinguishes the System from System-of-Systems.

On the other hand it is an open question whether emergence can actually exist in a system made up entirely of engineered components. Edmonds, B., in his PhD thesis (1999) "Syntactic measures of complexity" has eloquently posed this as the question of emergence versus ignorance – in other words, is an unpredictable result truly a feature of the complex system, or merely an artefact of our lack of understanding? Recent work has been undertaken that seeks to develop a mathematical test of this hypothesis, but the effort is still in its infancy [26].

2.5.6 Autonomous control on contemporary UAS

It is interesting to note that the most autonomous and famous contemporary UAS, like MQ-1 Predator and MQ-9 Reaper by General Atomics and RQ-4 Global Hawk by Northrop Grumman, only score 0-2 on the given Autonomy spectrum described in Table [3]. This is due to the fact that in reality these are remotely operated aircraft. For example a single Predator's crew consists of one pilot and two sensor operators, but it takes a total of 82 people, including technical support staff, to fly it [27].

Today in US military the discussions concerning autonomy of both the contemporary and also the newest autonomous UAS not in service yet (i.e. Boeing Phantom Eye (2013), autonomous take off, landing and flight, Northrop Grumman X-47B (2012), autonomous take off and landing on aircraft carrier), are not yet about designing fully autonomous UAS, but instead, several sources describe the US military endeavours to climb couple of levels up on autonomy levels to achieve the capability for the UAS to require less operators and maybe even having one operator to pilot several UAS. At the moment commonly at least two operators are required to pilot one UAS.

The biggest constraints for the UAS being fully autonomous is the lack of capability to comprehend the perceived sensor information at a level necessary to take the decisions needed for autonomous behaviour. Another shortfall is the high dependence on positioning service provided by GPS in their movement and path planning. Though some examples can be found of UAS operating without GPS, (i.e. using pre-mapped 3D models [28] or coastline or a road following algorithms which are said to be as good as human [29]) but those are still scientific studies and conducted by universities.

2.6 Data association

Data association means creating relations between data components. In current thesis it is used in context of adding temporal and spatial metainfo to a captured image. This is used when the UAS records an image in a position according to a subscription received from ground sensors and uploads it back to the subscriber. Having exact time and position associated with the image gives the final user a possibility to put the images into a larger context by comparing the image to the:

- Image from same area at an earlier time.
- Image taken almost simultaneously from a different position (images acquired with overlap and with short time interval could be even fused together)
- Other sensor measurements from same time and position.

A good example for the term "Data Association" can be found from military domain, where the principle is often used for associating detected radar contacts into usable information, for example radar tracks. This is achieved by relating an event with time and position plus having an historical overview.

When different actors exchange data associated with temporal and spatial metainfo, it enables a receiver to also validate the information both in time space in order to determine if the data can be used in the first place.

2.6.1 Detection and classification

In military terminology detection is: "In tactical operations, the perception of an object of possible military interest but unconfirmed by recognition" [30]. The event or object can be detected by sensors using some predefined signal characteristics thresholds. The process of detection can be conducted by either a fully autonomous cyber physical system, by a human with a computer aided system or by a human alone. Today the data acquired by sensors is mostly analysed by some computing system, and detection process is partially or fully automatic. After an event or object has been detected, the next step is to classify it. The simplest way to classify it in military terms is to decide whether the detected event or object is either probable fake or probable real object or event of interest like signal characteristics. Certainly when more measurements are fused/combined, a more detailed classification may hierarchically go further including for example classification steps like: noise on ground → possible ground vehicle → possible slow ground vehicle firing a gun → pos-

sible enemy slow ground vehicle.

The classification of events or objects is performed based on data acquired by sensors. The acquired sensor measurements can be processed using classification algorithms, and with certain confidence be categorized into predefined classes. The resulting events or objects classified and associated with spatial and temporal metadata can be used to create situational awareness in humans or machines. Owing to the fact that situational awareness is crucial in military operations, the common practice is still to have the human operator behind the sensor HMI making the decisions about identification and classification. This is mainly due to the very nature of situations that are taking place in the military theatre of operations. It is still easier and faster to brief human operators who can also draw conclusions on their own from their experience, than to reprogram the machines on situational context. This is to say that the criteria for classification can be very complex in terms of situational awareness.

2.6.2 Aided classification and situation identification

Already for several decades, ironically due to the development of sensor technology and cyber-physical systems, we have faced the complication of the sensors producing too much information to be processable by human operators. One solution also presented in this thesis is to split the tasks (and solve them by machines) into smaller pieces by using several autonomous agents that together form a large System of Systems that dynamically handles each detection, classification and identification case by interacting within its components.
The recognition and identification can be done in several ways. Current thesis presents an identification approach where a tactical System of Systems, containing a set of autonomous sensors, is used for detecting and classifying objects of interest. The UAS considered in the current thesis supports the classification tasks by providing additional visual information after the automatic classification methods that utilize the sensor information from UGS have identified an object of interest. The information generated by the UGS is supplemented with the visual information provided by the UAS and delivered to the human user for classification and situation identification. Having visual information associated with position and time information is crucial during this process.

3 Creating an autonomous UAS

The autonomous capability required for the UAS to perform as a component in an ISR SoS is built upon an existing UAV that is operator controlled and capable of simple waypoint navigation. The existing UAV is described later in current chapter, in the sub-chapter 3.1.4. The requirements of what must be added to the existing UAV are derived from the application scenario in chapter 1.2. The UAS must be capable of receiving the subscriptions from UGS and sending back the images. Furthermore UAS must be able to plan its mission autonomously and execute the mission and capture the images at required positions.

During the next sub-chapters (3.1 – 3.4) each of the components in an ISR SoS shall be described. The creation of UAS shall be explained in details in the rest of current chapter.

3.1 Components of the SoS in the Application Scenario

There are four different types of components in SoS used in current thesis: UAS, UGS, Net Server and Ground Control Station, see Figure [9].

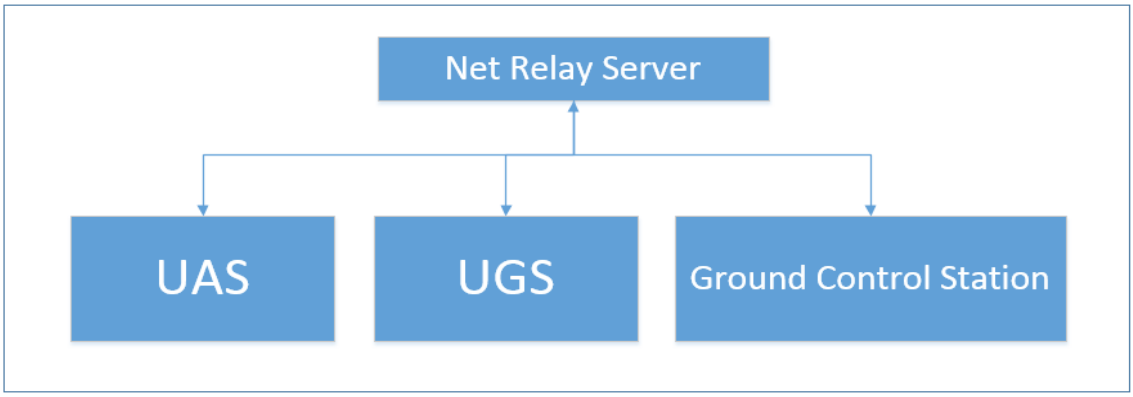

Figure 9: SOS architecture

The software implemented on UGS falls out of the scope of current thesis. The Net Relay Server is used as a temporary solution in the current configuration in order to implement a Client-Server model for communication between the UAS, UGS and Ground Control Station. The Net Relay Server is required as the data link (a 3G modem) used on the UAS enables only unidirectional data links from the UAS to the internet, the connection from the Internet is not enabled to the UAS. The implementation of software and hardware for UAS is explained below in chapters 3.2 and 3.3.

3.1.1 Net Relay Server

The Client-Server model is used because the requirement to transfer the images from UAS to the subscriber requires higher data rates than can be provided by sub-gigahertz modems which are typically utilized to establish data links from ground to a UAS. An ordinary network server is implemented to facilitate the communication between SoS components.

The proper implementation of communication architecture for such an SoS should be based on a dynamic ad-hoc local wireless network not reliant on internet and client server-model, but this topic falls out of the scope of the current thesis.

It must be also mentioned that the replacement of communication does not affect the general design of the SoS in question.

3.1.2 Ground control station

Ground control station consists of two parts, an open source ground control station software for monitoring the UAS telemetry information and mission progress, and a web application for displaying the images UAS is sending back to the subscriber.

The open source ground control station chosen for the case study is HappyKillmore [31], see Figure [10].

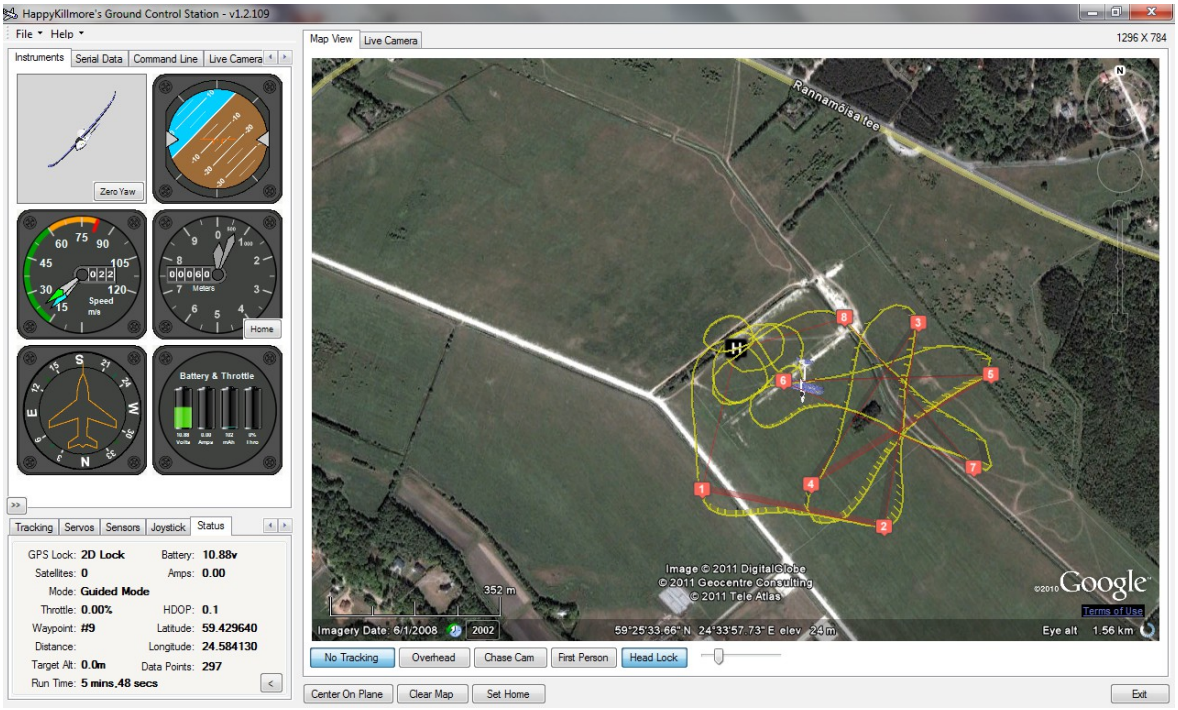

Figure 10: HappyKillmore Ground Control Station

It is an easy-to-use control station, providing the following features:

- supports MAVLink and NMEA protocols
- visual interface for UAS attitude
- Google Earth interface for displaying UAS location
- simple mission planning interface
- an interface for adjusting UAS autopilot parameters

3.1.3 The UGS network

The existing SoS is a tactical sensor network consisting of UGS (Unattended Ground Sensor) built according to SoS principles. Its purpose is to detect anomalous or illegal movement in the monitored area and to classify the types of the detected objects.

