NTNU – Trondheim
Norwegian University of
Science and Technology

Utilizing Unmanned Aerial Systems in Search and Rescue Operations

Gaute Gamnes

Master of Science in Cybernetics and Robotics
Submission date: June 2014
Supervisor: Amund Skavhaug, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Utilizing Unmanned Aerial Systems in Search and Rescue Operations

Gaute Gamnes

June 9th, 2014

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Supervisor: Associate Professor Amund Skavhaug

Task Description

Title:	Utilizing Unmanned Aerial Systems in Search and Rescue Operations
Task given:	January 6st, 2014
Deadline:	June 9th, 2014
Supervisor:	Associate Professor Amund Skavhaug, Dr.ing., Department of Engineering Cybernetics, NTNU
Student:	Gaute Gamnes

The Department of Engineering Cybernetics, NTNU, wishes to develop an unmanned aerial system that can be utilized during search and rescue operations. The main objective of this Master Thesis is to prove the feasibility of this concept. A possible solution shall be proposed and a prototype for this implemented.

The candidate shall amongst other:

1. Perform a study of necessary background theory and previous works
2. Propose a system solution
3. Implement a prototype of the proposed system
4. Test the developed system according to expected use
5. Evaluate the results and suggest possible further work

Oppgavebeskrivelse

Tittel:	Bruk av Ubemannede Luftfartøysystemer i Søk og Redningsaksjoner
Oppgave gitt:	6. januar 2014
Oppgave frist:	9. juni 2014
Veileder:	Førsteamanuensis Amund Skavhaug, Dr.ing.,
	Institutt for teknisk kybernetikk, NTNU
Student:	Gaute Gamnes

Institutt for teknisk kybernetikk ved NTNU ønsker å utvikle et ubemannet luftfartøysystem for bruk under søk- og redningsaksjoner. Hovedoppgaven for denne masteren er å studere og innsamle informasjon om hvordan et slikt system kan utvikles. En løsning skal presenteres og en prototype implementeres for å hjelpe framtidig forskning på dette feltet.

Oppgavene i denne masteren vil bestå av å:

1. Sette seg inn i nødvendig bakgrunnsteori og tidligere arbeider
2. Foreslå en systemløsning
3. Implementere en prototype av det foreslåtte systemet
4. Teste det utviklede systemet med tanke på forventet bruk
5. Evaluer resultatene og foreslå mulig videre arbeide

Preface

This Master Thesis is done as part of the study program Master of Science in Engineering Cybernetics at the Norwegian University of Science and Technology. The Master Thesis was written in the time period from January to June 2014, and is a continuation of a previous project performed at the Norwegian University of Science and Technology during the fall semester of 2013.

The reason for the topic of this Master Thesis was somewhat based on enthusiasm regarding RC helicopters and planes. Also the idea of providing the search and rescue organizations with a tool that might help save lives is a huge motivation factor.

Trondheim, June 9th, 2014

Gaute Gamnes

Acknowledgement

First, I would like to thank my supervisor, Associate Professor Amund Skavhaug, for all the help and input during the work on this Master Thesis. His guidance and suggestions during were of good use when writing this report.

Thanks to Professor Tor Arne Johansen, for letting me use some equipment in order to make the testing possible.

In addition, I would like to thank all the people who have helped me with testing and organization of tests. Special thanks to Ann Charlott Ringfoss, and Audun Sunde for helping me fly the plane during tests.

Summary and Conclusion

The main objective of this Master Thesis has been to prove the feasibility of utilizing an unmanned aerial system during search and rescue operations, and also to propose and develop a solution for such a system. This concept is proven feasible throughout this report, and a possible system solution for this concept is presented. A prototype system is developed using many already developed devices and the interfaces between these devices are developed. The unmanned aerial system is developed to function with a high degree of autonomy because search and rescue personnel are often occupied with other assignments and do not necessarily have the time to operate such a system. The use of autonomous systems also reduces the need for experienced personnel in order for the system to be utilized; hence making it more likely that the system will be used.

The need and desire to use unmanned aerial systems during search and rescue operations has been established through an extensive literature study and this report has been written to help the future development of such a system. Many unmanned aerial systems are in use in today's society for many different applications. However, not many are specifically intended for use during search and rescue operations, and consequently the development of such a specialized system is desired. The advances made in unmanned system technology has recently made such systems more complete, and thus considered for being utilized in more applications than before. A well-known statement is that unmanned system are usually considered for use in operations that are dull, dirty, or dangerous for humans to perform. During search and rescue operations, the search crew often have to cover massive areas in order to locate the missing persons and this work could be tiresome. Having a system that is simple to set up and use, which can do the same search as the SAR crew only faster and more efficiently, will be advantageous for the searchers and thus also the missing persons. In a search and rescue scenario, the more time that goes by, the more dangerous the situation for the missing persons could become, and thus time saved, might mean lives saved. Having a system, which helps save time during search and rescue operations, is therefore much desired.

In order to prove the feasibility of the concept of utilizing an unmanned aerial system during search and rescue operations, a literature study was conducted to provide information on other solutions and projects and the result of this study is presented in this report. The report also summarizes how a search and rescue operation could be performed, and this is done in order to provide information that can be used when designing a search and rescue unmanned aerial system. A solution for a system design, and a prototype based on this solution is presented and implemented. The system is also tested and the test results are presented and discussed in the report. The prototype unmanned aerial system includes a commercial autopilot system with its associating components needed for it to function, which is mounted on a plane that is a flying wing. The system also has the option of having a camera detection system mounted on board the plane such that it can detect humans from the air. The detection system was developed such that it can be used on board an unmanned aerial vehicle and consists of a camera and an embedded computer, which can run a detection application. This detection system is also tested and evaluated in this report. The report is written to provide other researchers with a road map to follow during the future development of the concept investigated in this Thesis, and thus getting closer to realizing an operational search and rescue unmanned aerial system.

The developed prototype unmanned aerial system of this Master Thesis is a close to functional search and rescue unmanned aerial system. Some parts and some development are missing for the system to be considered operational, but it is possible to further develop it into something usable for search and rescue operations within a reasonable short time. An unmanned aerial vehicle used in previous projects at NTNU was further developed and used as part of the prototype unmanned aerial system developed in this Thesis. A human detection system was also created to be part of the prototype system and some tests of this system are elaborated in the report. The testing of the implemented unmanned aerial system proved the feasibility of developing such a system for use during search and rescue operations, and all system components used performed well.

The findings of this report can be used for many other applications as well as search and rescue. The prototype shows that a simple, low cost, easy-to-use unmanned aerial system can be set up to perform quite complex tasks and can thus be used for many applications. An unmanned aerial system is possible to develop for use in search and rescue and the prototype made in this Thesis can be used as a starting point for further development of such systems.

This report concludes that the concept of utilizing unmanned aerial system in search and rescue operations is highly feasible and possible to develop. A low cost, easy-to-use, integrated unmanned aerial system can be developed for use during search and rescue operations and could possibly be a valuable tool for the search and rescue personnel. The system could help save time and increase the efficiency of search and rescue operations, and provide an alternative to the manned aircraft used today. The intent of an unmanned aerial system is not to replace any part of today's working search and rescue operations, but rather be a supplement or alternative for the personnel to use if desired.

Sammendrag og Konklusjon

Hovedmålet ved denne masteroppgaven var å bevise muligheten av å bruke et ubemannet luftfartøysystem under søk- og redningsaksjoner, og også å presentere og utvikle en løsning for et slikt system. Dette konseptet er bevist gjennomførbart i denne rapporten, og en mulig systemløsning for dette konseptet er presentert. Et prototypesystem er utviklet ved å bruke mange allerede utviklede enheter og sammenkoble disse. Det ubemannede luftfartøysystemet er utviklet for å fungere med en høy grad av autonomitet fordi søk- og redningspersonell ofte er opptatt med andre oppgaver og har dermed ikke nødvendigvis tid til å operere et slikt system. Bruk av autonome systemer vil også redusere nødvendigheten av å trenge erfarent personell for at systemet skal kunne benyttes, og på denne måten gjøre det mer sannsynlig at systemet vil bli brukt.

Nødvendigheten av og ønsket om å bruke ubemannede luftfartøysystemer under søk- og redningsaksjoner har fremkommet gjennom et omfattende litteratursøk og denne rapporten har blitt skrevet for å hjelpe fremtidig utvikling av et slikt system. Mange ubemannede luftfartøysystemer er allerede i bruk i dagens samfunn i mange ulike applikasjoner. Dog, ikke mange er spesielt utviklet med tanke på bruk under søk- og redningsaksjoner og følgelig er utviklingen av et slikt spesialisert system ønsket. Fremgangen gjort i forhold til teknologien av ubemannede luftfartøysystemer har nylig gjort slike systemer mer komplett, og dermed er de blitt ansett for bruk i flere applikasjoner enn før. Et godt kjent utsagn er at ubemannede systemer blir ofte tenkt på for bruk i operasjoner som er kjedelige, slitsomme eller farlige for mennesker å utføre. Under søk- og redningsaksjoner må ofte søke-personalet dekke store områder for å lokalisere de savnede personene og dette arbeidet kan være slitsomt. Et system som er enkelt å sette opp og bruke og som kan gjøre det samme søket som søk- og redningspersonalet bare raskere og mer effektivt, vil være fordelaktig for søkepersonellet og dermed også de savnede personene. I et søk- og redningsscenario vil det være viktig å redusere tidsbruken fordi desto mer tid som brukes, desto farligere vil situasjonen kunne bli for de savnede personene og dermed kan spart tid, bety spart liv. A ha et system som kan korte ned tidsbruken under søk- og redningsaksjoner er dermed veldig ønsket.

For å bevise at konseptet ved å bruke ubemannede luftfartøysystemer under søk- og redningsaksjoner er gjennomførbart, ble et litteratursøk utført for å skaffe informasjon rundt andre løsninger og prosjekter og resultatet av dette litteratursøket er presentert i denne rapporten. Rapporten oppsummerer også hvordan en søk- og redningsaksjon kan bli utført, og dette er gjort for å kunne gi informasjon rundt hvordan designet av et søk- og rednings ubemannet luftfartøysystem kan være. En løsning for et systemdesign og en prototype basert på denne løsningen er presentert og implementert.

Systemet er også testet, og testresultatene er presentert og diskutert i denne rapporten. Det ubemannede prototype luftfartøysystemet består blant annet av et kommersielt autopilot system sammen med sine komponenter som er nødvendig for at dette systemet skal fungere, og dette er montert på et fly som er en flygende vinge. Systemet har også muligheten for å få montert et kameradeteksjonssystem om bord på flyet slik at det kan detektere mennesker fra luften. Deteksjonssystemet ble utviklet slik at det kunne bli brukt om bord på et ubemannet luftfartøy og består av et kamera og en innebygget datamaskin som kan kjøre et deteksjonsprogram. Dette deteksjonssystemet er også testet og evaluert i denne rapporten. Rapporten er skrevet for å gi andre utviklere et slags veikart å følge under framtidig utvikling av konseptet som er utforsket i denne masteroppgaven, og dermed komme nærmere realiseringen av et operasjonelt søk- og rednings ubemannet luftfartøysystem.

Prototypen for et ubemannet luftfartøysystem som er utviklet i denne masteroppgaven er nært et funksjonelt system. Noe deler og noe utvikling mangler for at systemet skal kunne vurderes som operasjonelt, men det er mulig å videreutvikle systemet til noe som er brukbart under søk- og redningsaksjoner innen relativt kort tid. Et ubemannet luftfartøy brukt i tidligere prosjekter ved NTNU ble videreutviklet og brukt som en del av det utviklede ubemannede prototype luftfartøysystemet i denne masteroppgaven. Et menneske deteksjonssystem ble også utviklet for å være en del av det utviklede prototype systemet og noe testing av dette systemet er forklart i rapporten. Testingen som ble utført av det implementerte ubemannede luftfartøysystemet beviste at utviklingen av et slikt system for bruk under søk- og redningsaksjoner, er gjennomførbart, og alle systemkomponenter som ble brukt fungerte bra.

Resultatene i denne rapporten kan i tillegg til søk- og redning, bli brukt for mange forskjellige applikasjoner. Prototype systemet viste at et enkelt, lavkostnads, enkelt å bruke ubemannet luftfartøysystem kan bli satt opp til å utføre ganske komplekse oppgaver og dermed bli brukt i mange forskjellige applikasjoner. Et ubemannet luftfartøysystem er mulig å utvikle for bruk under søk- og redningsaksjoner og prototypen som ble utviklet i denne masteroppgaven kan bli brukt som et utgangspunkt for en videre utvikling av slike system.

Denne rapporten konkluderer med at konseptet for å bruke ubemannede luftfartøysystemer i søk- og redningsaksjoner er høyst gjennomførbart og mulig å utvikle. Et lavkostnads, enkelt å bruke, integrert ubemannet luftfartøysystem kan utvikles for bruk under søk- og redningsaksjoner og vil kunne være et viktig verktøy å bruke for søk- og redningspersonell. Systemet kan kunne hjelpe med å spare tid og øke effektiviteten under søk- og redningsaksjoner og være et alternativ til de bemannede luftfartøyene som brukes i dag. Intensjonen med et ubemannet luftfartøysystem er ikke å erstatte noen deler av dagens allerede fungerende søk- og redningsaksjoner, men heller kunne brukes av søk- og redningspersonalet som et supplement eller alternativ hvis de ønsker dette.

Acronyms

AHRS	Attitude and Heading Reference System
BLOS	Beyond Line-Of-Sight
ESC	Electronic Speed Controller
FMU	Flight Management Unit
GCS	Ground Control Station
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
LOS	Line-Of-Sight
MCU	Micro Controller Unit
MIDCAS	MID-air Collision Avoidance System
NTNU	Norges Teknisk-Naturvitenskapelige Universitet Norwegian University of Science and Technology
RC	Radio Control
RF	Radio Frequency
ROI	Region Of Interest
RPAS	Remotely Piloted Aircraft System
SAR	Search And Rescue
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
VLOS	Visual Line-Of-Sight
VTOL	Vertical Take-Off and Landing

List of Figures

Contents

Chapter 1

Introduction

Unmanned Aerial System (UAS) technology has been around for some time and its use has recently been given more attention than previously. This is because the technology is getting better and cheaper and is thus available for use in more applications. UAS technology includes aerial systems that can act on their own without, or with minimal need, for human input, i.e. autonomously. The technology can be used for increasing the efficiency in many applications, like for instance during Search And Rescue (SAR) operations.