The UGS use various sensor modalities, including audio, seismic and magnetic field, being also able to perform classification in a collaborative manner. The interactions in the SoS are subscription based and mediated using ProWare [32]. The sensors are aware of their own position and among each other they exchange situational parameters in order to obtain and keep a local situational awareness picture.

When an object of interest is detected and classified, the UGS transmits an imaging subscription to the UAS. Once the image from the area has been received by the UGS, it is combined with the classification results and communicated to the information consumers who had requested information from the area.

3.1.4 UAS platform for ISR SoS

Vaatlusdroon UAS is a small tactical fixed wing aircraft, see Figure [11].

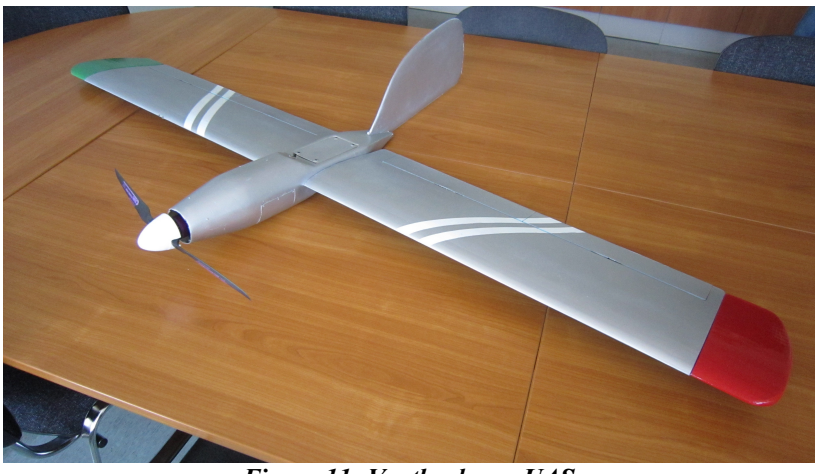

Figure 11: Vaatlusdroon UAS

Its main task is to provide images (intelligence data) about enemy activity in some given location. It weighs around 1.9kg, has a wingspan of 1.5m, and its average flight time is 40 minutes. The payload used for testing is a single COTS (commercial off the shelf) camera able to capture high resolution images.

The range and flight time is limited by battery power, and Vaatlusdroon keeps track of and is aware of this limit. During the image capturing session, the autopilot has built in algorithm for keeping the aircraft as steady as possible in current environmental conditions. Vaatlusdroon can operate in limited weather conditions - it can not tolerate high wind speeds or heavy precipitation, and in order to use its camera it needs daylight. It has no collision avoidance capability. Vaatlusdroon can be completely autonomous after it has been launched, as it is able to follow a preplanned trajectory uploaded from a ground station created by a human operator.

In order for the UAS to be able to perform as an autonomous system in an ISR SoS, the following abilities are to be added:

- ability to receive subscriptions for imaging service
- autonomous mission planning capability
- camera control capability
- capability of sending the images to the subscriber.
- if requested, to be able to provide situational parameters about its position, movement and flight time (also environmental data like wind speed and air pressure are available),

In future it is planned to add:

- an ability to decide also when a mission is not possible and thus the service can not be provided
- a capability to detect when it is in communication range with other components of SoS.

3.2 Description of the UAS on-board components

The components for UAV platform are chosen using the following criteria:

- low power consumption
- minimal dimensions
- low cost
- hardware compatibility and software support

Low power consumption is needed in order to maximize the flight time. The minimal dimensions are required as the payload space on board UAV is very limited. Low cost on the whole may be difficult to achieve as a UAS is a very complex system, but it is important that the each component is as cost-effective as possible. This is because during the development, testing and learning about the aircraft behaviour, it is impossible to fully avoid accidents or rough landings (take-off and landing are always most critical). This means that there must always be a certain amount of spare parts and materials ready, and this can easily double the developmental costs. Good hardware compatibility and wide software support is crucial when integrating the components into an integral whole. Most of these criteria cannot be satisfied simultaneously. Low power consumption or minimal dimensions versus low cost can be especially contradictory.

Following components are described in more detail in the thesis:

1. Platform, fuselage,
2. Servos and motors,
3. Batteries,
4. On-board remote control receiver (2,4Ghz),
5. Payload (camera),
6. Autopilot and sensors,
7. PilotControl module (Raspberry Pi),
8. 3G modem.

3.2.1 Fuselage

The fuselage is designed to fit into a rucksack and to be assembled very quickly, have good stability, and one of the requirements is also a low stall speed for capturing images. Also in case of poor landing or crashing, the wings will easily break off and thus avoid more dam-

age. Furthermore, the UAS is designed to be launched from hand. The material of the fuselage is a mixture of carbon and kevlar, making the fuselage strong and light. The design of the fuselage is also dictated by the planned electronics and the payload, as the centre of gravity is directly related to the position and aerodynamic design of the wings. This means that during fuselage design, the placement of all the internal components is planned ahead and cannot be changed much afterwards without compromises to flight time.

3.2.2 Servos and motors

UAS main electrical motor is of type A30-24M-UAV RHT. Its maximum power (15s) is 350W. With the current configuration of the battery set, the maximum power to the motor is 230W. UAS has four ailerons and a servo to control each one of the ailerons. All servos are of type MKS DS6100 Servo-10mm 9.5g 3.3Kg·cm. The servos for ailerons have 3.3 kg·cm torque and a speed of 0.12s/60°, which makes them suitable for electric glider and faster sport sloper steering (a sloper is an R/C plane without motor, thrown off the slope and using warm air currents).

3.2.3 Batteries

UAS battery pack consists of two 3000mAh lithium polymer batteries. Each battery has 3 cells and produces 11,1V. Batteries provide electricity to speed regulator, which in turn distributes the electricity to main motor and electronic equipment. The batteries are placed inside the wings.

3.2.4 On-board remote control receiver

Vaatlusdroon also has one on-board remote control receiver for manual override of control in case of emergencies. This is an 8 channel 2,4Ghz receiver for remote control. The channels are divided as follows: 2 channels are used for setting the mode/state of the UAS (AK-Pilot has four states, see chapter 3.2.6.2), 4 channels are used for ailerons, 1 channel is used for electric motor and 1 channel is available for payload control. The remote control option is mainly used while assisting the UAS take-off and landing, and also when the human operator intends to abort the mission.

3.2.5 Payload

Vaatlusdroon UAS is capable of carrying approximately 200g of payload. For testing purposes the Raspberry Pi camera module is used (OmniVision OV5647 image sensor [33]). It is a very lightweight camera designed for mainstream mobile phone market, has a 5 megapixel sensor, capable of making HD (720p/60) video, has following physical dimensions: 25mm x 20mm x 9mm and weighs 3g. As OV5647 camera is shipped together with Raspberry Pi, it is easily installed and can be fully controlled by software.

Other possibilities exist also. For example the fuselage for payload part is built cylindrical keeping in mind the possibility for one-axis gimbal. The design of one-axis gimbal is also supported by "Vaatlusdroon" UAS autopilot where also camera stabilization functionality has been implemented.

During the testing it is also planned to make a test with one of the sensor nodes as a payload on board UAS, interfaced to UGS, in order to test the ad hoc short range wireless communication possibilities.

3.2.6 Autopilot

The autopilot used for UAS is also developed by team "Vaatlusdroon" and is called by its programmers initials "AKPilot" (SW architecture has been designed by Andres Kangro and Andrus Tamboom, SW is programmed by Andres Kangro, hardware architecture is designed and assembled by Veljo Sinivee, Phd, author of current thesis participated in testing and debugging).

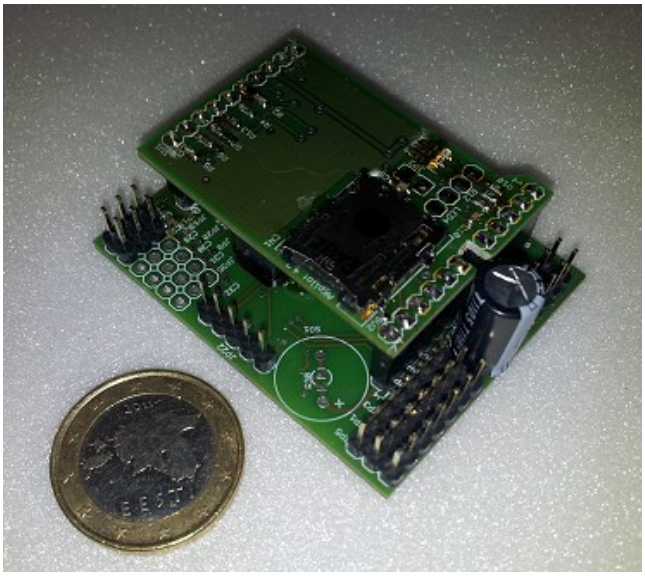

Figure 12: AKPilot

AKPilot, see Fig. 12, consists of two parts, the autopilot itself and the sensor board. The sensors are:

1. Acceleration sensors used on AKPilot are a ±3g, ±9g Three Axis Low-g Micromachined Accelerometers MMA7341LC.
2. There are two electrical gyroscopes on AKPilot of type: MEMS motion sensors, dual axis pitch and yaw ±30°/s analogue gyroscopes.
3. Air pressure sensors for altitude and air speed measurement.

The sensors board also contains a Secure Digital (SD) memory card reader.

The stabilization and flight control part is designed on a Texas Instrument Digital Signal Processing (DSP) microcontroller dsPIC30F6012 chipset with a 16bit 30MHz processor and 144KB of flash memory.

The autopilot uses Global Positioning System (GPS) for navigating from one position to another. An Ublox NEO-6M GPS Module is used. The GPS update rate is 5Hz.

3.2.6.1 AKPilot IO ports

A view of AKPilot interfaces is illustrated on Figure. [13]. The UAS is interfaced with the embedded system for high level control (Raspberry Pi) via the serial port.

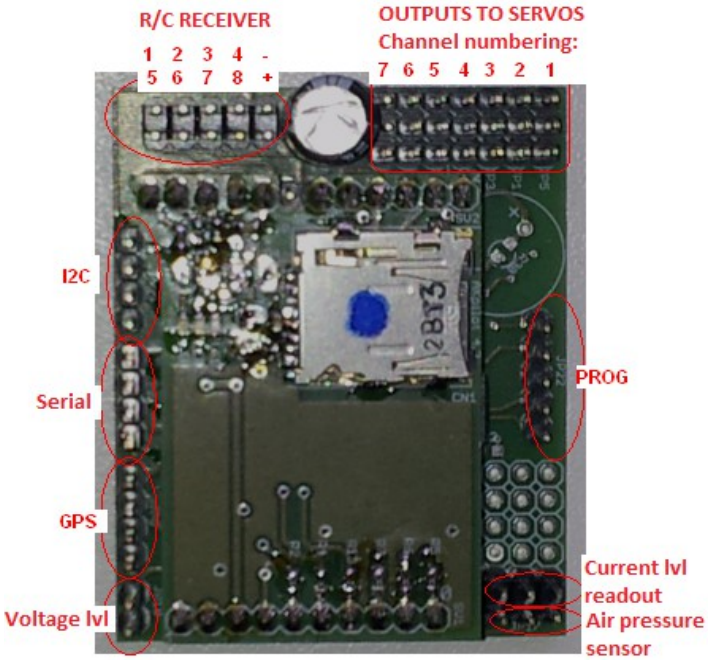

Figure 13: AKpilot connections

- R/C Receiver – a port for a radio control receiver.

- Outputs to servos – AKPilot supports up to 7 servos, current UAS has 4 ailerons.
- I2C – software not implemented.
- Serial – a serial connection to PilotControl module (telemetry and mission related messages). Pins from up to down: Ground, 5V, TX, RX.
- GPS – serial port for GPS messages (NMEA 0183 standard)
- Voltage lvl – a port for measuring battery voltage.
- Current lvl – a port for measuring current consumption
- Air pressure sensor – an analogue IO port for measuring the feedback from air pressure sensor.
- PROG – a port for programming interface.

3.2.6.2 The different states of AKPilot

AKPilot has four different states or modes, see Table [5]. After AKPilot is turned on, it will enter an initialization phase. During this phase the autopilot is initializing its GPS and other sensors (accelerometers, gyroscopes, pressure sensor). The position acquired during the initialization mode is set as HOME position. When only simulation is required then this mode can be skipped and HOME position is either loaded from a hardcoded memory location, or if there exists a previously uploaded mission on SD memory card, then the first waypoint is loaded as HOME position.

State/mode	definiton
Initialization	preflight Initialization of the sensors
Manual	human pilot has full command over UAS
Half autonomous	autopilot stabilizes and lets human pilot to steer the UAS
Fully autonomous	autopilot has full command over UAS

Table 5: States/modes of AKPilot

After initialization the UAV can be switched into any other of the abovementioned by sending a correct MAVLink command over an established communication link. In case AKPilot is switched into fully autonomous mode, it will search for a saved mission plan, and if it finds one, the AKPilot will start following the mission, otherwise it will start loitering over the HOME position at an altitude of 100m and begins waiting for commands.

In modes like “half autonomous” or “fully manual” mode the UAV does not accept commands from the groundstation. Half autonomous stabilizes the aircraft and human pilot

only needs to steer a general course, this is mainly used for landing and take-off procedures in more difficult conditions, when human assistance is required in order to avoid obstacles or landscape characteristics.

3.2.7 Embedded system for UAS autonomous control

In order to implement autonomous capabilities of solving tasks necessary for application scenario described in chapter 1.2, an additional embedded system needed to be installed on board UAS.

Two embedded systems were considered for PilotControl module: “Gumstix Overo” with “Pinto-TH” expansion board and “Raspberry Pi”. The comparisson of the two embedded system can be seen on Table [6]. Other parameters were not considered, because both systems had enough computing power and necessary input and output ports.

	Gumstix Overo on Pinto-TH	Raspberry Pi
Size (L × W × H)	76,2mm × 23.0mm × 9.0 mm	85.60mm × 56.0mm × 21.2 mm
Weight	12.2g	45g
Price	200 €	30 €
Power ratings	400mA (2,0W)	700mA (3,5W)

Table 6: Gumstix Overo on Pinto-TH expansion board vs Raspberry Pi

The first choice was Gumstix embedded system as its size, weight and maximum power consumption are considerably smaller than Raspberry Pi. But the Raspberry Pi price is lower, which is important when using it on a developmental aircraft; and more importantly - the software support of Raspberry Pi operating system is considerably better. Another aspect in the consideration was also available UAS payload size, which was 200g and both embedded systems fit well into it, so for those reasons the Raspberry Pi embedded system was chosen.

Raspberry Pi is a small single-board computer, see Figure [14], developed for educational purposes.

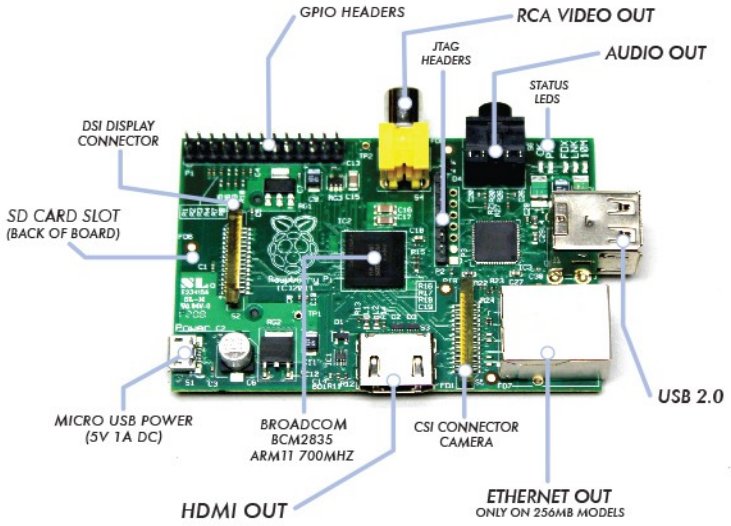

Figure 14: Raspberry Pi [34]

Some selected specifications relevant for PilotControl module:

- Broadcom BCM2835 System on a Chip (SoC)
- ARM1176JZF-S 700Mhz processor
- 512MB RAM
- Raspbian GNU/Linux 7 (wheezy)
- SD-Card for booting and persistent storage
- 2 USB ports
- 10/100 Ethernet controller
- 1 UART serial port (with 3,3V logic levels)

In the UAS configuration the Ethernet connection is used for debugging, first USB port is for 3G modem and serial port (among General Purpose Input Output (GPIO) ports) is connected to autopilot serial port.

3.2.8 Modem

The requirements for UAS communication capability with components of a tactical SoS are derived from the necessity to send high quality images to the subscribing UGS and to do it within the time constrains allocated for UAS task. The duration of the UAS task may be constrained by other tasks, UAS flight time, the communications availability etc., but it is highly preferred to keep the image upload times as short as possible. Each image is a JPEG file with approximate size of 3-5MB.

Thus in summary the following considerations are relevant:

- High data rate and long distance is required
- Sub GHz modems provide data rate around 200Kbps
- WiFi modems have ranges up to 100m.

As a result a COTS 3G modem is used for communication. It is both easy to implement and affordable. A Huawei E3131 which supports 3G and 3.5G (HSUPA, HSDPA), with speeds from 5.76Mbps up to 21.6Mbps is chosen. (In case of 3G or 3.5G is not available the 3G modem is expected to fall back to supported modes of GPRS or EDGE.)

3.3 Description of software implemented for UAS

This chapter gives an overview of the requirements and software components that were implemented. The vehicle based software (autopilot) itself has been left unchanged, only mission related software has been specified and developed.

3.3.1 The requirements for the software

The requirements for the software are derived from the mission scenario that UAS needs to perform as a component in SoS. The typical mission scenario is described in Figure [15].

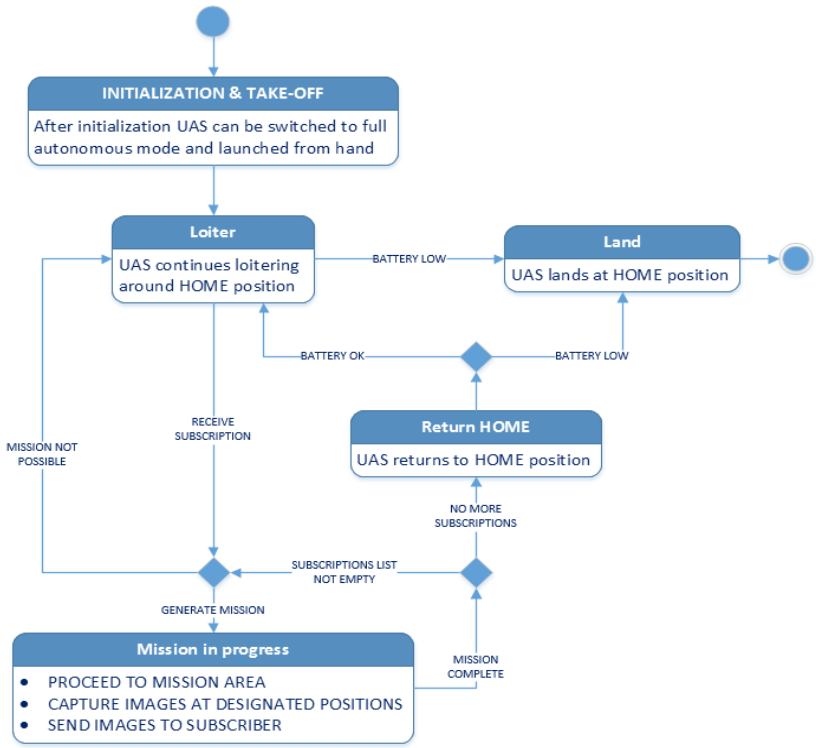

Figure 15: UAS mission scenario

After UAS has completed the initialization mode, it can be switched to full automatic mode and launched from hand. UAS then starts loitering above its initialization position called HOME. When UAS receives a subscription for an image capturing mission and if battery voltage has not dropped below pre-set threshold the UAS will proceed with the mission. If UAS receives more subscriptions, they will be handled by order of priority and order of arrival. When UAS has no more missions or decides that it is unable to complete more missions due to the low battery power (or weather conditions - UAS speed over ground depends on wind) it will return to HOME position and land.