Utilizing UAS in SAR operations is a very interesting research field to study in today's society. UAS is being more accepted and is integrated into many applications, thus developing a tool that can be used during SAR operations is a very relevant topic to study. Many tasks during SAR might be tiresome or dangerous for humans to perform and it could therefore be wise to remove the humans from performing these tasks. A correctly designed SAR UAS will possibly increase the efficiency of a SAR operation and making the UAS autonomous could relinquish the SAR crew to perform other tasks than those, which an Unmanned Aerial Vehicle (UAV) can perform with ease. A possible scenario of utilizing UAS during a SAR operation is illustrated in Figure 1.1.

The goal of this Master Thesis is to investigate and provide information as to if and how a SAR UAS can be developed. The design of a prototype UAS that can be used during SAR operations with adequate low degree of interference to the ordinary operation should be studied and a solution to such a design presented. The intent of this Master Thesis is to provide other developers and researchers with a report that lowers the threshold of utilizing UAS during SAR operations, and provide information as to how such a system can be developed.

1.1 Background

The usage areas of UAS during SAR operations are many and various. The will and desire to develop a SAR UAS is clear and supported by for example a competition called *UAV Challenge - Outback Rescue* that is conducted in Australia every year [OutbackChallenge]. Using a UAS to help find and recover humans during a SAR mission will be very helpful for the personnel performing this work.

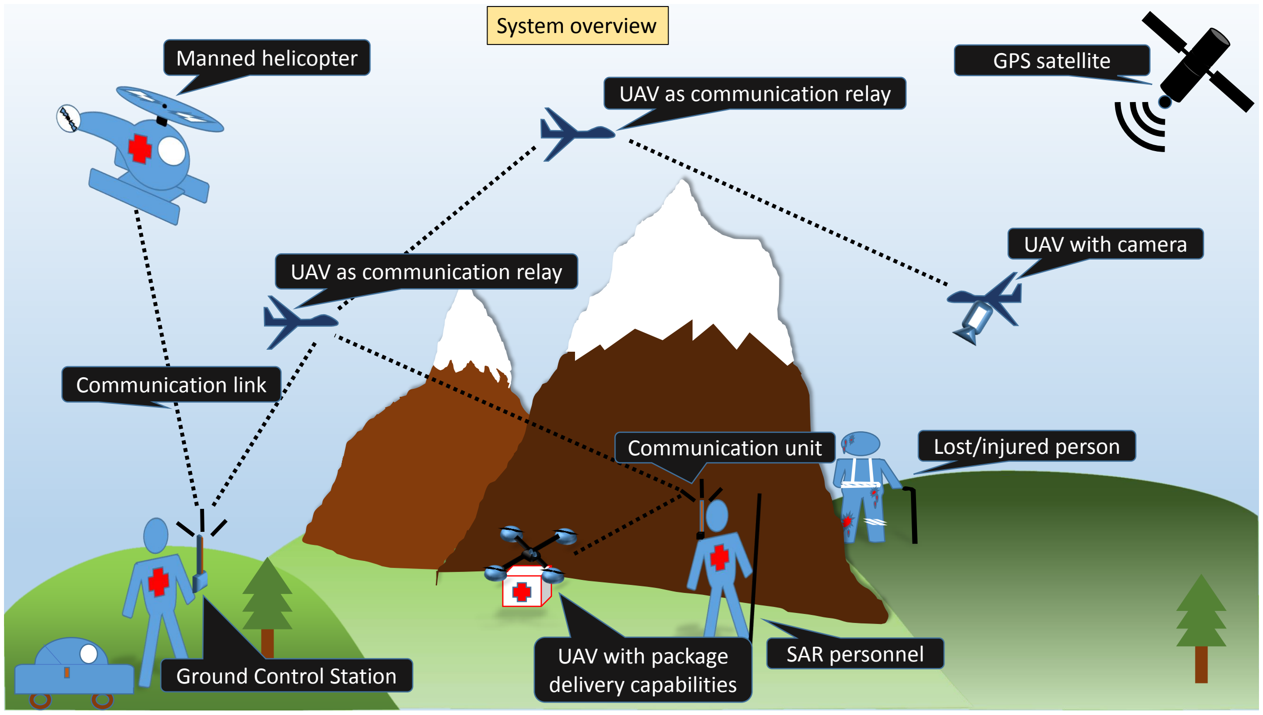


Figure 1.1: Utilizing UAS in a SAR operation

Some UAS producers claims to have developed a UAS that possesses similar capabilities that the UAS discussed during this report sought after. However, these system are either not available, or the price is too high to be considered.

This master thesis is a continuation of a project conducted at NTNU during the fall semester of 2013, which again was a continuation of a Master Thesis performed in the spring semester of 2013 [Gamnes, 2013; Hammerseth, 2013]. The result of the project was a report containing information regarding how a UAS for use during SAR operations should be developed and the report concluded with the need and desire for such a system.

1.2 Motivation

Norway is a large and sparsely populated country. The geographical character varies from the north to the south and so does the climate. The temperature can vary from 35°C in the summer to -50°C in the winter. The coastline of Norway is massive and strong winds can occur. Norway is a country where people like to explore the wilderness and consequently people get lost or injured whilst doing this, possibly resulting in the need for a SAR operation to bring them home.

The Norwegian SAR service started with some public participants, voluntary organizations, and private companies all taking part in SAR operations, but without any kind of coordination or clear line of responsibility. In 1970, the government established two Joint Rescue Coordination Centers and thus the SAR service was established. The SAR service is responsible for immediate response to an emergency to rescue persons from death or injury [Royal Ministry of Justice and Police, 2002]. In 2013, there was an increase of 9.1% compared to 2012 in the total number of incidents that the Norwegian SAR service took part of [Hovedredningssentralen, 2014]. On land, there was an increase of roughly 100 incidents, and at sea there was an increase of roughly 600 incidents. This tendency is not likely to stop as it has been increasing for some years, thus creating a tool that can help during SAR operations could be very helpful since the number of operations probably will increase in the years to come.

The reason this Master Thesis is conducted is to help the development of a tool that can aid during SAR operations, thus possibly increase its efficiency and possibly save more people than without it. The people involved in SAR operations are doing a great job and deserve any help they can get. The scenarios where UAS might help in the civil society are many such as commercial air transport, general aviation such as pleasure trips or flight training, aerial advertising, aerial observation, aerial patrol, aerial survey and mapping, firefighting, and emergency medical services [UVS-INFO]. By developing a UAS that can used during SAR operations, it will provide information of how to develop a UAS for other applications as well.

1.3 Limitations

An UAS created to be used during SAR operations will necessarily be a large and complex system. This Thesis will focus on explaining some parts in more detail than others. The final product of this Thesis is a report that can be used during the development of a SAR UAS. Due to obvious limitations during this Thesis, such as money and time, some areas had to be given less focus than others and a complete and working SAR UAS was not possible to create. Having a usable UAS was more interesting than having something that was not functional at all, thus a less extensive prototype was developed. The integration of UAS with the already present participants of a search and rescue operation should also be studied, as this will be a very important factor for UAS usage during SAR operations. This was not looked that much into during this Master Thesis because that will be easier when a functional prototype SAR UAS is developed.

This Thesis will evaluate the use of UAS in SAR situations outside large cities, i.e. wilderness SAR operations. This simplifies some of the attributes needed by the system and makes the development a bit easier. The UAS is developed for Norwegian conditions and the features are discussed accordingly.

1.4 Approach

The approach used during this Thesis was not to build everything from scratch, but rather use already made equipment and devices. Having a system that works was more appealing than having something that did not. When a system is functional, it is much easier to see which parts needs to be replaced or removed than when starting from scratch with nothing and deciding everything without any references. The parts of a system can be exchanged at a later stage for other parts and thus the system can be modified to fit the exact needs of the scenario it is to be used for. For example, in this Thesis an open-source working autopilot system is utilized instead of developing a new one. At a later stage, this system can be exchanged for some other system, possibly made from scratch if that is desired.

Another reason not to develop everything from scratch is that there exists so much equipment and devices from before. By doing everything all over, one wastes time that could be spent on other tasks. Rather than redesigning the wheel, already made components are utilized and these are interconnected and designed to collaborate so that a working system may be developed.

1.5 Main Contributions

The main contribution of this Master Thesis has been to bring UAS technology closer to being used in SAR operations, by investigating how a SAR UAS could be developed and propose a solution for such a system. The need and desire for a SAR UAS has been established and a prototype of this concept has been created to prove its feasibility. The developed prototype can be used during further research and as a starting point for realizing a complete and functional SAR UAS. The further work needed to be done to the system has been presented and some solutions discussed to provide information to future UAS developers.

The final product of this Master Thesis is a report that can be used as a guidebook when developing SAR UAS. This report presents a prototype system and proves its functionality by performing some tests using the developed system. The developed prototype UAS can thus be used during future developments either as a reference, or as a basis for the development itself. A human recognition system is also presented and implemented during this Thesis. This system can be further developed into a fully functional human detection system that is possible to use during SAR operations.

1.6 Structure of the Report

This report is intended for readers interested in the technological aspects of SAR UAS as well as for readers interested in how such a system can be utilized during a SAR operation. Not all technological terms throughout this Thesis will be explained or elaborated.

This Thesis will for some sections have an evaluation and discussion part directly after the information is presented. This is done to avoid repeating the same information several places, which in turn allows the Thesis to include more information. The programming code of the applications that are developed and tested throughout this report, is mostly listed in the appendix.

In Chapter 2, Literature Study, information regarding the literature study conducted during this Thesis is provided. The study is summarized and some of the information is discussed.

Chapter 3, Background Theory, explains the different theories used during the report. This chapter is included so that information will not have to be repeated several places, and it can be used as a reference for the reader.

Chapter 4, Search and Rescue, describes how a SAR operation is conducted and discusses how a UAS can contribute to such operations.

Chapter 5, Unmanned Aerial System, introduces UAS technology and also presents and discusses a solution for a system that can be used in a SAR scenario.

Chapter 6, Object Recognition System, describes the design of a low cost, low power, human recognition system and describes how this can be implemented to work on board a UAV during SAR.

Chapter 7, UAS Prototype, presents and discusses a prototype UAS developed based on the solution presented in Chapter 5.

Chapter 8, Prototype UAS Setup and Testing, explains the setup and testing of the prototype system presented in Chapter 7.

Chapter 9, Further Work, presents and discusses the possible future work that can and needs to be performed in order to obtain a complete and fully functional SAR UAS.

Chapter 10, Discussion and Conclusion, summarizes and discusses the project as a whole and concludes the work that has been accomplished.

Finally, Appendix A, B, and C list some information and also includes the source code of the applications that were developed throughout the report.

Chapter 2

Literature Study

This chapter will present how the information retrieval process was performed during this Master Thesis. The chapter will explain how the information was gathered, and why the specific information sources were used. The most important literature findings will be discussed in some detail towards the end of the chapter.

2.1 Research Method

The process of retrieving information can be approached in many different ways. Reading books and articles is an important method for gathering information, but using solely this approach would be insufficient or ineffective in many situations. Socializing with other researchers and using the Internet is nowadays the most important source of acquiring information quickly. Using this approach, the basic literature is much easier and more quickly found and thus one saves a lot of time compared to solely reading books and articles.

Any literature that is acquired needs to be assessed based on credibility, objectivity, accuracy, and suitability [VIKO]. A lot of the information on the Internet is not trustworthy and one needs to be sure that the information gathered is correct. The credibility is partly based on where the information is found. By using well-known sources, or perhaps the school library, one is almost certain to find good and valid information. Doing a quick Google search might come up with many answers, but the credibility of these answers might be harder to determine. Objectivity is a hard criterion to evaluate. Almost all persons that writes a book or paper on a topic has their own opinion on this topic. A way to determine the objectivity is to find other sources of information that backs up each other. Finding two independent articles on the same topic with the same point of view is a good approach. The accuracy and suitability of the information are also hard to determine. Having some background knowledge on the topics that are studied is essential to determine if the information is valid for the scenario oneself is researching or not.

2.1.1 Choice of Information Source

The primary information sources used for this Thesis, was attending a conference and reading literature. A lot of information was acquired during the pre-study, which facilitated the literature search during this Master Thesis. The information was easier to find since the search words were easier to establish. Using key-words from the articles already acquired, made finding new articles easier.

Using the capabilities of the school's library as well as its online journals proved to be a valuable asset. *Scopus* was used as the main search engine for finding articles and Google was the second choice [Scopus; Google.com]. If articles or books were not available online, the library was used in order to acquire this, or similar information regarding the topic in interest. The *Scopus* search engine is not personalized in any way and thus provide search results that are more objective than what *Google* provides. *Google* has the tendency of personalizing the results based on previous searches and "clicks", and will therefore not necessarily provide the most objective search results to its users.

2.2 Conference

This section will list some of the information acquired during a conference that was attended in the pre-study of this Thesis. The conference, called *UNC-13*, was organized by *UAS Norway* and was attended at November 12th, 2013 [UAS Norway]. The benefits of attending the conference were:

- Learned more about the status of the current UAS technology
- Learned more about who the main UAS market contributors are
- Knowledge of how to develop a UAS for use in SAR operations was acquired

The most important knowledge gotten from the conference was that UAS is a relatively new technology area and thus not all regulations and rules regarding their use are specified yet. Therefore, if a system is developed, it has to be made such that it can be upgraded or changed in order to satisfy these regulations once they are determined.

Another important fact that was learned, was that the need for harmonized laws is essential for the use of UAS in the world. A UAV is not necessarily created to fly in only one country and thus having harmonized rules and regulations between countries will make the development and use much easier. The government are the ones that are deciding these laws, but they do not possess the information or knowledge to make all the necessary decisions regarding UAS themselves. It is therefore important for UAS developers to provide them with the information they need in order to make the correct decisions.

Safety is of course very important when designing systems that will be operated in the public arena. A concern regarding UAS is that they will disturb the normal air-traffic and thus be dangerous for the public. A solution for this could be incorporating UAS into the systems that are already used by today's air-traffic.

2.3 Related Work

When designing or developing a system there will almost certainly be someone who has done, or tried to develop the same or similar system to the one that is being developed. Using their work as background information during the development of a similar system is smart as it might remove the possibility of making similar errors as they did.

This section will summarize some of the most important findings of the literature study performed during this Thesis.