In order to fill its role in SoS, the UAS must be able to:

1. receive a subscription for new target position (includes requirement for communication with other components in SoS),
2. plan a new mission according to a given target GPS position,
3. follow a pre-planned mission,
4. capture images on indicated positions,
5. upload images to the subscribing component.

These five capabilities that UAS must be capable to perform are also the high level requirements that are pursed in following paragraphs. The autonomous landing capability is out of context of current thesis.

3.3.2 UAS software architecture for the high level control module

The software architecture for the high level hierarchical structural design is used as depicted in Figure [16].

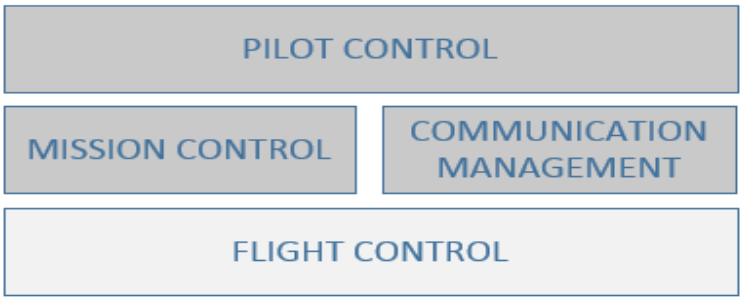

Figure 16: UAS software architecture

On the lowest level the autopilot manages the UAS stabilization and waypoint navigation. The second level in hierarchy is Mission Control, which according to the input from level above generates a mission plan and also handles the camera control during the mission. On the same level with the Mission Control a Communication Management logic is implemented. The Communications Management handles all the communication with both other members of the SoS and also relays the navigational telemetry between the internal components. The overall high level UAS control is managed by Pilot Control layer. It exercises the overall control of Mission Control and Communications Management layer and handles the subscriptions received from other members of SoS.

3.3.3 Pilot Control module

The Pilot Control module is a Python program for Raspberry Pi embedded system, written by Erki Suurjaak, an engineer at ProLab. The author of current thesis provided input on the requirements and operation logic for the module.

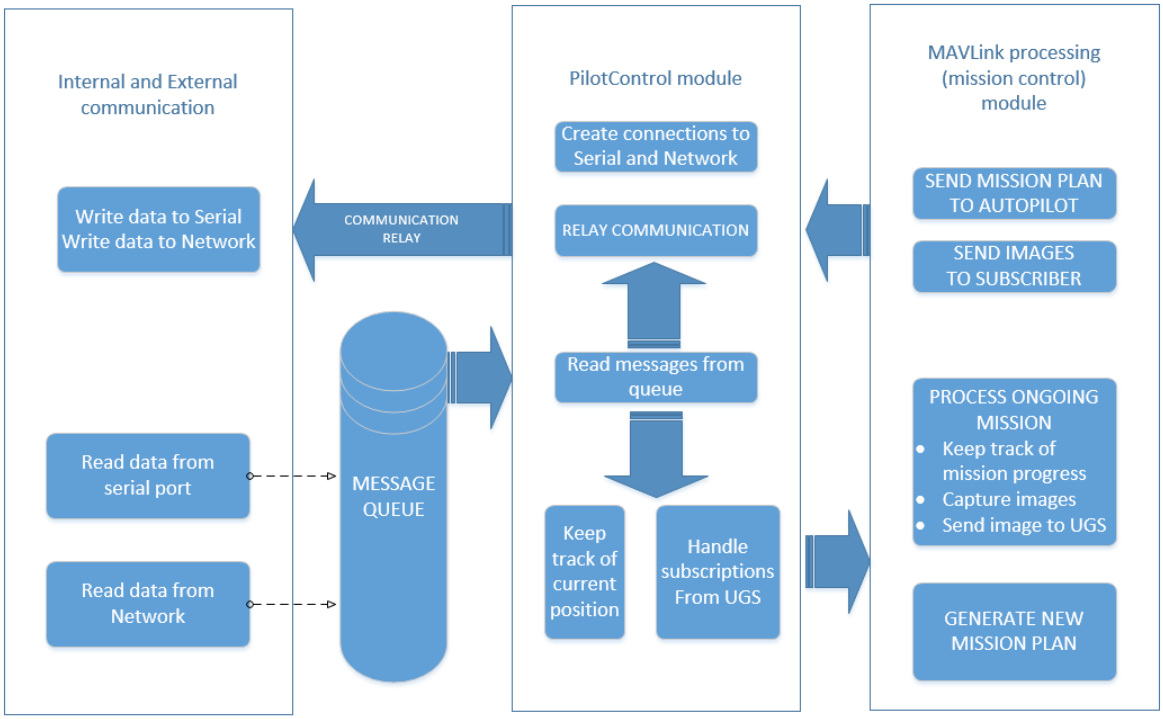

Figure 17: Pilot Control module

The Pilot Control module handles high level control of internal and external UAS communication, and mission control. The module creates both a serial connection to autopilot and a network connection to the remote network server, parses the relevant information from internal and external communication, keeps the vehicle state variables and acts accordingly.

The data read from network socket can either be a stream of MAVLink messages from ground control station or a subscription from UGS. The stream of MAVLink messages is relayed directly via the serial connection to the autopilot. In case Pilot Control module receives a subscription over network from a UGS, it will call the relevant software functions from Mission Control module and produce a new mission plan which then is communicated to autopilot.

The data read from serial connection is a stream of MAVLink messages from autopilot. It contains the UAS vehicle telemetry and system info, mission related requests and reports

and GPS data. The MAVLink messages are parsed by Pilot Control module and the obtained information is used according to its type. From GPS data the Pilot Control module extracts current position and memorizes it until it is updated. Mission request and report messages from autopilot are used by Pilot Control module for controlling the UAS mission. (starting new mission, aborting the mission, control of camera, etc.). The vehicle telemetry and system info contains various information about vehicle attitude, movement (including airspeed and heading) and also battery voltage and current.

All data red from serial port is also relayed to the ground control station.

The procedures for generating a mission, uploading a mission and controlling the camera are described more thoroughly below. Finally the Pilot Control module logs everything to a local file.

3.3.4 The Mission Control module

The software for managing mission planning and execution is implemented as a part of Pilot Control module. Its tasks upon receiving appropriate commands from Pilot Control module are generating a new mission plan, uploading it to the autopilot, execution of the plan and managing the camera control. The implementation of this software required thorough understanding of MAVLink protocol subset implemented on board AKPilot.

3.3.4.1 The mission plan generator

The mission plan generator is an independent software component, developed by the author of the current thesis and is executed by the Pilot Control module. The mission plan generator takes two arguments as its input. First argument is a set of GPS coordinates of the UAS current position and the second argument is the position of the target. The mission plan generator produces a list of MAVLink mission plan items where the first item is a current position and the rest are items which will command the UAS to fly over the given target and to make three images, first before the target, second above the target and third after the target. The distance between the image coordinates is hard coded constant. The reason for this is to adjust the overlap of images during the testing period.

It is also possible to generate various different mission plans, i.e. for flying several times over a given point with different angles, or to cover a given area with constantly spaced tracks in order to create a large mosaic image covering the whole area, but these options

are removed from current implementation as they are not needed for the scenario used in current thesis.

The navigational formulas for calculating distances, angles and new positions in WGS84 (World Geodetic System 1984) coordinate system make use of spherical geometry.

The final mission plan is composed of a list of mission items. Each mission item is a command that autopilot must execute. Currently only two types of mission items have been implemented on AKPilot's software. A waypoint command to navigate to a certain position according to GPS x, y and z coordinates and a command for camera triggering. The MAVLink command for waypoints is a "MAV_CMD_NAV_WAYPOINT" and its function as its name indicates is to navigate to a given waypoint. The MAVLink command for camera triggering is "MAV_CMD_DO_REPEAT_RELAY" and allows to cycle a relay on and off for a desired number of cycles with the desired period.

Mission plan generator produces currently only one type of a mission plan. A plan which consists of 6 mission items. An example of a mission is given on Figure [18].

MAVLink_waypoint_message (99,1,0,0,16,0,1,0,0,0,0,59.014099,22.609684,120)
MAVLink_waypoint_message (99,1,1,0,182,0,1,1,1,2,0,0,0,0)
MAVLink_waypoint_message (99,1,2,0,16,0,1,0,0,0,0,59.017963,22.609621,120)
MAVLink_waypoint_message (99,1,30,0,182,0,1,1,1,2,0,0,0,0)
MAVLink_waypoint_message (99,1,4,0,16,0,1,0,0,0,0,59.014664,22.612465,120)
MAVLink_waypoint_message (99,1,5,0,182,0,1,1,1,2,0,0,0,0)

Figure 18: MAVLink mission example

Where arguments have the following meaning:

1. UAS designation (System ID)
2. UAS designation (Component ID, not used)
3. Sequence number
4. The coordinate system of the mission (not used)
5. The scheduled action of the mission (MAVLink_Command_Item)
    1. 16 – an ordinary waypoint
    2. 182 – relay repeat
6. Is current (boolean variable, if true then indicates that this is the next waypoint)
7. Autocontinue (1 indicates that UAS will autocontinue to the next waypoint)

8. In case the scheduled action is relay repeat then: Relay number, otherwise not used
9. In case the scheduled action is relay repeat then: Cycle count, otherwise not used
10. In case the scheduled action is relay repeat then: Cycle time (seconds, decimal), otherwise not used
11. Not used
12. GPS latitude
13. GPS longitude
14. GPS altitude.