2.3.1 Projects

In order to establish the need and desired for a UAS to be used during SAR operations, a search was done trying to find similar projects or systems attempting to accomplish the same objective as this Thesis. During the pre-study, several projects that were dedicated to UAS usage during SAR missions, such as CLOSE-SEARCH and ICARUS were found [CLOSE-SEARCH; ICARUS]. These projects proves the need and desire for such a system. A person in the Norwegian Red Cross was also contacted and asked some questions regarding the need for tools during SAR operations. This person was very positive to a SAR UAS development and gave some important feedback as to how the system could be utilized. In addition, as mentioned in the introduction, a competition called *UAV Challenge - Outback Rescue* is conducted in Australia and is held to increase and encouraged the development of UAS that can be used during SAR operations [OutbackChallenge]. A online community called *S.W.A.R.M.*, which stands for Search With Aerial Rc Multirotor, is a worldwide volunteer SAR network [S.W.A.R.M.]. The network consists of UAV pilots dedicated to searching for missing persons. They offer their services during ongoing SAR operations and provide the SAR personnel or families with aerial footage and searching help at no cost.

2.3.2 Flight Systems

The design and capabilities of other flight systems are of interest during this Thesis because that is essentially what the final product will be. Other flight systems can be used for determining the design and help answer design questions that might arise during the development of a SAR UAS. For instance, a flight system called *Trimble UX5 Aerial Imaging Rover System* is claimed to be a high-speed autonomous UAV system that can be used for precision image mapping with an efficient workflow [Trimble]. The system is created to handle strong winds and "Norwegian" weather conditions. This system can be used for determining the layout of a UAV frame that could be used for SAR operations. The usage of the *Trimble* UAS is essentially how the usage of the UAS created in this Thesis is imagined, thus its design might give information as to how a SAR UAS should be developed.

2.3.3 Detection Systems

The idea behind the UAS elaborated during this Thesis is that it can used to help find and recover the missing persons during SAR operations. One obvious usage application for a SAR UAS is thus for human detection purposes. Utilizing the UAS for detecting missing persons during a SAR operation will be a highly relevant usage application. Therefore, finding literature that investigates and possibly develops human detection systems will be important.

An article presenting a computer vision approach combining the usage of thermal and color images was found during the pre-study and shows how a detection system can be created and used during SAR operations for locating humans on the ground beneath a UAV [Rudol and Doherty, 2008]. A Master Thesis was also found explained the usage of a low power device as a collision detection system [Kalvå, 2014]. This system, if further developed could be used as a detection system on board a UAV and thus be part of SAR UAS.

2.4 Summary

There are many articles and books written regarding UAS and its capabilities, but there is not that much information as to how a system can actually be developed and especially not for utilization during SAR operations. A lot of the literature presents alternatives and theories, but fails to show how this can be implemented, thus making it hard to replicate the results. The desired and benefit of utilizing UAS during SAR operations is well established by the number of projects and articles regarding this subject, but no one has yet brought the technology together in a low cost, easy-to-use specialized SAR system which can be incorporated in SAR operations, as is the intent of this Master Thesis.

Based on the pre-study and literature study of this Thesis, the need and desire for utilizing UAS during SAR operations is well established. The number of people investigating this technology area enhances the will to develop such a system and thus increases the motivation to contribute to this development. The conference attended during the pre-study gave important insight in the UAS community and made it clear that UAS rules and regulations are possible to influence. Developers needs to contribute to this work in order for the correct laws to be established and thus enabling USA to be used in the society.

Chapter 3

Background Theory

In order to make this report easier to read, this chapter will explain some background information regarding some of the topics discussed in this Thesis. This chapter is not intended to be very thorough on all topics, but rather as a reference for readers who are not specialists in all the topics discussed throughout the report. The chapter can be skipped and later used as a reference if needed.

3.1 Unmanned Aircraft

Unmanned aircraft technology is a relatively new technology area and opens possibilities in society for both private and commercial applications. This section will discuss some issues regarding the technology and its use.

3.1.1 Definition

Since unmanned aerial equipment is an up and coming, relatively new technology area, Norway does not currently have any formal definition of what an unmanned aircraft is. However, the *Civil Aviation Authority of Norway* provides the following temporary explanation:

> Any remotely controlled device that is meant to move in the air and that can be used for some sort of utility or commercial flight is to be reckoned as an unmanned aircraft.
>
> Translated and adapted from [Luftfartstilsynet.no]

Some acronyms and definitions often used when discussing UAS are listed below [UVS-INFO]. The acronyms are invariant, meaning they refer both to singular and plural form.

UAS - Unmanned Aerial System. An unmanned aircraft and its associating system elements

UAV - Unmanned Aerial Vehicle. Describes the flying part of a UAS. Can be used instead of unmanned aircraft.

RPAS - Remotely Piloted Aircraft System. A system consisting of a remotely piloted aircraft, remote piloting station, command and control links and other system elements required for the remotely piloted system to work.

RPA - Remotely Piloted Aircraft. An aircraft where a pilot is not present on board that can be remotely piloted.

RPS - Remote Pilot Station. The ground part of a RPAS where the pilot controls one or more RPA from. Can be compared with a cockpit that sits on the ground.

LOS - Line-of-Sight. Describes the viewing area between operator and aircraft. The operator can physically see the aircraft.

VLOS - Visual Line-of-Sight. Same as LOS.

BVLOS - Beyond Visual Line-of-Sight. Operation of an unmanned aerial vehicle beyond visual range of the operator.

BLOS - Beyond Line-of-Sight. Same as BVLOS.

EVLOS - Extended Visual Line-of-Sight. Operations where the pilot maintains visual contact with the unmanned aircraft through a remote pilot or observer.

3.1.2 Laws and Regulations

When dealing with technology that will interact in society where people might interfere or get in the way, there are almost certainly some laws and regulations that have to be followed. Because UAS is a relatively new technology area, the laws and regulations regarding the technology have not yet been completely determined by the government and therefore there might be some alteration to the current rules at a later stage. Designing a system when laws and regulations are not completed might yield difficulties at a later stage and this will have to be taken into consideration regarding the system design.

In Norway, the current rules for commercial unmanned flight are determined by a regulation called *AIC-N13* [Luftfartstilsynet.no]. This is a temporary regulation and is far from complete, but serves the purpose until the regulations are properly decided. *AIC-N13* explains the demands for utilizing unmanned aircraft systems commercially in Norway and on Svalbard. There are several categories of UAS and civilians in Norway are only allowed to use Remotely Piloted Aircraft System (RPAS). This is mainly because of the dangers associated with the other types, such as for example fully autonomous UAS, and there are some concerns regarding the responsibility when using fully autonomous vehicles. Beyond Line-Of-Sight (BLOS) operations, compared to Visual Line-Of-Sight (VLOS), are more dangerous and for this reason, there are higher demands to the systems that needs this feature than to the systems that do not. In Norway, one is currently not allowed to operate a UAV without having visual contact with the unit, i.e. it has to stay within VLOS. There are exceptions that can be made when the UAV is flown in a designated, controlled area where the airspace is cleared of other aircrafts, but that is solely for test purposes and has to be approved in advanced by the government. VLOS operations are more relaxed than BLOS and it is allowed to operate a UAV below the altitude of 400 feet (~122m) above ground level, and the UAV must stay within VLOS the entire flight for the operation to be legal.

The problems that the government faces regarding fully autonomous vehicles is that if an autonomous vehicle, i.e. a vehicle acting on its own without user input, is involved in an accident, who is then responsible for the damages caused by the system. Whether if it is the people who designed the system, the people who manufactured the system, or the persons who used the system is difficult to determine, and this is the reason for not allowing these system to be operated yet. An interesting topic regarding the design of autonomous systems is mentioned in an article written by an associate philosophy professor, Patrick Lin [Lin]. If cars become fully autonomous, the designers will be faced with a difficult problem designing the actions the system should perform in the case of an imminent accident. A question arises if the system is bound to hit a manned vehicle, but has to choose between two, which the autonomous system then should choose, could be close to impossible to determine.

Photography

During the pre-study of this Master Thesis, a lot of time was spent trying to get a permission to capture pictures and video from the air using a UAV. Unfortunately, getting the permission was not an easy task and slow bureaucracy and regulations made the applications process drag out for months. A temporary permission was granted by email during the pre-study and thus some pictures could be captured and used in the report.

However, during the spring of 2014, a new regulation was presented by the *Civil Aviation Authority of Norway*, which made it legal for civilians to capture and use photography taken from a UAV as long as it is not above any restricted flight or footage areas [Luftfartstilsynet.no]. This new regulations makes it possible for civilians to capture images and video without having to apply for a permission. The regulation is only applicable if the images and video are not used for business, but can be used for recreational purposes only. The regulation is also a temporary regulation lasting throughout the year of 2014 for when they will reevaluate and possibly make the regulation permanent.

3.2 UAV Control System

A UAV control system is responsible for the control of the aircraft and whether this is forwarding manual input from a pilot or controlling the aircraft autonomously, does not matter. The UAV control system can be referred to as an autopilot or Flight Management Unit (FMU) and is typically an embedded computer managing the flight, communication, and any other tasks the UAV might need to perform. There are many commercially available autopilots, but not many are considered during this Thesis as they are simply too expensive. An example of this is the *Kestrel Autopilot system v2.4 Fixed Wing*, which is a small, advanced fixed wing controller [Lockheed Martin Corporation]. The cost of this system is $5,000.00 and that does not include control application software, which costs an additional $3,695.00. An autopilot that is considered during this Thesis is a system called *APM 2.6*, which has a cost of around $160.00 [3DRobotics, a].

The name autopilot might be misleading since the plane in most cases is still controlled by a pilot through either waypoint flight, or manual flight control. However, the term autopilot will be used during this Thesis as it is the most common term used for UAV control systems.

3.2.1 APM

APM is a low cost, open-source, multiplatform autopilot [3DRobotics, a]. The system was previously called *ArduPilot*. The most recent release of the system is the *APM 2.6* and this version is used during this Thesis. It supports various unmanned vehicle types such as rovers, planes, and multi-rotor vehicle. The *APM* system is a collection of hardware, firmware, and software combined into a complete control system for UAV. Everything is open-source and the system has a large community surrounding itself. *APM* is an affordable, reliable, and flexible autopilot system and the hardware does not require any changes if another vehicle platform is desired. The different firmware versions of *APM* are called *APM:Plane*, *APM:Copter*, and *APM:Rover*. The system supports both piloted and unpiloted flight and has a pre-made Ground Control Station (GCS) application freely available for download on its website [APM, c]. The autopilot is able to handle one external UART communication link, given that the provided firmware is used as it is. Alterations in the source code may allow a second link to be established, but that is not done in this Thesis. The autopilot also has a secondary Micro Controller Unit (MCU) that is used if the primary MCU should fail. Thus, the secondary MCU acts as the autopilots failsafe controller.

The specifications of *APM 2.6* are listed in Table 3.1.

Table 3.1: Specifications of APM 2.6, [3DRobotics, a]

APM 2.6	
Price:	$159.99
Processor:	ATMEGA2560, 8 bit, 16 MHz
RAM:	8 KB
Flash:	256 KB for code, 4 MB storage flash
Input Power:	5 VDC
Size:	66 x 41 mm
Weight:	33g
Interfaces:	UART, I^{2}C, microUSB, Interface to sensors and actuators (PWM, etc.)

Features

The features of the *APM* system includes amongst others point-and-click flight mission configuration, multiple command modes, mission planning including waypoints capabilities, autonomous flight including take-off and landing, failsafe actions, real-time data transmission between UAV and GCS, and in-flight data logging.

The *APM:Plane* firmware provides several built in flight modes and some of them are listed and explained in Table 3.2.

Ground Control Station

There are several GCS applications available for controlling the *APM* system, and some are freely available for downloading online. During this Thesis, the most well developed application called *Mission Planner* is used [APM, c]. *Mission Planner* is an open-source, ground control station

Table 3.2: APM:Plane Flight Modes, [APM, d]

Flight Mode	**Description**
Manual:	In *Manual* mode, the *APM* performs no stabilization of the plane and the plane is controlled by regular RC control input.
Stabilize:	In *Stabilize* mode the *APM* performs simple stabilization of the plane. If no input is received from the pilot, the autopilot levels the plane.
FBWA:	In *FBWA*, Fly-By-Wire-A, mode, the *APM* will assist the flight similar to Stabilize mode. This is said to be the best mode for inexperienced pilots. *APM* will hold the roll and pitch specified by the control input and return to level when the input is stopped.
Auto:	In *Auto* mode, the *APM* will control the aircraft and follow a mission set by the GCS. This could be a set of waypoints or some other commands. The *Auto* mode contains two important sub-modes called *Takeoff* and *Landing*. These modes can be set by the GCS and will thus be a part of the mission that is planned for the aircraft.
RTL:	In *RTL*, Return-To-Launch, mode, the *APM* will return the plane autonomously to a pre-defined "home" location. This mode can be used in the case of communication link loss or some other error. Alternatively, simply to get the plane back to the starting point.
Loiter:	In *Loiter* mode the *APM* will control the aircraft such that it circles around a location. The plane will hold its altitude at the altitude that the plane had when Loiter mode was entered.

application and the software allows for configuration and communication with the UAV before, during, and after flight. The software can used for *APM:Plane*, *APM:Copter*, and *APM:Rover* and has the opportunity for the user to perform initial setup, configuration and tuning, simulation, flight data logging, mission status updating, and mission planning. The UAV flight can be visualized on a map during, and after flight.

Unfortunately, the *Mission Planner* software is only compatible with Windows, and the software does not include support for multiple UAV connections. There are other applications that can be used for Linux such as the newer software called *APM Planner 2.0* [APM, a]. *APM Planner 2.0* is currently under development and is an open-source GCS application for MAVLink based autopilots, including the *APM*. The application has multiple UAV connection support and runs on Windows, Mac OSX, and Linux. Unfortunately, the most recent version of *APM Planner 2.0* is a beta release and there does not exist any stable releases. The application is therefore neither used nor evaluated during this Thesis.

3.2.2 MAVLink

Micro Air Vehicle Communication Protocol, known as *MAVLink*, is a lightweight, header-only, message library used in many micro-air vehicle systems [MAVLink]. The *APM* system uses the *MAVLink* protocol as its communication backbone between the MCU and Inertial Measurement

Unit (IMU) as well as for communication with the GCS. *MAVLink* is a GNU Lesser General Public License, LGPL, licensed protocol and can hence be used in closed- and open-source applications royalty-free. *MAVLink* is used in many systems and is a good option when it comes to low cost, UAS communication protocols.

The *MAVLink* protocol does not have to be compiled along with the program in order to be used, because it is a header-only library. The library only has to be added to the include directories of the MCU. It is possible to create new and modified message types using pre-made templates and the protocol can thus be personalized for a specific system. *MAVLink* is a stateless protocol, but uses heartbeat messages to track if a system is still present or alive. The *MAVLink* protocol is inspired by the reliable CAN protocol and supports 255 aircrafts at the same time [CAN-cia.org]. Every aircraft is assigned an ID ranging from 1 to 255. *MAVLink* claims to be a non-intrusive protocol, which means that it does not become a central part of the onboard architecture, hence allowing a simpler on board system architecture to be used.

Packet Anatomy

The anatomy of a *MAVLink* packet or frame is shown in Figure 3.1. The packet begins with a start sign byte (STX) indicating the start of a frame and a length indicator byte (LEN) indicating how long the packet is. Then there is a sequence number byte (SEQ) allowing packet loss to be detected and a system ID byte (SYS) for determining the origin of the packet. The component ID byte (COMP) is used for differentiating different components of the same system. Then follows the message ID byte (MSG) which tells the system what sort of message the packet contains. The following bytes are the payload and the last two bytes of the frame are checksums.

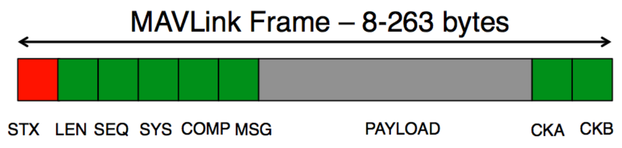


Figure 3.1: MAVLink packet anatomy [MAVLink]

The message ID is used for determining what sort of payload that is transmitted with the package and what the different payload bytes signifies. For example, if the message ID is 24, the standard message descriptions tells that the packet contains the global position information of the unit as given by the position receiver on board the unit. Of course, if personalized messages are used, this message ID could have a different meaning. A list found in the MAVLink documentation can be used for determining the standard meaning of the different bytes in the payload of these packets given the message IDs.

3.3 Global Navigation Satellite System

A Global Navigation Satellite System (GNSS) is a system used for determining the location of a device relative to some coordinate system [GNS]. The most commonly known GNSS is the NAVSTAR Global Positioning System (GPS), which is the system used during this Thesis.

3.3.1 Global Positioning System

A GPS can be used for determining the position of an object relative to a coordinate system [Parkinson and Spilker, 1996]. To achieve a three-dimensional position, the GPS receiver needs to have contact with at least four GPS satellites. By utilizing the signal transmitted from these satellites the latitude, longitude, and altitude of the receiver can be calculated. More information regarding GPS is mentioned in the pre-study of this thesis [Gamnes, 2013].

3.4 Embedded Computer System

An embedded computer system can be described as:

> *A computer system:*
>
> - *that is designed and optimized to execute a dedicated task*
> - *that is embedded as a part of a larger system, which typically consists of other types of components than computers*
> - *where the computer itself is not necessarily the "purpose" of the system*
>
> Quoted from [TTK4155, NTNU, 2012]

Embedded computers will be used during this Thesis not only to control the UAV, but also as part of a human detection system. The two embedded computers used in this Thesis are the *BeagleBone Black* and the *APM 2.6* [BeagleBoard.org; 3DRobotics, a].

3.4.1 BeagleBone Black

This section will summarize some of the features of the *BeagleBone Black* embedded computer. In addition to what was discovered in the pre-study, the *BeagleBone Black* has been proved viable for use as a collision detection system given some changes to its operating system [Kalvå, 2014]. In the thesis written by Andreas Kalvå, the *BeagleBone Black* was proved better suited in a low cost, low power system for performing computer vision tasks than the popular embedded computer named *Pandaboard* [Pandaboard.org]. Therefore, the *Pandaboard* will not be evaluated in this Thesis. During the pre-study, several embedded computers were evaluated and of those, the *BeagleBone Black* was found to be the better option and is therefore used during this Thesis.

The *BeagleBone Black* is an open-source, open-hardware, single-board embedded computer that has a processor running the ARMv7 instruction-set. All the device inputs are ESD-protected and the outputs have over-current protection. The serial communication of the *BeagleBone* is accomplished using 3.3 V and it does not accept 5 V. The specification of the *BeagleBone Black* are listed in Table 3.3.

Operating System

The *BeagleBone Black* can be used with many different operating systems such as *Ubuntu* or *Angstrom. Angstrom* is the default operating system and will therefore be used during this The-

Table 3.3: Specifications of BeagleBone Black

BeagleBone Black	
Price:	$45
Processor:	Sitara AM3359, ARM cortex A8, 1GHz
RAM:	512 MB DDR3L
Flash:	2 GB eMMC, and uSD
Power:	210-460 mA@5V
Size:	86 x 53 mm
Weight:	40g
Interfaces:	69 GPIO, I^2C, CAN, SPI, 4 timers, 4 serial ports, 8 PWM, 7 analog in, USB
Ethernet:	Yes
HDMI:	MiniHDMI

sis.

The setup of the *BeagleBone Black* is explained in Appendix A.

3.4.2 A Single Embedded Computer as a Complete Flight System

The *APM 2.6* is not powerful enough to perform any other tasks than flight control due to the limitations of its MCU. The *BeagleBone Black* is powerful enough to perform both flight control and other tasks, but there is currently no available working implementation of UAV control system for the device. A project is undergoing that has the goal of turning the *BeagleBone Black* into a complete autopilot for use in a UAS, but this has not been completed yet and it is therefore not available for use [APM, b].

Having several embedded computers in a system might increase the system complexity and depending on the application, it might be better to avoid it. If an embedded computer can run both the flight control and the other tasks that are necessary to perform, it might be better to implement the entire system on one device. This would probably simplify the system architecture. However, having two devices gives the safety of redundancy if one of them should fail. Thus, having two devices both capable of performing flight control would be safer than only having one, and perhaps this is the better option to use.

3.5 Computer Vision

Computer vision is a research field that includes many subjects such as acquiring, processing, and interpreting input from vision systems, usually cameras [Szeliski, 2011]. Computer vision can be used in many applications, for example in optical character recognition, automotive safety, object detection, and surveillance. Computer vision will be used during this Thesis for object detection purposes.

Computer vision algorithms are often computationally extensive that requires substantial processing power by the device doing the processing. For this reason, implementation of computer vision programs on small, low power, embedded computer system may cause problems for time critical applications and writing efficient code will thus be important to maintain the necessary processing speed.

3.5.1 OpenCV

OpenCV is a free, open-source computer vision library, free to use under the free-software BSD license [OpenCV]. *OpenCV* is regarded as the leading standard within open-source computer vision software and is surrounded by a large community. The library comes with many pre-implemented algorithms, which facilitates development and prototyping. Computer vision algorithms can be quite complex and thus creating them from scratch can be difficult and time consuming. The *OpenCV* library will be used in this Thesis for prototyping computer vision detection applications.

The *OpenCV* library is written in the programming language C++ and therefore its primary programming interface is available in C++. There are other options that enables C, Python, Java, and MATLAB to be used and the library supports the operations systems Windows, Linux, MAC OS, iOS, and Android. However, all the newest algorithms and developments of *OpenCV* are done and optimized for C++. The focus during the development of *OpenCV* is real-time applications and therefore many of the algorithms are designed and optimized to run fast. The library can take advantage of several processor if a multi-core computer is used.

Optimization

OpenCV utilizes matrix operations a lot in its calculations and accelerating these operations will increase the performance of an application utilizing *OpenCV*. The library was originally created for the Intel platform computer and since the code of *OpenCV* was written to utilize the Intel processors, the performance on ARM processors, like the one on the *BeagleBone Black*, is not as good as it could have been. In order to increase the performance of *OpenCV* on ARM processors, some optimization steps can be used during the compilation of the *OpenCV* library [Kalvå, 2014]. Andreas Kalvå tried this in his Master Thesis and got a speed up in run-time of something in between 8%, up to 40%, depending on the algorithm that was run.

The optimization is not performed for the detection system developed in this Thesis and the *OpenCV* library is used as pre-installed on the Angstrom distribution, which is the recommended operating system to installed on the *BeagleBone Black*.

3.5.2 Webcam

The webcam used for the prototype detection system in Chapter 6, is a Logitech c270 HD Webcam. The specifications of the webcam are listed in Table 3.4.

The webcam was already at hand and worked right out of the box with the *BeagleBone Black*. In the future, a better suited, perhaps with a lower power consumption, webcam can be chosen.

Table 3.4: Specifications of Logitech c270 HD Webcam, [Logitech]

Logitech c270 HD Webcam	
Price:	$50
Connection:	USB
Resolution:	1280 x 720
Microphone:	Built-in
Power:	up to 500 mA
Size:	21 x 7.6 x 15.2 cm
Weight:	227g

3.6 Cameras

A camera is essentially a device that performs a mapping between the 3D world and a 2D image [Hartley and Zisserman, 2004]. Several models are possible to use for explaining this mapping, and the simplest, most well known is called *the basic pinhole model.*

3.6.1 Pinhole Camera Model

In the model, a point in space ($\mathbf{X}_c$) is mapped to a pixel point ($\mathbf{X}_i$) on an image plane. This image plane is imagined to lie with a distance of f (focal length) in front of the center of projection (camera center). By imagining a line joining the point in space and the center of projection, the pixel point will be the point in which this line meets the image plane. This model is illustrated in Figure 3.2 and shows the point $\mathbf{X}_C$ in the camera coordinate space mapped to pixel $\mathbf{X}_i$ on the image plane.

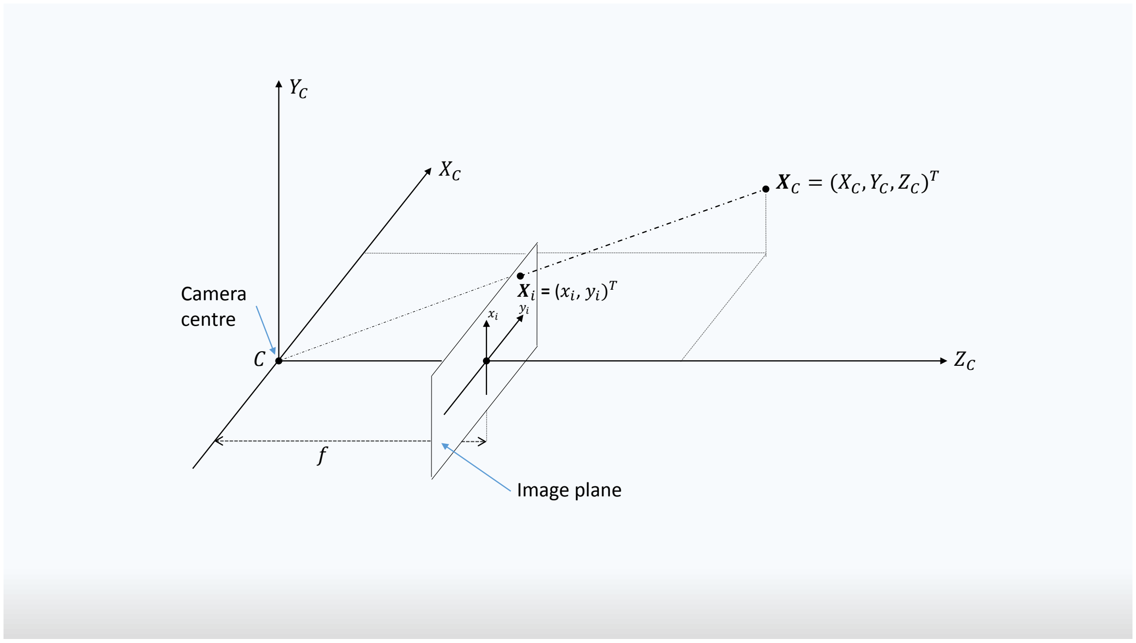

Y_C
X_C
$\boldsymbol{X}_C = (X_C, Y_C, Z_C)^T$
$\boldsymbol{X}_i = (x_i, y_i)^T$
Camera centre
x_i
y_i
C
Z_C
f
Image plane

Figure 3.2: Basic Pinhole Model

3.6.2 Camera Matrix

A notation often used for describing the mapping between a homogeneous world ($\mathbf{X}_w$) and a homogeneous pixel coordinate ($\mathbf{X}_i$) is as following

$$\mathbf{X}_i = \mathbf{P}\mathbf{X}_w \tag{3.1}$$

where $\mathbf{X}_i = (x_i, y_i, 1)^T$ is a pixel location on the image plane, $\mathbf{P}$ is a 3 x 4 *camera projection matrix*, and $\mathbf{X}_w = (X_w, Y_w, Z_w, 1)^T$ is a world coordinate. Included in the *camera projection matrix* are the 3 x 3 *camera calibration matrix* called $\mathbf{K}$, the 3 x 3 rotation matrix $\mathbf{R}$, and the inhomogeneous position of the camera center ($\widetilde{\mathbf{C}}_w$) in the world coordinate system. The inhomogeneous position of the camera relative to the world coordinate system is the translation of the camera coordinate system assuming that the camera center was initially at the origin of the world coordinate system. The translation and rotation of a camera coordinate system relative to a world coordinate system is illustrated in Figure 3.3.

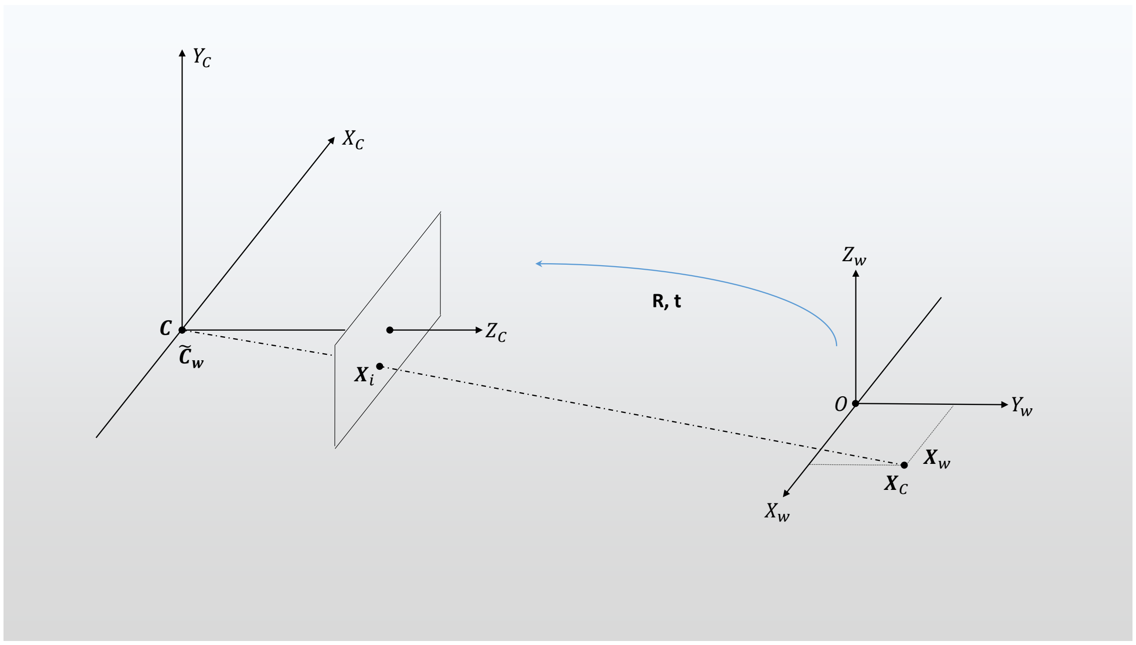


Figure 3.3: Coordinate system rotation and translation

Equation 3.1 can also be written as

$$\mathbf{X}_i = \mathbf{K}\mathbf{R}\,[\mathbf{I} \mid -\widetilde{\mathbf{C}}_w]\,\mathbf{X}_w \tag{3.2}$$

where the *camera projection matrix* is written out with all its components [Hartley and Zisserman, 2004]. $\mathbf{I}$ is a 3 x 3 identity matrix.