In given mission example there are six mission items, 3 waypoints and 3 commands (relay repeat command) to trigger the camera. Simply said, the UAS receives an imaging subscription and decides to proceed to the mission, it will plan a straight line to the target coordinates and then make 3 images, one before the target, one on the target and third after the target.

3.3.4.2 Uploading a mission plan to AKPilot

Procedure for uploading a mission items to AKPilot is the following:

1. Mission planner sends a mission items count to AKPilot
2. AKPilot requests each mission item
3. Mission planner sends each mission item as a response
4. When AKPilot has received the correct number (mission item count) of mission items, it will respond by sending a mission acknowledged message.

3.3.4.3 Execution of the mission plan

When a mission plan has been uploaded to autopilot memory, the mission can be executed by setting the AKPilot mode to “auto mission”. This can be achieved by using the following MAVLink message: “MAVLINK_MSG_ID_SET_MODE” with an integer argument indicating the mode number. Once the UAV is in “auto mission” mode, the UAV starts processing the mission items one by one according to the set sequence. In this mode it is also at any time possible to task the UAV to any waypoint that is contained in the current mission plan. This can be achieved by using MAVLink message: “MAVLINK_MSG_ID_WAYPOINT_SET_CURRENT”. This message takes one integer argument, which is the mission item sequence number. As the AKPilot executes one mis-

sion item at a time and does not consider the next mission item, the set waypoint current message commands the autopilot simply to abort currently ongoing task and to start executing the mission item given by the function argument. There can be a maximum of 255 waypoints in one mission. Setting the argument to any larger value causes AKPilot to abort mission and to return to HOME position.

3.3.4.4 UAV camera control

The camera triggering functionality is implemented within Mission Control module. While "PilotControl" software is relaying the AKPilot telemetry and mission related communication, it can parse out and memorise the mission commands that are designed for camera control and then AKPilot reports back that a certain mission item has been reached and next item has been set to "current", the Mission Control module will take action and command the camera capturing according to the previously memorised messages, see Figure [19].

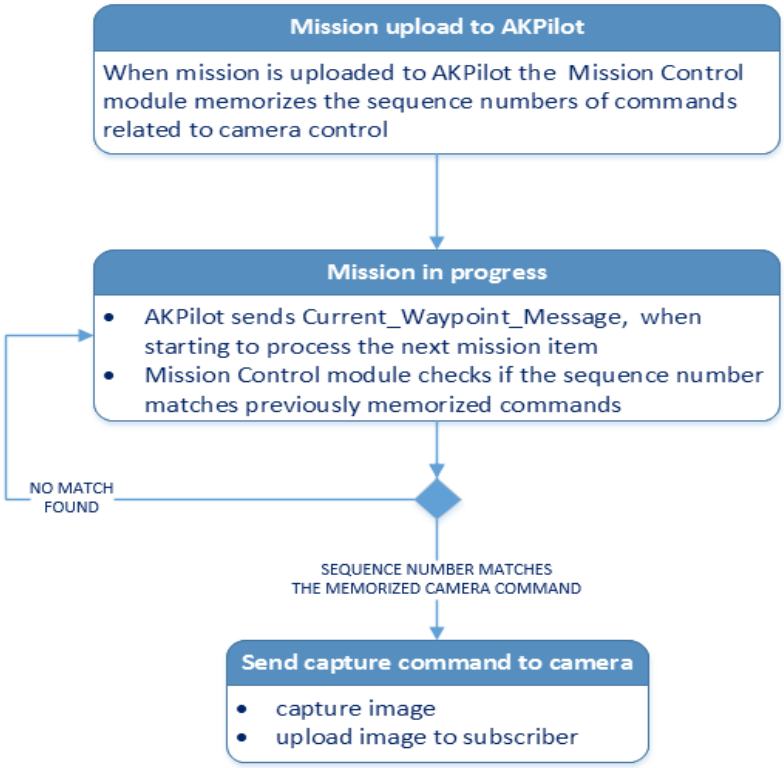

Figure 19: Camera control procedure

The images will be transferred to the requester of the images automatically right after the image has been captured.

3.3.5 Internal and external communication

The communication on board the UAS is managed by Pilot Control module (implemented on Raspberry Pi embedded system) and is divided into two main parts: an internal serial connection between the autopilot and the Raspberry Pi embedded system, and TCIP/IP connection using 3G modem to connect the UAS to Net Relay Server, see Fig. 20.

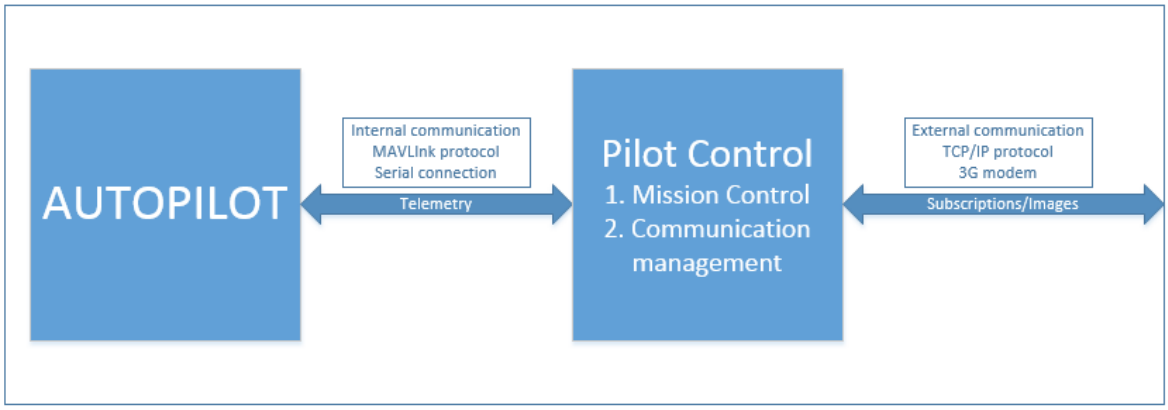

Figure 20: Internal and external communication model

The internal communication uses MAVLink protocol for flight related telemetry information. MAVLink protocol is explained in chapter 2.4.

The external communication uses TCP/IP communication protocol in order to exchange information between UAS, UGS and GCS. There are two main tasks for external communication. First task is to relay UAS telemetry information to UGS and GCS and the second task is to receive subscriptions from UGS and send mission results in form of images and related meta data for validation to the subscribing UGS. It is also the need to send high definition images that has resulted in a more complex Client-Server communication model for UAS external communication, as this allows the usage of high bandwidth connection to communicate captured images back to subscribing UGS reliably.

It must be emphasized that this approach is only required for the design-implementation-test purposes. For the final solution, which is out of the scope of current thesis, the client-server model shall be replaced with a more appropriate communication model designed specifically for loose collection of systems of systems.

4 Testing

Evaluation of the UAS platform was conducted in several stages, starting from tests in laboratory and finalized by tests with flying UAS in the field, communicating with the ground sensors.

4.1 Laboratory tests

Before commencing the flight tests it is necessary to test every component separately and whole system thoroughly on the ground.

4.1.1 Generating a simple plan, uploading it and running the simulation on it.

Test description:

Test was run on regular personal computers running Windows 7 operating system. During the test different positions were given to Mission Planner module as a function argument in order to validate it.

Lessons learned:

1. The UAS internal communication between autopilot and Pilot Control module is hidden from the human observer. The groundstation HMI must be updated manually in order to see the generated mission on groundstation map.
2. In case the communication is cut in the middle of mission upload then only some mission items are overwritten on AKPilot's memory and the resulting mission may consist of a mix of new mission waypoints and mission waypoints from previous mission. The AKPilot itself will not be aware of this and if ordered to execute, it will proceed with the mission as it is. Only indication of mission upload failure will be missing acknowledgement from AKPilot. **Fixed after software debugging**.

4.1.2 Testing the UAS-GCS communication over Net Relay server

Test description:

Net Relay server is run on regular personal computer running on Windows 7 operating system. Local WiFi network is used for communication medium. The software for UAS is run on another PC. The goal of the test is to validate the UAS communication with the HappyKillmore GCS. This is necessary for maintaining the overview of the UAS during

the flight tests.

Lessons learned:

1. During the software testing phase the components were run on PC with Windows operating system. It turned out that most of the programs for creating virtual serial ports readily downloadable from internet were unreliable. The serial data was buffered by the operating system and represented with a considerable delay. The problem was solved by switching to Linux operating system “Ubuntu”. Actually as the final solution must run on Linux operating system anyway, this was a quicker progression to the testing in final configuration.
2. The most interesting part for this test was to test the interactions between the on-board mission planning tool and HappyKillmore ground station. The mission planning tool software had to be updated to ignore the MAVLink packets sent by ground station and only listen to the packets with correct id (UAV system id). A better solution was later designed in Pilot Control module by routing the messages correctly. In updated software the mission planning tool only sees the messages coming from autopilot and is not aware of the groundstation. In case several missions is sent from both the groundstation and on-board mission planning tool, the missions are loaded to autopilot in order of arrival and the final mission is the one which is executed.

4.1.3 Testing software components on Gumstix and Raspberry Pi embedded systems

Test description:

Run different simulations using Gumstix and Raspberry Pi (RP) embedded system. In order to decide which embedded system to choose.

Lessons learned:

It turned out that RP operating system has much better software support and was easier get systems running. The more problematic hindrances with Gumstix embedded system were: serial interface support, USB drivers and software libraries for camera control. The RP embedded system has its own native camera and libraries for controlling it, also 3G modem was detected over USB without problems.

4.1.4 Testing the UAS in full configuration in the laboratory

Test description:

The test is ran in full configuration and in simulation mode. In Figure [21] illustrates the Raspberry Pi embedded system is installed in the UAS. The Raspberry Pi is powered by the UAS on board wiring and is connected to the autopilot via serial connections. The black cable on Figure [21] in upper right corner leads to 3G modem by which the UAS is connected to the internet.