Rotation and translation

Describing the rotation and translation between coordinate systems is a research topic in itself. During this Thesis, the rotation matrices used for the x, y, and z-axis are shown in Equation 3.3 [Diebel, 2006]. The order these are multiplied is $\mathbf{R}_x$, $\mathbf{R}_y$, $\mathbf{R}_z$, which in turn constitutes the complete $\mathbf{R}$ matrix in Equation 3.2. ϕ, θ, ψ are respectively the Euler angles of rotation around the x, y, and z-axis, defined using the right-hand coordinate system rule. A problem with using Euler angles, as done in this Thesis, is singularities. Given certain angles, singularities issues with the rotation matrices could arise. This can be avoided by switching the order of multiplication around the singularity points, or using unit quaternions. However, since the UAV used during SAR operations will most likely not rotate around itself or fly upside-down, this will probably not be a big problem during this particular application.

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}, \mathbf{R}_y = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \mathbf{R}_z = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

3.6.3 Localization

A 2D point in an image can be back-projected to a collection of 3D points in the camera coordinate system and hence, world coordinate system [Morvan, 2009]. These 3D points are all projected to the same particular pixel on the image plane. The 3D points can be thought of as a ray connecting the camera center going through the pixel point on the image plane, to each of the 3D points. The stapled line in Figure 3.3 is an illustration of such a ray. All 3D points on this ray will be mapped to the same pixel $\mathbf{X}_i$ on the image plane.

By including the information about the Z-world coordinate of an object in an image, the two other coordinates can be derived. In SAR, the UAV will fly above the ground with a known altitude, and thus the objects located below on the ground can be assumed to be at a known Z-world coordinate. Setting this Z coordinate to zero is an assumption made to facilitate the calculations and Equation 3.1 can thus be rewritten as

$$\mathbf{X}_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \mathbf{P}\mathbf{X}_w = \begin{bmatrix} \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3 & \mathbf{P}_4 \end{bmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w = 0 \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_4 \end{bmatrix} \begin{pmatrix} X_w \\ Y_w \\ 1 \end{pmatrix} \tag{3.4}$$

where the third row of $\mathbf{P}$ is removed because of the Z-world coordinate being zero [Hartley and Zisserman, 2004]. The new reduced $\mathbf{P}$ matrix can be renamed $\mathbf{H}$, and a formula for calculating the homogeneous world coordinate $\mathbf{X}_w$, given that the Z-coordinate has a value of zero, can be given. The new homogeneous world coordinate can be called $\hat{\mathbf{X}}_w = (X_w, Y_x, 1)^T$, and given the homogeneous pixel position $\mathbf{X}_i$, this coordinate can be calculated using the following equation

$$\hat{\mathbf{X}}_w = \mathbf{H}^{-1}\mathbf{X}_i \tag{3.5}$$

where $\mathbf{H}^{-1}$ is the same as $[\mathbf{P}_1\ \mathbf{P}_2\ \mathbf{P}_4]^{-1}$. Normalizing $\hat{\mathbf{X}}_w$ by dividing the entire vector with the third element transforms the homogeneous coordinate to the inhomogeneous world coordinates X_w, Y_w, and $\mathrm{Z}_w = 0$, calculated from a pixel position on the image plane.

3.6.4 Camera Calibration

As explained, a camera can be described as a unit performing a mapping of 3D world points, to a 2D image plane. This camera mapping if often not perfect and in some cases, the mapping results in distortions, which are visible on the captured image. Barrel, pincushion, and mustache distortions are common [Kalvå, 2014].

The *camera calibration matrix*, **K**, from Equation 3.2 is a matrix used for mapping points in the camera coordinate system to a pixel position in the image. The matrix can be listed as following

$$\mathbf{K} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \tag{3.6}$$

where the element f is the camera focal length mentioned earlier, and the elements p_x and p_y are the coordinates of the principal point [Hartley and Zisserman, 2004]. The **K** matrix is specific for each individual camera, and this matrix along with some distortion factors can be used for calibrating an image and rectifying it, thus removing the distortions of that image.

A camera calibration is a process that helps remove the distortions from an image. The process determines the *camera calibration matrix* **K** and other distortions factors needed to remove distortions from an image. The *OpenCV* computer vision library is capable of performing such a calibration of a camera, and provides a standardized program for doing this. By running this program, the library finds out the camera calibration matrix and distortions factors of the camera used. These parameters can then be used for rectifying an image and thus removing the distortions in that image.

3.7 Concurrent Programming

In order to make a computer capable of performing multiple tasks apparently at the same time, the term concurrent programming comes into hand. Concurrent programming can be described as techniques that allow parallelism within a computer program [Burns and Wellings, 2009]. The concurrent programming techniques solves synchronization and communication issues and allows for writing programs more natural to the way humans think.

There are three main reasons for programming concurrently; the real world is a concurrent place, processors can utilized more efficiently, and more than one processor can solve the same problem at hand simultaneously.

3.7.1 Threads and Processes

Programming concurrently can be accomplished using what is referred to as threads or processes. The difference between a thread and a process is often not very clear and they often get interchanges with each other. To be clear, in this report a process is a single program execution that can contain several threads. A process has its own memory context and does not share this with other processes. Threads represents a single thread of execution. A process may contain several threads and thus these threads share the same memory space.

During the development of the detections system in Chapter 6 and in Section 8.3, the standard *POSIX, Threads extensions (IEEE Std 1003.1, 2013 Ed)*, more commonly known as *Pthreads*, is used for creating and manipulating threads [IEEE]. The POSIX standard defines an API in the C programming language for doing thread operations, and the API can be used with C++.

3.7.2 Inter Process Communication

Inter process communication is a term used to describe communication between threads or processes. The fact that threads share the same memory space makes inter process communication a lot easier for them as oppose to for processes. The term inter-task communication can be used for thread communication as not to mix it with the communication between processes. Shared variables is probably the easiest form of inter-task communication.

3.7.3 Synchronization

In thread programming, there might be sequences of statements that has to be executed indivisibly or in the correct order to achieve the desired result. If not, some error or unwanted behavior might occur. If two threads share a variable, the access to this variable needs to be controlled so that both threads does not access the variable at the same time. This is where synchronization methods enter. Using mutual exclusion, or popularly called a mutex, when accessing a variable is a method for accessing shared variables safely. Another synchronization method between threads is called condition synchronization. This allows one thread to wait for another thread to perform some operation or be in a defined state, before performing its own operation and continue its execution. Starvation is a term used in concurrent programming to define an error condition where a thread is never allowed access to a desired resource due to some other thread gaining access before it.

3.8 Radio Frequency Communication

Radio Frequency (RF) communication is accomplished by sending information between devices using electromagnetic energy waves [Digi International Inc., a]. A source creates electromagnetic waves, which are interpreted by a receiver, thus enabling communication between the source and receiver. The frequency of a radio signal is measured in Hertz, which is the same as cycles per second. A wavelength is inversely proportional to the frequency of a signal. A signal with longer wavelength can travel farther and penetrate through more obstacles than a signal with shorter wavelength. The unit decibel (dB) is used to represent the power of a RF signal. Some common measurements of RF signals are transmitting power and receiver sensitivity. A rule of thumb says that using a lower frequency gives longer range and lower capacity compared to a higher frequency.

3.8.1 Laws and Regulations

High intensity radio signals have the potential of harming people or disturbing other radio systems such as the emergency network. Because of this, there are several laws and regulations regarding the usage of radio frequencies. Transmitting power and different frequencies are controlled and regulated, and the rules are decided by the country in which one operates. This means that some devices developed in USA cannot be used in Norway because of the frequency or power output they utilize. The regulations in Norway are decided by the "Post- og teletilsynet" [Post- og teletilsynet, 2012]. An overview of the different frequencies, and which are currently in use, can be found at their website [Post- og teletilsynet].

3.8.2 Network Topology

The setup, or structure of devices or nodes in a network, can be called a network topology. Some of the most common topologies will be briefly explained here [Meador, 2008]. The topologies are illustrated in Figure 3.4.

Point-to-Point Network

This is the simplest form of network topology consisting of a permanent link between two devices.

Point-to-Multipoint

In a point-to-multipoint, or star network, there is one central top-level device, which all others have a point-to-point connection with. The top-level device works as the master of the network initiating communication between the devices.

Peer-to-Peer

A peer-to-peer network is a network where all devices have equal status. They all act as suppliers and consumers of resources and the network works without the need of a top-level centralized device. The network may consist of many "types" of topologies such as a mixture of point-to-point and ring (each device is connected to two others in a ring manner) topology.

Mesh Network

A mesh network is a network where each device may have multiple paths to another destination device. The best route from sender to destination is based upon connectivity, speed, and pending tasks of the devices. There does not have to be a direct link between all devices because transmissions can be forwarded from device to device.

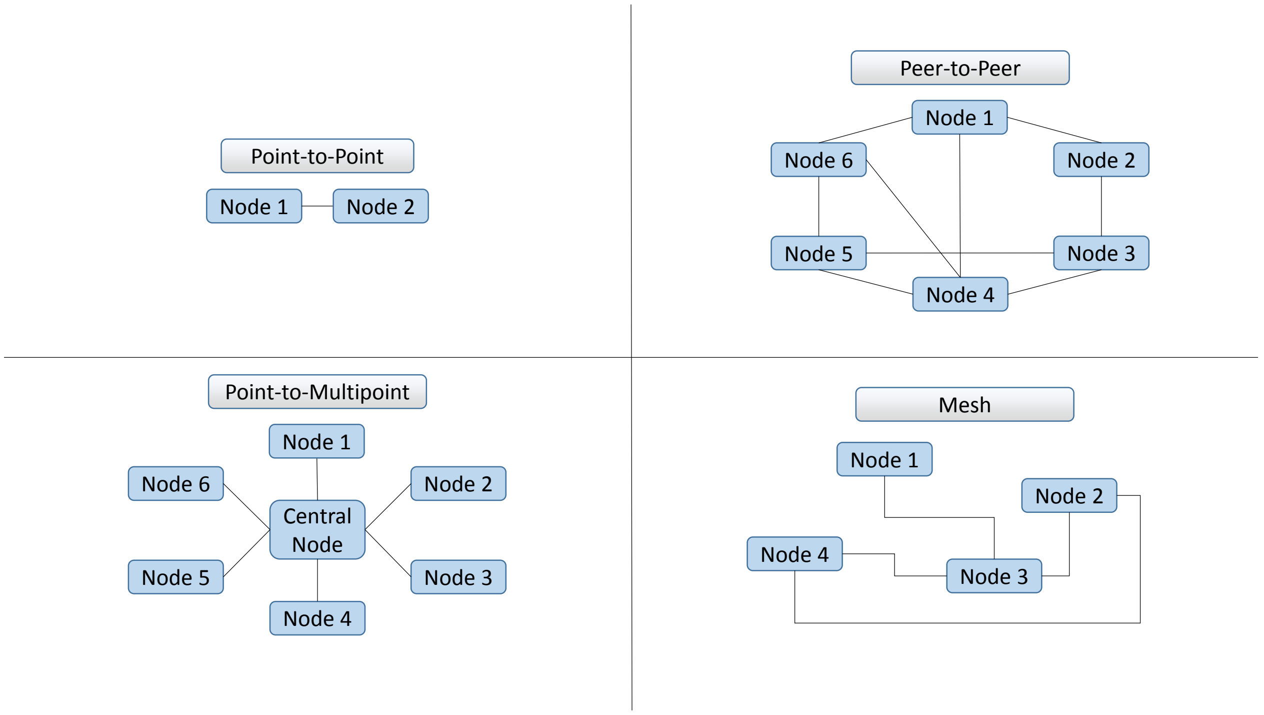


Figure 3.4: Network Topologies

3.8.3 RF Modules

Several possible choices of RF modules can be used in a low cost system. However, finding a module that has long range, low power consumption, and is easy to use can be a difficult task. RF modules are often created to possess different types of features and it is necessary to choose which one to use based on the particular application.

Two modules that were evaluated during this Thesis were the *XBee-PRO S2B* and *Radiocrafts RC1280HP*. They were both available at the institute and both are certified for license-free use in Europe and therefore evaluated in this Thesis.

Given that SAR operations are somewhat governmentally controlled, the possibility of obtaining and utilizing a separate, application dedicated radio frequency is a possibility. If a dedicated frequency is obtained, the radio module used during SAR operations could be developed specifically for this frequency, and thus could probably made more powerful thus possibly extending the communication range and increasing the data-throughput for a SAR UAS. A special license could also be acquired by the SAR UAS operators allowing more powerful radio modules to be used even without such a dedicated frequency.

XBee

XBee radio modules are easy to use embedded radio solutions giving the user the capability of utilizing multiple topologies for communication [Digi International Inc., b]. *XBee* utilizes its own protocol, which is built upon the more commonly known *ZigBee* protocol, i.e. a protocol aimed towards enabling low cost, low power wireless sensor and control networks [ZigBee.org].

The *XBee-PRO S2B* module is a small RF module with a range of up to 1500 m Line-Of-Sight (LOS). The unit can be programmed to consume very little power and the module is interoperable with other *ZigBee* devices. The specifications of the module are listed in Table 3.5.