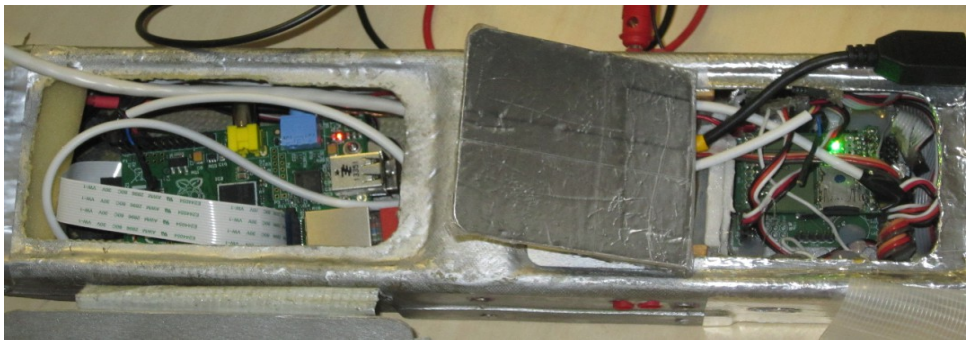

Figure 21: UAS testing in laboratory

Running the test scenario in simulation. The positions for subscriptions are simulated by a normal computer. There are three targets simulated on Figure [22].

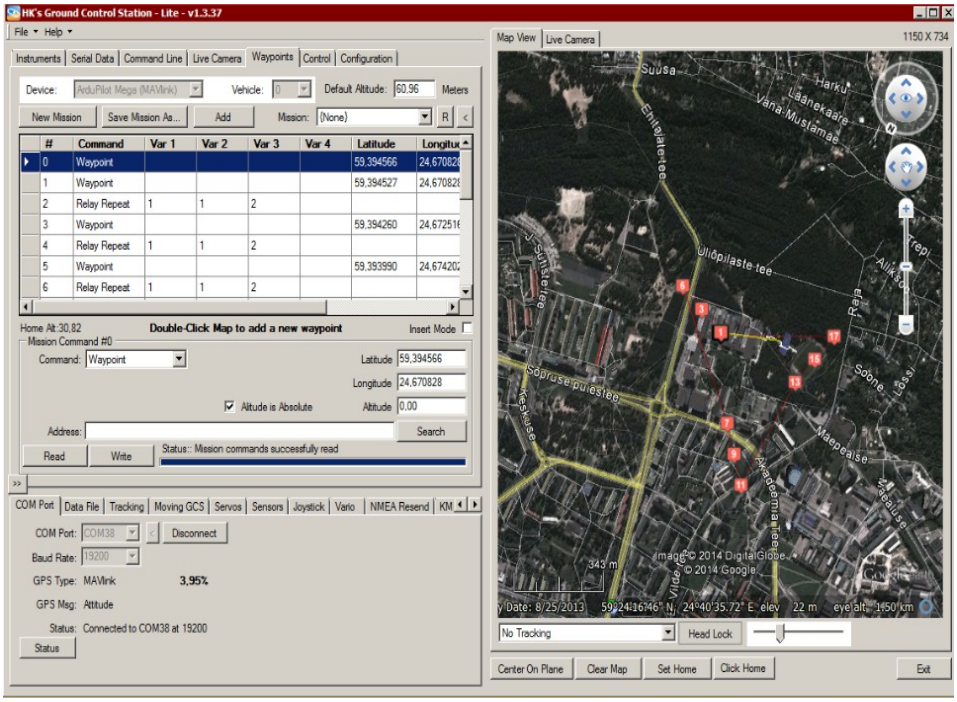

Figure 22: Generated mission plan downloaded from AKPilot

Lessons learned:

During the laboratory testing the uploading speed of images from UAS via the 3G modem was about 200-400Kb/s. For this reason the size of the captured images was regulated by limiting the quality of the captured images to 15% of their original captured quality. The result was that for a 1290x960 pixel and 160kB image it took 3-5 seconds to upload. The time interval during which the UAS is uploading the images back to the subscriber depends on local 3G connection speed and in current configuration the telemetry info is not sent while UAS is uploading images. It must be also mentioned that in final configuration the telemetry from UAS is not required.

4.2 Flight tests

4.2.1 Testing UAS without payload

Test description:

The purpose of the first flight tests is to validate the UAS capability to fly reliably and safely before the payload is installed and mission capabilities are tested. During the first flight tests the AKPilot is constantly measuring the requested attitude of the vehicle against the actual attitude and adjusting its regulator weight coefficients accordingly. UAS memorizes the coefficients every 5 minutes, so UAS has to be in half-autonomous or fully autonomous mode at least 5 minutes.

4.2.1.1 Lessons learned (10.05.2014; 16.05.2014; 18.05.2014):

During the first two flight test UAS behaved so badly that it was decided to not to let it fly by itself, but was landed already after 2-3 minutes flight in order to inspect the log files. First suspicion was that ailerons were not fully balanced, after rebalancing ailerons it turned out that this was not the problem. Another possible cause was found to be the propeller cone which caused vibration. From log files it was also discovered that GPS performance was getting worse when UAS tilted close to 60°. Thus for the third flight test the propeller cone was removed and allowable tilt was changed to 45°. After those changes the UAS started to behave better and the test was successful. This is the first UAS the Ublox NEO-6M GPS Module was used on. Further testing will show how GPS signal will behave under different conditions. It may well be that this model GPS is not usable for on-board UAS.

4.2.2 Testing UAS with payload

Test description:

The purpose of the test is to validate the chosen external communication model and the communication speed while UAS is flying. The Raspberry Pi (RP) is connected to the internet with the 3G modem and serial cable to the autopilot is disconnected. The goal is that after booting the RP, a start up script will create the connection to the internet and create a dedicated ssh tunnel that will enable the creation of the (Secure Shell) SSH connection from laptop computer during the field to test.

4.2.2.1 Lessons learned (23.05.2014):

A permanent connection was created and it was possible to send the camera commands to the RP during whole flight. A captured image from 100m hight with UAS camera can be seen on Figure [23].

Figure 23: Example of UAS camera quality from 100m hight

4.2.3 Testing UAS in full mission configuration

Test description:

Test according to the mission application scenario described in chapter 1.2. Sending the UGS subscription is simulated on a personal computer. Then UAS started loitering around home position the three subscriptions were sent in succession:

1. Target 1 (GPS: 59.428494° N – 24.581936° E)
2. Target 2 (GPS: 59.428753° N – 24.585783° E)
3. Target 3 (GPS: 59.428753° N – 24.585783° E)

The UAS had to plan the mission for each of the targets, execute the missions and send images back to the subscriber. (A mission contains 3 imaging positions with 50m spacing.)

4.2.3.1 Lessons learned (30.05.2014):

1. At first just before the launch it turned out that UAS was in simulation mode, this had never happened before on the field. It was noticed that ailerons were not reacting as they should and then HappyKillmore GS was checked it showed that UAS was already circling above us. Which it in reality of course was not. This was a software bug, the PilotControl module sent simulation command each time it started up, this had worked fine in the laboratory tests.
2. After the software had been fixed, the UAS was launched without any further trouble. It started the loitering pattern above, waiting for the subscriptions. The simulated subscriptions were sent and the UAS proceeded with the missions. The result is illustrated on Figure [24]. On the figure it can be seen that the UAS generated

Figure 24: The first testing of the application scenario

each mission for the next target in regard (the approach course chosen) to the last target, not to the last waypoint. This must be fixed in order to avoid the excessive loops then targets are very close to each other.

1. Due to yet unresolved software or rather communication problems the images for the first target were lost.
2. For the target 2 all three images were sent back to the subscriber but two first images did not capture the target due to the large UAS tilt. The third image was successful. See Figure [25].

Figure 25: Target 2, image 3

3. For the third target the first and the third images did not capture the target as the UAS tilt was too large. The second image, however did capture the target area, see Figure [26].

Figure 26: Target 3, image 2

In conclusion for the first test of the application scenario it can be said that a lot was learned. The UAS is able to receive subscriptions and is able to plan its missions. But the communication needs some more testing and also in case the UAS receives more than one subscription at a time, the mission generation needs improvements.

4.2.3.2 Lessons learned (31.05.2014):

This is the second outdoor testing in full configuration after software fixes. Two flights and several missions were run in order to test the communication and mission planning. During the testing a few times the connection to the UAS was lost, but the UAS managed to re-establish the connection to the internet. The flight trajectory of one of the scenarios can be seen on figure [27]. The green arrows on the figure indicate the positions were images were captured and sent back over the 3G connection to the subscriber. During this scenario the UAS was tasked to send 6 images for each target. It can also be seen the images before the target and after the target and after the target are captured too far and no overlap is gained.

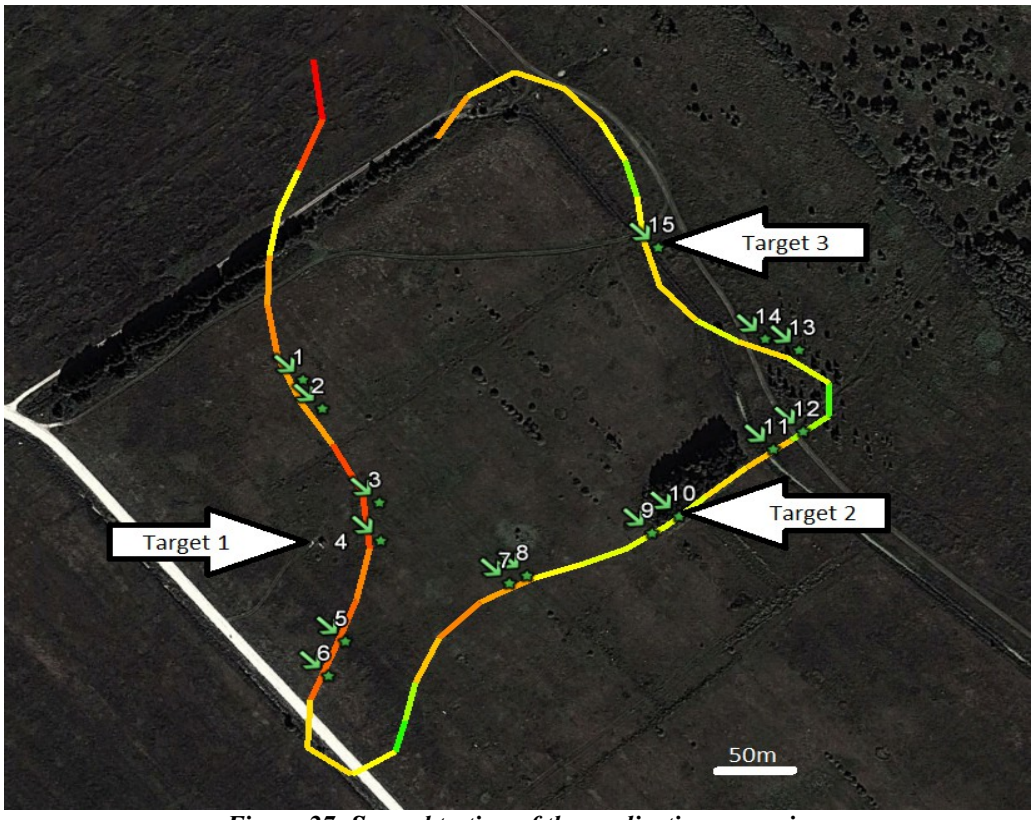

Figure 27: Second testing of the application scenario

The first target was not captured on image due to the UAS is still not allowed to start its

mission on its own. After the subscription was sent, it was checked that UAS had received it and generated the mission and the point where UAS was allowed to begin with the mission by operator on the ground, it was positioned so that it did have time to manoeuvre itself to fly over the first target.

The second target (the west corner of the grove) is approached and captured as required, See Figure [28].

Figure 28: Target 2 image 1

After reaching the third target there was some problems with the communication and only 3 images instead of 6 were sent back to the subscriber. Although the third image for target 3 captured it very well. All in all the second outdoor test was much more successful than the first, a lot more confidence was gained.

One conclusion from the tests so far can be drawn that 1 axis gimbal is definitely needed, as it can be seen that UAS can even in difficult conditions follow its plan with a quite a high confidence, but it is often impossible to hold the vehicle tilt needed to capture the targets required.

5 Conclusion

The thesis started by exploring the background for creating an autonomous UAS and integrating it as a component in an ISR SoS. Thesis investigated what can be considered as an UAS, what is the state of current legislation relevant for UAV and UAS, what is an SoS and how autonomy is measured in contemporary UAS. The contemporary UAS are mostly remotely operated, not autonomous and are not used as autonomous components in larger systems in a way described in current thesis. The current thesis shows that it is feasible to do so.

It is clear that an ISR SoS described in the application scenario of the case study benefits greatly from having a UAS as a component. The autonomous UAS developed within the scope of the thesis features all the necessary characteristics identified in the first half of the current thesis. The UAS - UGS collaboration model is based on choice and loose connectivity, fitting well under the SoS characteristics of belonging and connectivity. The requirement for diversity is also satisfied by comparing the UAS to any of the UGS nodes, they are very different both by design, behaviour and services they can provide. Quite likely the newly created SoS will feature emergent behaviour, but this is subject to further studies.

The author concludes that the most important characteristic for a system to be introduced into an existing SoS is autonomy. The UAS has its constraints, but as long as the conditions allow it to operate, it will be able to do so as a component in a System of Systems.

The thesis also gives an overview of hardware of the existing UAV system and what was needed to be added or replaced. Within the scope of the thesis a new UAV airframe was developed and implemented, hardware was selected, software designed and implemented. The resulting UAS is able to communicate with the ground sensors systems, get data requests, plan the missions, take pictures and send them back to the subscriber.

The UAV taken as a basis for building an autonomous UAS was custom built separately of current thesis. The author provided input in design aspects regarding the new computing system that had to be added and payload, and helped with minor details. The development tracks of the UAS were conducted concurrently, the construction of the new airframe for the UAS and authors work on the hardware design and requirements engineering for the

UAS. The requirements of the UAS were derived from the application scenario described in chapter 1.2 and were in short following:

1. UAS must be able to receive data subscriptions from UGS,
2. UAS must be able to plan its mission,
3. UAS must be able to execute its missions,
4. UAS must be able to send images back to the subscriber.

Also a single axis gimbal for controlling the angle of the payload was planned for the UAS and initial design was completed, but due to the time constraints this functionality was not implemented within the context of this thesis.

The software (written in the Python programming language) was initially developed and tested on a desktop computer and then moved to the embedded platform. The set-up of the environment for the testing and setting up the ports/interfaces for the data communication was an especially challenging task within the development cycle.

The challenges encountered in the final outdoor tests showed what could have been predicted: such a system is greatly affected by the physical environment and most of these aspects cannot be easily simulated or predicted in a laboratory environment. For example then the UAS is loitering above the starting position or executing its mission it actually flies very fast (the average speed is 80-90km/h), which may have an impact on the 3G modem communication. UAS also makes random turns while it is loitering (disappearing sometimes for several seconds behind the tree line) which makes it hard to follow and, while it is just a psychological effect - makes the observer think “is it still coming back” and adds to the already tense atmosphere of the testing. The outdoor testing sessions are also challenging as they lack the comfortable environment of the laboratory – the tools and the equipment.

The completion of the thesis is just the start for the following work – additional field tests with the UAS platform are needed to gain increase confidence in the platform and to develop the system further. Live tests with operational smart network of ground sensor systems can be started once the ground sensor systems are able to offer all the required functionality (i.e. classification and tracking of mobile objects of interest). There are also several plans for upgrades of the UAS, the 1 axis gimbal is to be added, the mission planning decision process needs improving so that UAS can work with several ground based ISR systems simultaneously.

The UAS solution created within the scope of the thesis can be effectively used for a perimeter control in an environment of asymmetric threat in collaboration with UGS. The UGS detect and classify the threat and the UAS provides visual information in order to improve the situational awareness. The developed solution can be also applied in the context of Estonian national defence.

Resümee

Magistritöö sissejuhatuse käigus selgitatakse eesmärke ja soovitud tulemust näidisstsenaariumi näol. Nimelt on tänapäevase sõjapidamise üks suurimaid väljakutseid asümmeetrilise ohu vähendamine. Käesoleva töö kontekstis loodud lahendus, mis kombineerib ja täiendab olemasolevaid süsteeme, on kasutatav selle väljakutse lahendamiseks. Käesolev töö seob mehitamata autonoomse lennuki autonoomsete maapealsete sensorsüsteemide süsteemiga.

Magistritöö toob ülevaate kirjeldatud ülesande lahendamisega seotud mõistetest ning tehnoloogiatest. Üks tähtis eristamine on vahe UAV ja UAS vahel. UAV on motoriseeritud, juhitud, mehitamata õhu-sõiduk, UAS on keerukam ja autonoomne õhusõiduk, mida võib pidada omaette süsteemiks. Antud töös on UAS lennuk, mis on võimeline autonoomselt lendama ning oma tegevust planeerima vastavalt süsteemist saabuvatele andmete soovile.

Töös tuuakse ka lühike ülevaade mehitamata lennukite käitamise regulatsioonist, mis on hetkel kehtestamisel Eesti seadusandluses. Mehitamata sõidukitele kategooriate ja reeglite seadmine on ennekõike ohutuse tagamiseks. Näiteks kõige madalama kategooria mehitamata lennukiga, milleks on lennuk stardimassiga alla 5 kg, tohiks lennata 1km sõõri sees ja maksimaalselt 150m kõrgusel, mis on allpool regulaarse õhuliikluse jaoks kehtestatud piire. Lennates sellest välja või või raskema stardimassiga kehtivad lennukile juba hoopis rangemad reeglid. Teisalt on käesoleva töö eesmärk loodud süsteemi kasutus militaarvaldkonnas. Militaarmaailmas pole mehitamata õhu-sõidukeid juriidilise külje pealt vaja nii rangelt reguleerida, kuna militaarvaldkonnas on vastutus juba tagatud süsteemi organisatsioonilisest ülesehitusest tulenevalt. Samuti on militaarvaldkonnas süsteemide testimise ning kasutamise harjutamiseks katsepolügoonid kus tavainimesi ei ohustata. Veel toob autor välja olulise nüansi täiesti autonoomsete relvasüsteemide loomise valdkonnast. Nimelt vastutab relvasüsteeme loov iga riik ise selle eest, et süsteemid vastaksid rahvusva-

helistele nõuetele.

Töö annab ülevaate ka NATOs kehtivates UAV-de kommunikatsiooni standarditest. NATO standardite eesmärk on reguleerida protseduuride ja erinevat tehnika standardiseerimist üle kogu NATO riikide, selleks et tagada parem sõjaline ja administratiivne koostöö riikide vahel.

Kuna lõputöö peamine eesmärk on integreerida mehitamata lennuk Süsteemide Süsteemi, siis annab töö ka ülevaate sellest, mis on Süsteemide Süsteem, autonoomsus ja kuidas mõõdetakse autonoomsust mehitamata sõidukitel, eesmärgiga teada saada mida on selleks vaja, et olemasoleva Süsteemide Süsteemiga liita veel üks olemasolev süsteem. Vaadatakse ka Süsteemide Süsteemide loomisega seotud aspekte.

Süsteemide Süsteem on süsteem, mis on suhteliselt suur ja keerukas süsteem, mille komponendid on eelnevalt olemasolevad ning mis on üldiselt füüsiliselt hajutatud ning dünaamiliselt arenev.

Töö teises pooles kirjeldatakse põhjalikult kuidas ja millised täiendused oli vaja viia sisse selleks, et töös kasutatud mehitamata lennuk muuta autonoomseks Süsteemide Süsteemi osaks. Alustades nõuetest luuakse UAS arhitektuurne disain ning valitakse selle realiseerimiseks vajalikud riistvaralised ning tarkvaralised komponendid, selgitatakse riistvaralisi valikud ja uuendusi, seejärel antakse ülevaade lisatud sardsüsteemi tarkvarast ning lõpuks kirjeldatakse läbiviidud teste. Riistvara kirjelduse juures tuuakse välja kirjeldus kõigi lennuki riistvaraliste komponentide kohta. Uued lisatud komponendid on: sardsüsteem (Raspberry Pi), 3G modem ning sardsüsteemi poolt täpselt juhitav kaamera. Sardsüsteemiks valiti Raspberry Pi, kuna see on mõistliku hinnaga (võrreldes teiste sarnaste platvormidega) ning ning lisaks on Raspberry Pi tarkvaraline toetus parem kui teistel sarnastel platvormidel.

Nõuded tarkvara jaoks identifitseeriti lähtudes stsenaariumist tuletatud alam-ülesannetest, mis on järgmised:

- tellimuste vastuvõtmine maapeal asuvatelt sensorsüsteemidelt,
- missiooni planeerimine vastavalt tellimustele,
- missiooni laadimine autopilooti,
- kaamera juhtimine vastavalt missiooni käigule,
- ning piltide saatmine tellijale.