Radiocrafts

Another radio module considered during this Thesis is the *Radiocrafts RC1280HP* RF module [Radiocrafts]. The unit is very small and incredibly light and can achieve a range of up to 6 km LOS. To mention some applications that the module may be used in are automatic meter reading, alarm and security systems, fleet management, and telemetry stations. The specifications of the devices are listed in Table 3.5.

Table 3.5: Specifications of XBee-PRO and Radiocrafts modules

	XBee-PRO S2B, International variant	**Radiocrafts RC1280HP**
Price:	$29	$55
Range:	1500 m LOS	6000 m LOS
Power Consumption:	Transmit: 117 - 132 mA	28 - 650 mA
	Receive: 47 - 62 mA	20.7 mA
	Idle: 15 mA	0.9 mA
Power-down current:	3.5 μA	0.003 μA
Frequency:	2.4 GHz	868-870 MHz
Protocol:	XBee - ZigBee based	Embedded RC232
Tx Power:	10 mW	max 500 mW
Data Rate:	250 Kbps	4.8 Kbps
Antenna:	Integrated whip antenna,	Multiple options
	Embedded PCB Antenna	
	RPSMA	
	U.FL connector	
Supported Topologies:	Point-to-point	Point-to-point
	Point-to-multipoint	Point-to-multipoint
	Peer-to-peer	Peer-to-peer
	Mesh	
Supply Voltage:	2.7 - 3.6 V	3.3 V
Serial Data Interface:	3.3 V Serial	UART 3/5 V tolerant
Dimensions:	2.438 x 3.294 cm	19.5 x 60.5 x 6.0 mm
Operating Temperature:	-40 to 85°C	-20 to 55°C

Other Options

There are of course other options than the two modules mentioned in this section. A module called *XBee-PRO 868* could be used for the same purpose as the other two. The specifications of the module are listed in Table 3.6 and the unit has almost the same capabilities and interface as the *XBee-PRO S2B*.

Table 3.6: XBee-PRO 868 specifications

	XBee-PRO 868
Price:	\$45
Range:	40 km LOS
Power Consumption:	Transmit: 85 - 500 mA @ 3.3 V
	Receive: 65 mA
Power-down current:	55 μA
Frequency:	868 MHz
Protocol:	XBee - ZigBee based
Tx Power:	1 - 315 mW
Data Rate:	24 Kbps
Antenna:	Wired Whip,
	U.FL connector
	RPSMA connector
Supported Topologies:	Point-to-point
	Point-to-multipoint
	Peer-to-peer
Supply Voltage:	3.0 - 3.6 VDC
Serial Data Interface:	3.3 V CMOS Serial UART (3 and 5 V tolerant)
Dimensions:	2.438 x 3.294 cm
Operating Temperature:	-40 to 85°C

Chapter 4

Search and Rescue

Search and rescue can be defined as the task of searching for and recovering someone who is lost or somehow incapable of getting home by themselves. The final goal of this Thesis is to develop a UAS that can contribute to SAR operations and be a valuable tool for the SAR crew to utilize. In order to design such a system it is important to have some knowledge about how an actual SAR operation is executed. This chapter will briefly explain how a SAR operation or mission is performed and how a UAS can contribute to such a mission.

Because of the limited time frame of this Thesis, the focus area will be on Norwegian wilderness SAR operations and the UAS evaluated will be optimized and discussed according to this. However, the UAS design should not present any challenges if other SAR crews should want to adopt the solution for another usage.

Readers are encouraged to read the pre-study of this Thesis for additional information regarding SAR operations [Gamnes, 2013].

4.1 Normal Search Approach

A SAR mission can roughly be divided into four steps; information retrieval, analysis, planning, and mission execution as illustrated in Figure 4.1. The time-line at the bottom of the figure indicates the time from when the SAR personnel is notified until the various steps occur. The last dotted line is included to illustrate that the process might be repeated at a later stage in the mission if new information is acquired.

A SAR mission generally involves many actors such as volunteers, humanitarian organizations, and some government appointed instances. SAR operations are complex operations and developing new SAR tools are challenging. Nevertheless, having tools that might help a SAR mission is highly desirable and thus the subject should be studied as is done in this Thesis.

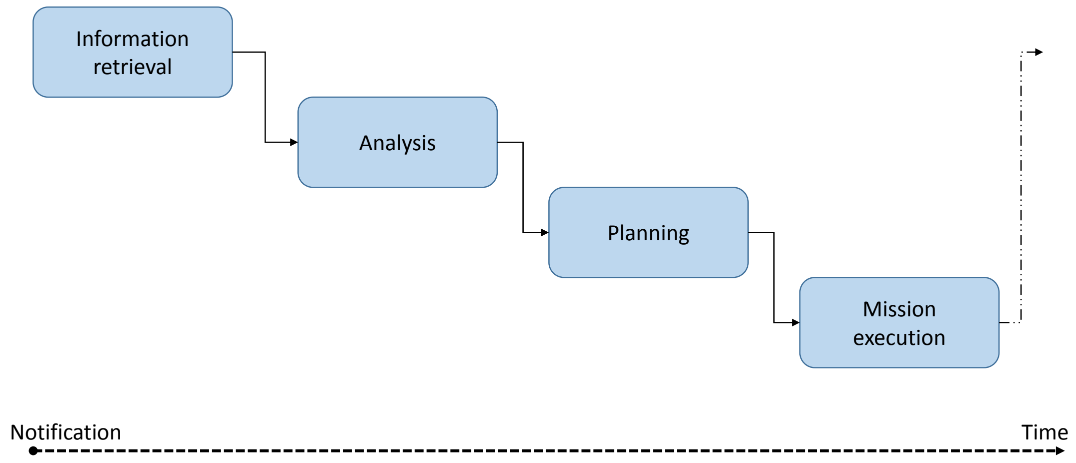


Figure 4.1: SAR process [adapted from Guldbrandsøy et al., 2009]

4.2 UAS Utilization during SAR

A UAS may be utilized during SAR missions for several purposes. It might be used for communication, searching, delivering supplies, surveillance, streaming of real-time footage, and tracking. The possibilities are many and the UAS should be created such that the user can choose which feature they want to utilize and thus be able to adopt the system to better fit their current situation.

Referring to the different steps of a SAR mission as illustrated in Figure 4.1, utilizing a UAS could affect all these steps in different ways. A UAS may be used to gather footage of an area prior to the analysis, i.e. during the information retrieval step. The UAS may be directly involved during the analysis of the collected information. For instance, by using artificial intelligence, a likely map of where a person could be located can be derived and thus help the SAR crew plan their search. The planning stage may be affected by the UAS as it can derive the path for searchers to follow based on optimal area coverage. Helping the SAR crew find the most sensible path to search is a possible usage application of a UAS. Finally, the UAS can be used for many different applications during the mission execution, such as a being a communication relay or autonomous searching unit.

4.2.1 Berry-Picker

In order to illustrate the advantages of utilizing a UAS during SAR operations, an example SAR situation is described below.

An old man is reported missing and is suspected to have gone on a berry-picking trip. A search area *is defined to be roughly 1km^2 around the area where the man is assumed to be located. Volunteers started searching at ten o'clock at night by walking around the search area, and the missing person was found one hour after midnight. He had suffered a stroke and was thus incapable of moving. The man was found 150 meters from a walking track and some hundred meters from a driving road.* [Torkildsen, 2009].

UAS Advantage

In this SAR situation, a UAS could have been used in the execution step of the SAR operation to track down the missing person. The UAS could have been used to give the SAR crew video footage from the air using a thermal camera. Since the situation unfolds in Norway, it is not necessarily dark at nighttime and if that was the case, even a normal color camera could have been used to provide the footage. The UAS could also have been programmed to search autonomously for the missing person and programmed to follow the known hiking paths in the area. Since the missing person was found near a hiking path, it is likely that he would have been spotted by a UAV flying above the paths. The use of a UAS might have saved time and thus gotten the missing person to a hospital faster. This could be essential in the case of a heart attack.

Utilizing the UAV during the planning stages of this mission would have given the SAR personnel a better overview of the area. The UAV could have flown over and be used to create an aerial image map that could be used for planning the mission. If the missing person that is searched for is not spotted in the images, the areas where the view from the UAV was blocked by for instance trees can be searched instead of the open fields where nothing was seen on the images. Even though the SAR crew probably has a map of the search area, during a flood, roads and building might have collapsed making the area different from what the map shows. In such a scenario, having real-time footage might be vital for the mission and could for instance be used to find roads that are still accessible and not blocked.

This example just shows some of the utilization benefits of utilizing a UAS during a SAR operation. There are of course many other scenarios and application examples and the UAS should be developed such that many of these applications can be realized.

4.2.2 UAS features

The features, or attributes, needed by a UAS to be considered well suited for SAR missions are discussed in the pre-study of this thesis [Gamnes, 2013]. These features can be roughly summarized as following:

- Low cost
- Multiple usage possibilities
- Modular design
- Fast response-time
- Long-range
- Long flight-time

- User-friendly
- Environmentally friendly
- Safe
- Different autonomy levels
- Integration into already existing SAR system

Some features might be more important than others, and which are more important will be highly dependent on the different SAR situations. Making the system multi-functional will be important as the system could then be used in several different scenarios.

4.3 Summary

A SAR operation is a complex, well implemented, and working operation that involves many different participants. Some are professional and some are volunteers. Coordinating this operation is definitively a difficult task and introducing tools that will contribute to the missions is a good step forward. However, introducing new tools might also disturb the already working process and thus the tools might not be of use because of this. Integrating the UAS into the already existing SAR system will therefore be a vital step for utilizing such a system and thus play part as an important design criterion during the development.

The features a UAS needs to possess to be considered used during a SAR operation are many and most likely a system cannot contain all of them. Most likely, the UAS will need to be designed in such a way that several of the features are possible to accomplish given a different setup. Low cost can be accomplished by having a modular system where modules can be chosen to be included or not. Costly modules may thus be omitted if desired or not possible to utilize.

Having a UAS that will contribute to SAR missions will be very convenient. The utilization of air-support in today's SAR missions are very costly and when the air-support is only used for visual purposes, there is really no need for a large helicopter capable of transporting people as well. Having a small, easy-to-use UAS available might mean that the large helicopters does not need to be used for this particular SAR mission and money could therefore be saved. The incorporation of UAS into SAR operations will possibly save money and time used during the SAR missions and possibly make the search easier and more efficient. If a UAS can contribute by saving one more person than without, the development of such a system will have been worth it.

Chapter 5

Unmanned Aerial System

Today's technology opens huge possibilities for utilizing unmanned systems. Using autonomous flight controllers and sensor technologies makes it possible to create an aerial system that can act autonomously. This is a huge advantage during operations where human resources are limited as they often are in SAR operations. UAS is a term describing a system containing one or more UAV and its or their associating components such as a GCS, launching ramp, landing equipment, etc.

An application area where using UAS is relevant will be during operations that are tedious, exhausting, or dangerous for humans to perform. There are many applications where the benefits of utilizing UAV is high, as for example utilizing UAS in applications such as firefighting, emergency medical service, and of course SAR operations. An example is the case where a snow avalanche has been triggered by a snowmobile driver and the driver becomes missing. If the area where the situation occurred is still prone to more avalanches, sending in human rescuers might not be an option as it could be too dangerous for the rescuers. In this scenario, having a UAV that can fly in and provide more information about the area will be useful. The UAV can be programmed to scan the entire area and create a mosaic image map that the SAR personnel could use for planning their mission. The UAV could also be programmed to search for certain objects and if any objects are detected, they can be marked on the mosaic map along with their corresponding GPS coordinates, hence facilitating the work of the SAR personnel regarding the localization of these objects and possibly the missing persons.

This chapter will discuss how a UAS can be developed for use during SAR operations and each part of the UAS will be explained separately. The idea is to have the UAS as autonomous as possible as not to occupy the SAR personnel more than necessary.

5.1 System Overview

Figure 5.1 illustrates a system overview of a UAS and contains the most common components of such a system. A component that is not illustrated in the figure is the airframe itself. The most important component of a UAS is the FMU, which provides the system with its own intelligence. The FMU controls the entire flight system and is responsible for handling the communication with sensors, actuators, and the GCS. A separate embedded system is in this Thesis introduced to handle detection tasks during flight, such as computer vision detection. If the main FMU is powerful enough to handle both the detection tasks and the flight controlling tasks, the extra embedded computer might be omitted and this could possibly simplify the system.

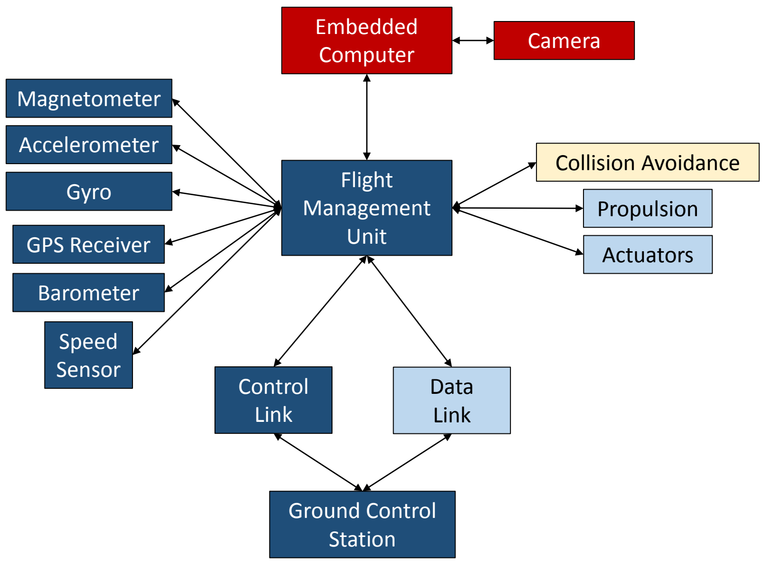


Figure 5.1: UAS Overview

The components with blue background and white writing in Figure 5.1, are all part of the *APM 2.6* system explained in Chapter 3. Since these components and this system does not need to be developed, this Thesis has focused on simply getting the system to work and performing an evaluation of it. The other parts of the system in the illustration, such as the detection part and the communication link between this and the FMU, has been looked into and is presented later in the report. Collision Avoidance will not be investigated in this Thesis, but might have to be part of a complete SAR UAS. The likelihood that the UAV will crash into something during flight is very small and depending on how the UAS is used, the avoidance system might not be needed at all. The data link is present to transmit data from either the FMU or the detection system to the UAS operators. The propulsion and actuators are attached to the airframe and controller by the FMU to enable the UAS to fly.