Tarkvara üldstruktuur on disainitud Proaktiivtehnoloogiate laboratooriumi inseneri Erki

Suurjaak poolt. Töö autor analüüsis kuidas toimub maajaama ning autopiloodi vaheline kommunikatsioon võimaldamaks missioonide laadimist autopilooti UAS-l paiknevast sard-arvutist. Lähtuvalt kogutud infost lõi autor missiooni genereerimise tarkvara ning missiooni autopilooti laadimise mooduli, samuti visuaalse sensori käivituse funktsionaal-suse. Autor koordineeris ja viis läbi UAS platvormi integratsiooniteste mille osaks oli muu-hulgas tarkvaramoodulite testimine ning tarkvara silumine erinevates konfiguratsioonides.

Käesoleva magistritöö valmimine on ainult edasise töö algus. Jätkata on vaja nii välitestidega, et jätkuvalt suurendada usaldust UAS vastu, kui ka seda edasi arendada. Kui sensorsüsteemide arendus jõuab nii kaugele, et sensorsüsteemid on võimelised pakkuma vajaminevat funktsionaalsust (huvipakkuvate mobiilsete objektide klassifitseerimine ning jälgimine) saab ka hakata läbi viima UAS väliteste koos sensorvõrgustiku süsteemiga. Ka UAS-i edasiarenduseks on rida plaane, näiteks on plaanis lisada UAS-le 1 teljeline kaamera stabiliseerija, samuti missiooni planeerimise otsustusprotsess vajab edasiarendust et UAS suudaks koos töötada mitme maapeal asuva ISR süsteemiga koos samaaegselt.

Lõppkokkuvõttes võib öelda, et käesoleva töö käigus loodud UAS lahendus on kindlasti kasutatav koostöös maapeal asuvate sensorsüsteemidega perimeetri kontrolli tarbeks asüm-meetrilise ohu keskkonnas. Maapeal asuvad sensorid tuvastavad ohu, klassifitseerivad selle ning UAS annab visuaalse ülevaate situatsiooniteadlikkuse suurendamiseks. Välja töötatud lahendust saab kasutada ka Eesti riigikaitse kontekstis.

Appendix I: Implemented MAVLink messages in Pilot Control software

MAVLink_attitude_message	Contains vehicle tilt, yaw and pitch.
MAVLink_gps_raw_message	Contains raw GPS information.
MAVLink_heartbeat_message	Periodic message to indicate originator availability.
MAVLink_set_mode_message	Sets vehicle mode (autonomous, manual, etc.).
MAVLink_sys_status_message	Sensors status message.
MAVLink_vfr_hud_message	Contains information of heading, speed, altitude, etc.
MAVLink_waypoint_ack_message	Acknowledge sent by UAS then mission has been received correctly.
MAVLink_waypoint_count_message	Sends waypoint count N, receiver starts requesting for N waypoints.
MAVLink_waypoint_current_message	Sets currently followed waypoint if originator is GCS or informs about that vehicle has set new waypoint as target if originator is UAS.
MAVLink_waypoint_message	Contains waypoint data.
MAVLink_waypoint_reached_message	Informs that waypoint is reached.
MAVLink_waypoint_request_message	Used for requesting next waypoint in list when uploading mission.

References

[1] J. Preden, L. Mõtus, J. Linas, R. Pahtma, R. Savimaa, M. Meriste, S. Astapov, „Improvised Explosive Devices in Asymmetric Conflicts“, „Multisource Data Fusion for Providing Situational Information“.

[2] Norman S. Sakamoto, UAV development and history at Northrop Grumman Corporation Ryan Aeronautical Center, URL: http://www.nps.edu/Academics/Institutes/Meyer/docs/August%2026%202004%20History%20of%20UAVs.pdf [accessed 30.04.2014]. p. 2.

[3] MEHITAMATA ÕHUSÕIDUKITE TSIVIILOTSTARVEL KÄITAMIE EESTI ÕHURUUMIS, Eelnõu kavand. URL: http://www.infopank.ee/UAS_reeglistik_2013_02_14.docx [accessed 29.05.2014].

[4] Proposals for change in [3]. URL: http://www.infopank.ee/Ettepanekud_eelnou_kavandisse_ver1.2.1.docx [accessed 29.05.2014].

[5] Richard M. O'Meara, Intersection: "The Rules of War and the Use of Unarmed, Remotely Operated, and Autonomous Robotics Systems, Platforms and Weapons…Some Cautions". URL: http://robots.law.miami.edu/wp-content/uploads/2012/01/Omeara-INTERSECTION.pdf [accessed 26.05.2014], p. 9.

[6] Protocol Additional to the Geneva Conventions of 12 August 1949, and relating to the Protection of Victims of International Armed Conflicts, 8 June 1977 Article 36 of 1977.

[7] Wikipedia. STANAG – Wikipedia, the free encyclopedia, 2014. URL: http://en.wikipedia.org/wiki/STANAG [accessed 24.04.14]

[8] T. Bandzul, STANAG 4586 –Enabling Interoperability URL: http://www.thesciencedude.com/projects/RESEARCH/MASSystem/References/Bandzul.pdf [Accessed 26.05.2014], p. 42.

[9] STANAG 4586 (EDITION 3) – STANDARD INTERFACES OF UAV CONTROL SYSTEM (UCS) FOR NATU UAV INTEROPERABILITY, URL: http://nsa.nato.int/nsa/zPublic/stanags/current/4586eed03.pdf [accessed 25.05.2014], p. VIII.

[10] AEDP-2 (Edition 1) – NATO Intelligence, Surveillance, and Reconnaissance (ISR) Interoperability Architecture (NIIA), Volume 1: Architecture Description. URL: http://www.nato.int/structur/ac/224/standard/AEDP2/AEDP2_Documents/AEDP-02v1.pdf [accessed 25.05.2014], p. 14.

[11] Wikipedia. MAVLink – Wikipedia, the free encyclopedia, 2014. URL: http://en.wikipedia.org/wiki/MAVLink [accessed 21.02.2014].

[12] QgroundControl. MAVLink packet structure. URL: http://qgroundcontrol.org/mavlink/start [accessed 24.05.2014].

[13] H. Huang, Autonomy Levels for Unmanned Systems. URL: http://www.nist.gov/el/isd/ks/upload/ALFUS-BG.pdf [accessed 21.03.2014], p. 6.

[14] H. Huang, Toward a Generic Model for Autonomy Levels for Unmanned Systems (ALFUS), URL: http://www.dtic.mil/dtic/tr/fulltext/u2/a515323.pdf [accessed 21.03.2014], p. 4.

[15] W. C. Marra, S. K. McNeil, "Understanding "The Loop": Humans and the Next Drone Generations", Issues in Governance Studies, August 2012.

[16] Endsley, M. R. Designing for Situation Awareness in Complex Systems, URL: http://209.238.175.8/Papers/pdf/SA%20design.pdf [accessed 03.03.2014], p. 4.

[17] Preden, J. Situation Awareness for Networked Systems –IEEE First International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), Miami Beach, Florida, February 22-24 2011, 2011, 125. [Onlne] IEEE Xplore (30.04.2014), p. 125.

[18] Raja Parasuraman, Thomas B. Sheridan, "A Model for Types and Levels of Human Interaction with Automation", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS – PART A: SYSTEMS AND HUMANS, VOL 30, NO 3, MAY 2000, p. 18.

[19] Wikipedia. OODA – Wikipedia, the free encyclopedia URL: http://en.wikipedia.org/wiki/OODA_loop [accessed 25.04.2014].

[20] E. Sholes, "Evolution of a UAV Autonomy Classification Taxonomy", Aerospace converence, 2007, IEEE, page 1 and p. 3.

[21] AUTONOMY LEVELS FOR UNMANNED SYSTEMS (ALFUS) FRAMEWORK, (Dec. 2007). URL: http://www.nist.gov/customcf/get_pdf.cfm?pub_id=823618 [accessed at 06.04.2014], p. 9.

[22] E. A. Lee and S. A. Seshia, Introduction to Embedded Systems - A Cyber-Physical Systems Approach, LeeSeshia.org, 2011, p. 1-5.

[23] Steven Eppinger, "A systems engineering view of boeings 787 dreamliner", URL: http://mitsloanexperts.mit.edu/a-systems-engineering-view-of-boeings-787-dreamliner-steve-eppinger/ [accessed 18.03.2014].

[24] Stan Franklin and Art Graesser (1996); Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents; Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996

[25] J. Boardman and B. Sauser "System of systems—The meaning of of", *Proc. IEEE/SMC Int. Conf. Syst. Syst. Eng.*, pp.118 -123 2006

[26] W. N. Felder "Interactions Among Components in Complex Systems", Stevens Institute of Technology, Hoboken,

NJ 07030, p. 3.
[27] Robert Valdes, “How the Predator UAV Works”, HOW STUFF WORKS (Apr. 1, 2004), URL: http://science.howstuffworks.com/predator.htm [accessed 5.04.2014].
[28] Bry, A. Massachusetts Inst. of Technol., Cambridge, MA, USA, State estimation for aggressive flight in GPS-denied environments using onboard sensing.
[29] A. Xu and G. Dudek: A Vision-Based Boundary Following Framework for Aerial Vehicles, in *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '10)*, p. 81--86, Taipei, Taiwan, October 2010. http://www.cim.mcgill.ca/~anqixu/.
[30] Joint Publication JP 1-02, Department of Defence Dictionary of Military and Associated Terms, URL: http://www.dtic.mil/doctrine/new_pubs/jp1_02.pdf [accessed 26.05.2014], p. 73.
[31] HappyKillmore Ground Control Station. URL: http://code.google.com/p/happykillmore-gcs/ [accessed 26.05.2014].
[32] J. Preden, L. Mõtus, J. Linas, R. Pahtma, R. Savimaa, M. Meriste, S. Astapov, Improvised Explosive Devices in Asymetric Conflicts, Multisource Data Fusion for Providing Situational Information, (2014).
[33] OmniVision OV5647 image sensor. URL: http://www.ovt.com/products/sensor.php?id=66 [accessed 04.06.2014].
[34] Raspberry Pi, Model B, URL: http://www.pcmag.com/article2/0,2817,2407058,00.asp [accessed 04.06.2014].