5.2 Utilization during Search and Rescue

Since UAS technology is a wide field and the usage areas for it are many, the different aspects of the UAS will have to be customized for the particular application it is to be used for. A UAS developed for military object tracking might not necessarily have the same demands as a UAS developed for SAR operations. For example, an object tracking UAS might need the capability of hovering to keep an object in sight if the object stops. This feature is not necessarily needed during SAR operations and thus the two systems do not need to possess the same features or capabilities.

The will and desire of utilizing UAS during SAR operations are well established and is discussed in the pre-study of this Thesis. Utilizing UAS during SAR missions makes it possible to search over large and possibly dangerous areas very quickly compared to when the SAR personnel has to perform this on foot. Having a UAS developed specifically for a SAR scenario will contribute to the operations by providing an efficient and highly usable tool. Some system features needed for utilizing a SAR UAS are listed in Section 4.2.2. However, these might not all be possible to incorporate in the same system and choices have to be made regarding which feature to value more than others are. These choices are better made when some experience regarding the usage is obtained and the UAS should be tested before deciding upon the final design.

5.2.1 System Idea

In order to develop a UAS that could be used during SAR missions, an idea of a working SAR UAS needs to be established. Figure 5.2, which is the same as Figure 1.1, illustrates such an idea and shows what could be a working SAR UAS.

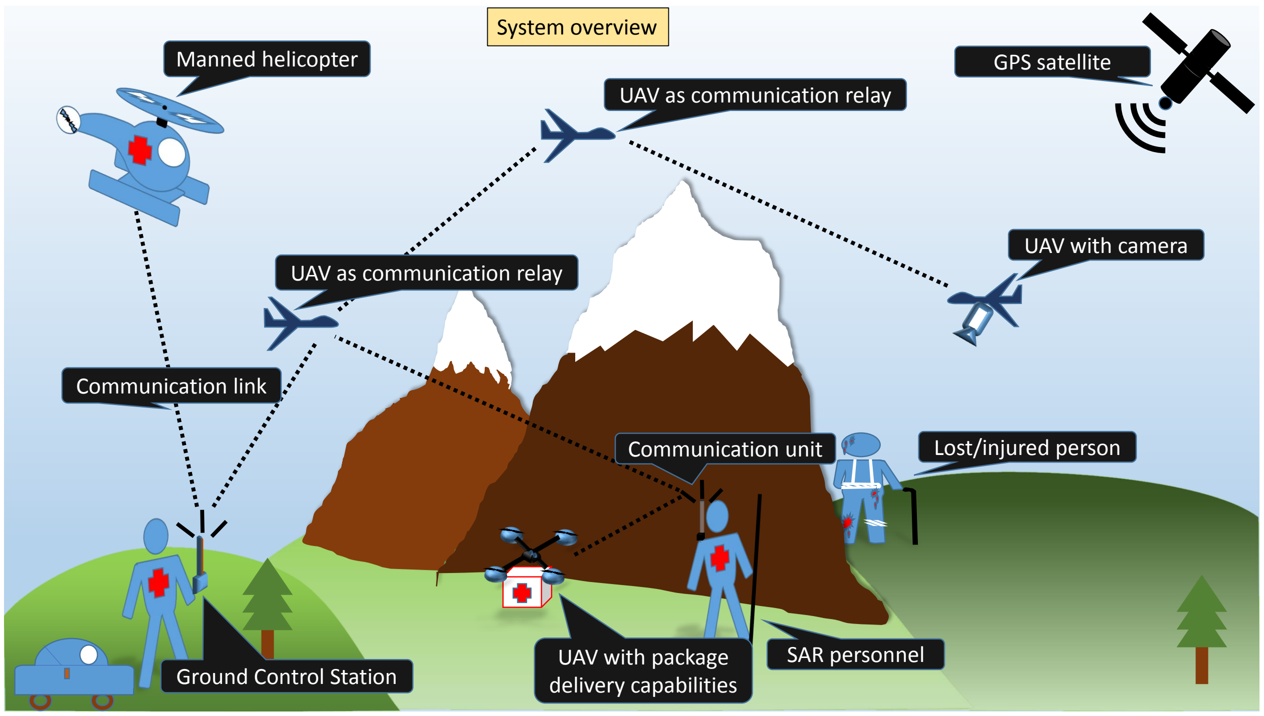


Figure 5.2: UAS utilization during SAR Operation

The idea in Figure 5.2 illustrates a UAS being used for several different purposes during a SAR operation. The illustration contains four UAV where each one can be programmed in its own way. The two UAV that are marked as communication relays are programmed to maintain a communication link between the GCS and the outermost UAV, and adapt their position thereafter. They also provide a communication link between the searcher in the field and the GCS if the LOS is blocked between them. The outermost UAV, which has a camera attached to its frame, is programmed to scan an area and report if any interesting objects are detected. The UAV with a rescue package underneath can be programmed to fly to a location and deliver this package. There are two different kinds of UAV present in the idea, planes and a multi-rotor vehicle.

If a UAS is to be used during a SAR operation, it needs to be incorporated with the already present participants, such as the searchers and manned helicopters. During a SAR mission, the manned helicopter is often used to transport searching personnel to an area, or to recover the missing person if they are located and not capable of getting back on their own. The helicopter might also be used for searching or providing aerial footage from the air during the search. The manned helicopters are a costly resource and they do not have as quick response time as a UAS could have. Having multiple UAV in the air while trying to fly the manned helicopter could possibly be unsafe and is probably not efficient and the design of the UAS will have to adapt because of this. The goal of the UAS is not to replace the manned helicopter or other units of a SAR operation, but rather be an extra or alternative resource for them to use. In the illustration of Figure 5.2, the lost person is hidden from the searchers view and a flying UAV is more likely to find him than the searcher walking around on foot.

The following sections will describe the different parts of a UAS developed specifically for a SAR scenario. There are many options to consider and this Thesis will only evaluate some. As to not need operators controlling the UAS all the time, the autonomy level of the system should be as high as possible.

5.3 SAR UAS Architecture

During the design and development of any system, many decisions has to be made. For example, when choosing which components to use, the components needs to be evaluated based on several criteria. For a UAS, these might be weight, power consumption, functionality, and cost. Choosing expensive components will probably result in lightweight, low power consumption, very functional devices, but the cost might be so high that a system consisting of such components will end up costing too much to be used. Choosing a less costly component might therefore be the better choice.

5.3.1 Features

The architecture of a system will be influenced by the features that the system needs. When designing a SAR UAS, the features listed in Section 4.2.2 will be important to incorporate. Having all of these features in a low cost system might be difficult, but they should all be sought after. The most important features from the list that a SAR UAS should possess might be integration, low cost, user-friendliness, and the possibility of autonomous flight.

If the system is to be used during a SAR mission, it has to be incorporated with the already existing operation. Having a system that will disrupt the work flow of the entire operation is not desirable. The UAS should also be low cost since the organizations and volunteers can not be expected to spend a lot of money on equipment. The system has to be user-friendly such that it does not cause problems for the operator, which would only be a source of frustration and cause time to be wasted. Finally, the UAS should be able to operate autonomously, if that is desired by its operator.

5.3.2 Modularity

When developing any system, it is difficult to know what the best solution is right from the start and having a modular design might be important. During the development and use of a system, different solutions will present themselves and therefore having a modular design makes changing or upgrading system parts easier. For instance, if a detection system is developed and six months later a different, improved detection system is found or developed. Having a modular architecture will make it possible to change the detection module of the system, allowing the better system to be used. Having a modular architecture will allow this and also other upgrades to be performed if necessary. A modular design will also allow a system to be fixed more easily as the parts can be changed individually from each other. If some part of the system is broken, solely this part can be replaced, and it does not have to affect the rest of the system.

Modularity is consequently a key word when discussing UAS architecture. As explained in the pre-study, modularity is important because the UAS technology is rapidly changing and having the opportunity of altering the system is essential for the system to be as good as possible. Since this is a Master Thesis, the possibility of somebody else continuing the work on this UAS is highly likely. Therefore, having a modular design will make the parts easy to replace if desired by the next developer.

5.3.3 Usage

The ultimate goal for a SAR UAS is to have a system that is possible to operate without the need for any formal training. Everything from launching to landing should be performed autonomously by the UAS. The SAR personnel should be able to operate the system perhaps by pushing some buttons and commanding the UAV what to do. Then all other tasks should be managed by the UAS. Having a very simple to use software along with a simple interface on the plane for control purposes, could be optimal for a SAR UAS.

5.3.4 Flight

Having a SAR UAS that autonomously handles everything regarding its flight will be useful for a SAR scenario because of reasons mentioned earlier. The launching of the UAV should be performed autonomously using some kind of launching method, and the flight and landing should also be executed autonomously. The operator should not be troubled with flight details and flight experience should not be needed for utilizing the system. This could be important as SAR personnel does not necessarily know how to operate a UAV, but if the flight is handled autonomously, the UAS can still be used even without this knowledge.

Scanning an Area

The UAS design will determine how the UAV flight is performed during a SAR mission. The UAS can be designed such that the UAV is programmed to scan an area for objects, report its findings, and return to the launch site where it waits (loiters) for the next assignment. This approach is illustrated in Figure 5.3 and shows the defined search area along with the flight path of the UAV. The mission person is also illustrated in the image hidden behind a mountain. In the illustration, the missing person is detected by the UAV and a notification is sent to the GCS, while the UAV continues it search of the region. Commands may be sent to the UAV during its search, like for instance commanding the UAV to fly above the area where the detection took place and search again to verify that there is in fact a person there. If the aircraft is being used for scanning an area, the aircraft might need to fly further then the region it is programmed to search in. This is illustrated in Figure 5.3 by the flight path going a bit further out on the sides of the search region. The reason for flying further is that if the camera is fastened to the UAV, thus pointing in only one direction relative to the UAV frame, the camera will be tilted during turns and thus might not cover the area specified. Having a Gimball mount for the camera would remove the need for doing this as the cameras view can be changed during flight.

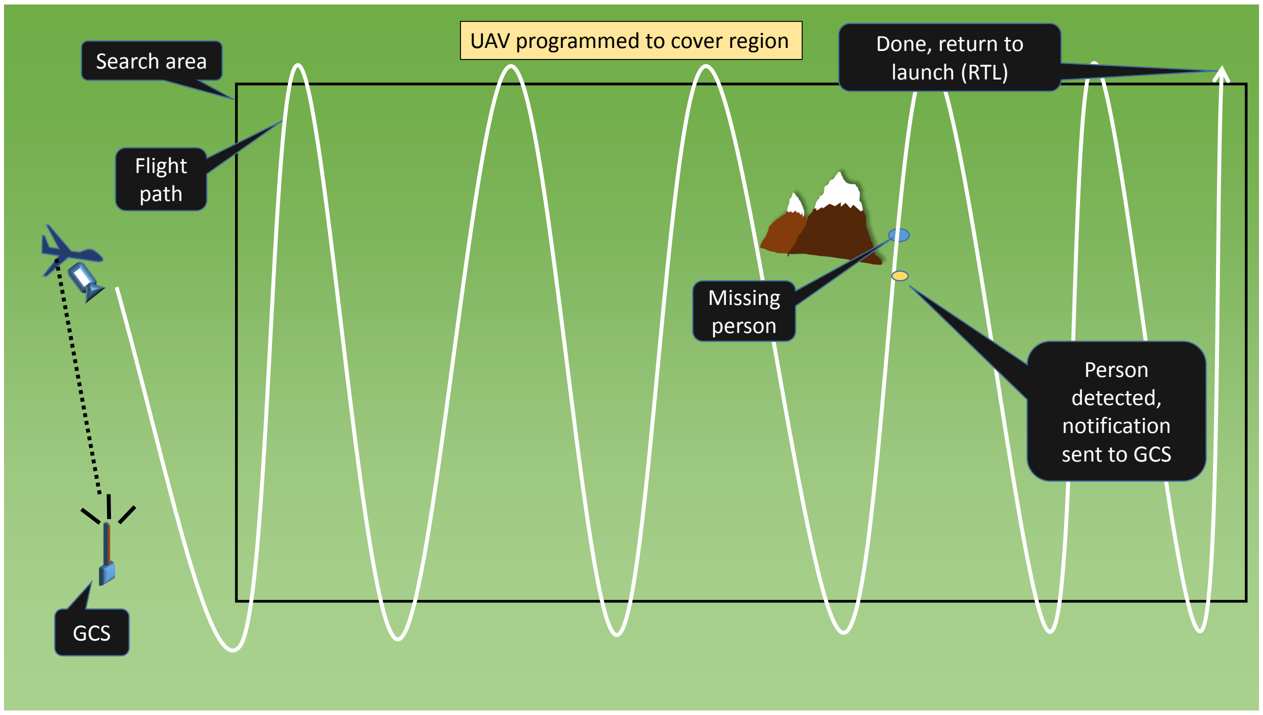

Search area
UAV programmed to cover region
Done, return to launch (RTL)
Flight path
Missing person
Person detected, notification sent to GCS
GCS

Figure 5.3: UAV Flight Path, Scanning an Area

A problem with this approach is that if the missing person moves into an area which is already covered by the UAV and is thus not detected, the search area might be classified as not containing any humans when the human is actually there. This could mislead the SAR personnel to thinking that the missing person is not located in the search area. A solution for this issue if to have the UAV fly and scan overlapping regions of the area, where the overlap could be calculated based on how fast a human could move across the regions already covered.

Following a Searcher

A different approach than letting the UAV scan an area, is to let the UAV follow one of the SAR personnel during their search. The UAV can be programmed to follow the searcher and fly above their head reporting any findings that the searchers might have missed to either a GCS, or the searcher. By utilizing the Wi-Fi connection on a searchers mobile phone, the UAV can keep track of the searcher and plan its flight accordingly, and the Wi-Fi connection could possibly be used for controlling the aircraft. This approach is illustrated in Figure 5.4 and the illustration shows the path of the searcher in black and the flight path of the UAV in white. The entire flight path of the UAV is not included in the illustration.

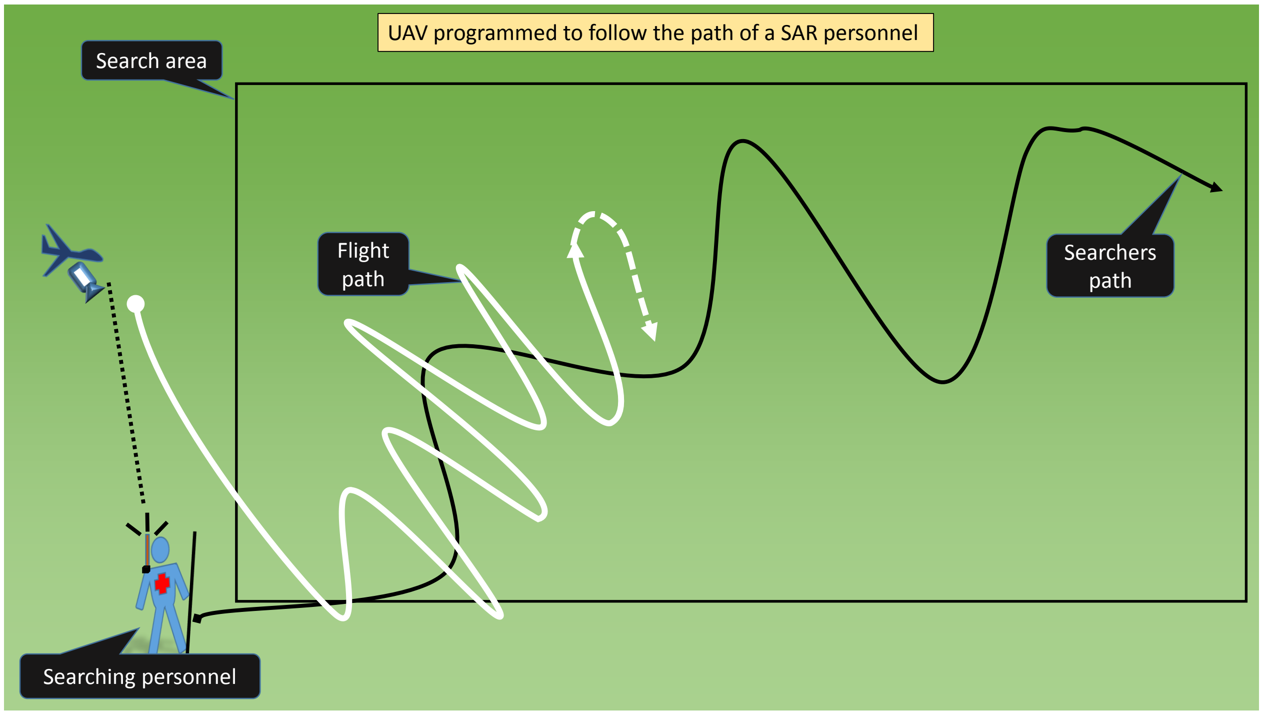


Figure 5.4: UAV Flight Path, Following a Searcher

The flight path of the UAV using this approach is continuously altered during the mission to adapt to the position of the searcher. A limit can be set so that the UAV does not fly farther than for instance 100 m to the sides of the searcher. This could be done as to avoid loosing the communication link connection between them and also keeping the UAV within VLOS with the searcher the entire time, thus following the current UAV regulations in Norway as mentioned in Section 3.1. A problem with this approach is that only the area surrounding the searcher will be covered. This could be solved by making the UAV plan the searchers path in order to get the entire search area scanned more efficiently.

Take-off

A UAV take-off can be performed in multiple ways and which method used will depend on which aircraft that is used. A multi-rotor vehicle can perform what is called Vertical Take-Off and Landing (VTOL) and hence does not need a lot of open space for doing this. A plane on the other hand, needs a runway or a somewhat open area to be able to take-off. A plane with wheels can take-off from a runway like ordinary manned planes do, but if the plane does not have wheels, some other method must be used.

The take-off of a SAR UAS needs to be performed autonomously or with as little involvement as possible from the operator. As shown in the tests later in this report, human errors during take-off are common and SAR personnel cannot be expected to do this perfectly every time. During this Thesis, the most common hobby technique called hand launching is used. The plane is tossed into the air and throttle input is applied, making the plane capable of taking off. Another approach would be to use what is called a bungee-launcher. This technique uses a rail and the plane is fastened to a bungee that is put under tension. Once the plane is ready for take-off, the bungee is released, hence pulling the aircraft forward into the air providing the aircraft with enough speed to accomplish a take-off. A bungee-launcher can be made quite small and handy, and could be used as part of a SAR UAS.

Landing

Landing of a multi-rotor vehicle is usually performed vertically and thus do not require a lot of open space. A RC plane on the other hand, needs either a runway of something to stop the motion of the plane during landing. During this Thesis, a landing method called belly landing will be used. Since the plane used in the tests does not have wheels attached to the airframe, it will be landed on its belly on a somewhat soft surface as to not damage the aircraft. The plane is flown close to the ground and then the engine is turned off, allowing the plane to glide onto the ground and thus stopping. If a plane has wheels, a run-way can be used in order to land. Another approach for landing would be to use a catching net. The aircraft can be flown into the net thus making the plane stop and prevent it from being damaged. This approach is often used on board boats, as there is no runway to land on. A parachute could also be used for landing an UAV, but could possibly cause problems as to where the plane lands because of winds and other environmental factors.

Which landing approach used will depend on the features of the aircraft, and for instance, a heavy aircraft might not be possible to land in a net due to its weight. In a SAR UAS, the landing should be performed as autonomously as possible. If belly landing is used, the operator should specify an area where the UAV should land, and the UAS should plan the flight such that this landing is possible and finally, land on its own.

An issue regarding the landing of a UAV is making sure that it does not hit people in the area around the landing location. When the UAV lands, it needs to land in such a way as to not harm the people which are present in the region surrounding the landing spot. If the UAV is not particularly heavy, using a parachute to land would make it so that no harm is done even if the UAV hits something or someone on the way down, because the speed of the descent will be so slow.

Side-winds

During a SAR operation, the possibility of weather conditions including strong winds is a very likely scenario. Having a plane without any tail control actuator will make flying in side-winds and getting images of the ground difficult. This can be solved by either flying straight towards or away from the wind or by having a Gimball mount for the camera. If neither of these approaches are used, a tail control actuator might be possible to use in order to correct the flight given the wind direction.

5.4 Aircraft

The type of aircraft that is best suited for a SAR scenario will be dependent on the specific SAR situation at hand. Because SAR missions are so intricate, designing the UAS such that utilizing different kinds of aircrafts is a possibility, is probably a good thing. The UAS should be designed such that planes as well as multi-rotor vehicles, or possibly even rovers can be used. The users should have the possibility of selecting the aircraft they feel is best suited for their particular scenario and having no troubles doing so.

5.4.1 Airframe

There are essentially two different types of airframe to choose from; a plane, or a multi-rotor vehicle. There are good and bad sides of both types, and the choice between them for use in a SAR UAS will as explained depend on the application the UAS is used for. A UAV used during SAR operations will most likely have to carry some sort of payload like a sensor, and will have to be able to support the weight of this load and still be able to fly. The choice of material for the airframe is also important and will have to be chosen based on integrity, weight, price, and availability [Hammerseth, 2013]. During prototyping, the need for a frame that can be modified and easily fixed is needed. Hence, using a foam-based material is a good option since foam usually can be glued back together if broken. The airframe should be designed so that it is easy to transport. For instance, making the UAV fit in the back of a car, by either having a frame that can be dismantled or simply being small enough to fit, will be advantageous during SAR missions as the most likely transportation method of the UAV to the search location during a SAR operation is by using a car.

Any equipment used in SAR operations needs to be robust, and might need to be able to perform in cold temperatures. Weather conditions in Norway are harsh and the temperature may fall below minus 50 degrees Celsius, however this is not common and minus 25 is a more realistic temperature level. Creating a system that can operate at such low temperatures will be very costly, and an evaluation would have to be made regarding cost and utility in order to determine how much can be spent on components for a SAR UAS. Perhaps making a UAS that cannot be used in the most extreme weather conditions is a possible solution.

When designing and developing a UAS or any system, it is always good to get ideas from others and adapt a solution from what is known to be working. In Chapter 2, a system called *Trimble* was mentioned [Trimble]. This is a UAS for capturing aerial footage and works autonomously. The airframe used is a flying wing that can withstand strong winds and "Norwegian" conditions. Adapting the solution of this UAS and their approach could be smart, as their system is known to be working. The propeller is created such that it folds in during landing so it does not break, and the frame is created to provide some protection to the motor during landing. The aircraft is launched using a bungee-launcher, which is as mentioned a possible solution for a SAR UAS as well.

Planes vs. Multi-Rotor Vehicles

There are of course positive and negatives sides with using both planes and helicopters, and which aircraft that is best to use, will as mentioned strongly depend on the particular SAR situation. Planes are better to use when there is more wind, longer flight distance is required, and if an area needs to be covered as quickly as possible. Planes normally fly faster than helicopter and uses less energy to stay aloft by gliding on the wind, hence giving it longer flight distance and flight time. Plane are often more robust regarding wind conditions and will handle harsh flight conditions better than multi-rotor vehicles.

If a plane is used, the frame could be wrapped in some kind of wrapping to protect the equipment and to strengthen the frame. If the wrapping is see-through, a manual could be placed underneath so that the operator always has it at hand and the manual would be protected from damages.

Multi-rotor vehicles should be used when VTOL, or hovering capabilities are needed. A multi-rotor vehicle can hover above a certain location and enter smaller areas such as caves, and will situations where hovering or entering of caves are needed be more useful than planes, as planes always have to keep moving to stay aloft. There exists some solutions combining a plane with multi-rotor capabilities, which might be a solution to use in a SAR scenario, but that is not investigated in this Thesis. Delivering supplies or first-aid kits from a multi-rotor vehicle is probably easier than delivering the same supplies from a plane. Thus, using a multi-rotor vehicle will be more advantageous to use in scenarios where the delivery of packages are necessary. A multi-rotor vehicle uses a lot of energy just to keep itself flying and will therefore have less flight-time and flight-distance than planes.

5.4.2 Energy Source

Two energy sources that can be used for a UAV are batteries and liquid fuel. The safest choice is probably batteries, which is also the most common choice for RC equipment nowadays. Battery technology is always progressing and the batteries are getting smaller and better every day.

Fuel or gas powered aircrafts makes more noise than battery driven ones and this might actually cause problems because they are not allowed to fly in certain areas due to the noise they produce. However, this noise might be desired in a SAR scenario as to notify the missing persons about the whereabouts of the UAV. This could perhaps make them signal towards the UAV, thus making it easier to detect them using a detection system. An advantage of using fuel is that the plane can be refueled in seconds when empty. A battery will use some time to get recharged and electricity might not be a common resource if the SAR situation unfolds far out in the woods. However, if the aircraft has multiple sets of batteries, one can simply swap them and this would take just as little time as refueling the fuel-powered unit. Another thing to note is that fuel-powered planes creates more vibrations than electrical driven ones. The navigation system of the autopilot might actually be disturbed by the vibrations of the engine and thus not function properly. A disadvantage using fuel-powered aircrafts is the maintenance involved with the engines. As with cars, the engine requires maintenance such as checking the oil and this is something battery powered aircrafts do not have. Having a system that is easily maintained is advantageous since the SAR personnel usually have other jobs when not performing the rescue missions. They might not have the time or skills to perform the maintenance needed by a fuel-powered aircraft and thus choosing a battery system is probably a better option for SAR UAS.

Batteries are good to use because they are small, light, safe, and rechargeable. Planes using batteries as its energy source are more environmentally friendly than fuel-powered planes, as they can be charged with electricity gotten from renewable energy sources. However, batteries do not function very well in the cold and this was experienced the hard way during one of the tests of this Thesis. During a test that was going to be performed at a cabin in wintertime, a RC helicopter was brought to get some aerial images of a "missing" person in the woods. The temperature was below zero and the batteries were stored in an aluminum casing which did not provide any temperature protection for the batteries. The end scenario was that the batteries were unable to provide the helicopter with enough power during take-off because they were simply too cold. A fuel driven helicopter might not have this problem, but the problem could have been avoided if the batteries were kept warm during transportation.

5.4.3 Propulsion

During the UAV flights in a SAR scenario, it is important to keep the energy usage as low as possible. Having to land the UAV often during a SAR mission in order to charge or change batteries will be time consuming and expose the system and its users to more danger than necessary. The propulsions system used should therefore be optimized according to the airframe and the payload it needs to carry. Since the complete UAS is not decided during prototyping, the choice of engine should be evaluated at a later stage of the development.

5.4.4 Actuators

The actuators are the units that perform the control operations on the UAV. The actuators on a plane are the units that control the parts of the wings, usually called flaps or elevons, which makes the plane go in different directions during flight. When discussing RC equipment, the actuators are often servos placed on the flaps of the wings. These servos needs to be robust enough so that they do not break during flight or landing, and will also have to handle the pressure they need to control and move as fast as they are needed to move. Normal RC servos will probably be sufficient for a SAR UAS. The temperature ratings of the components might also be important to take into consideration, as the system should be able to operate in Norwegian conditions.

5.5 Autopilot

The autopilot is responsible for controlling the aircraft and managing everything regarding the aircrafts flight, including take-off, landing, and normal flight. The users interaction with the autopilot should be in forms of executing commands, for example ordering the aircraft to fly to a GPS location and wait (loiter) for further notice. Another command might be to scan an area for humans or objects and report the findings when completed. Some more information regarding autopilots can be found in Section 3.2.

Choosing the best-suited autopilot for an application can be difficult and often there are several good options. An autopilot that is going to be used during SAR operations should have the alternative of manual control as well as performing missions autonomously. If the UAV can perform the mission by itself, it will relinquish the SAR personnel to perform other, perhaps more important tasks. The autopilot has to be relatively cheap and should be possible to use for multiple airframes. Its design should be modular such that components may be changed if desired or needed. Of course, the autopilot needs to be safe in the sense that safe actions should be available. A safe action is an action performed after something has gone wrong. For example, if the communication link between the UAV and GCS is lost, a safe action would be that the UAV flies back to where it began its flight and waits for the communication link to be re-established.

5.5.1 Attitude Heading Reference System

In order for an autopilot to function, it needs to know its attitude, i.e. orientation in 3D-space. An Attitude and Heading Reference System (AHRS) will provide the autopilot with information regarding the aircrafts attitude and compass heading. By using this information, the autopilot can control the aircraft. An AHRS can be constructed in several ways, but usually consists of microelectromechanical systems (MEMS) containing gyroscopes, accelerometers, and magnetometers on all three axis visualized in Figure 5.5; yaw, pitch, and roll. MEMS tend to be inaccurate and requires filters to get a precise reading, but they are usually lightweight, cheap, and does not use a lot of power, making them well suited for use as part of an autopilot in a low cost UAS.

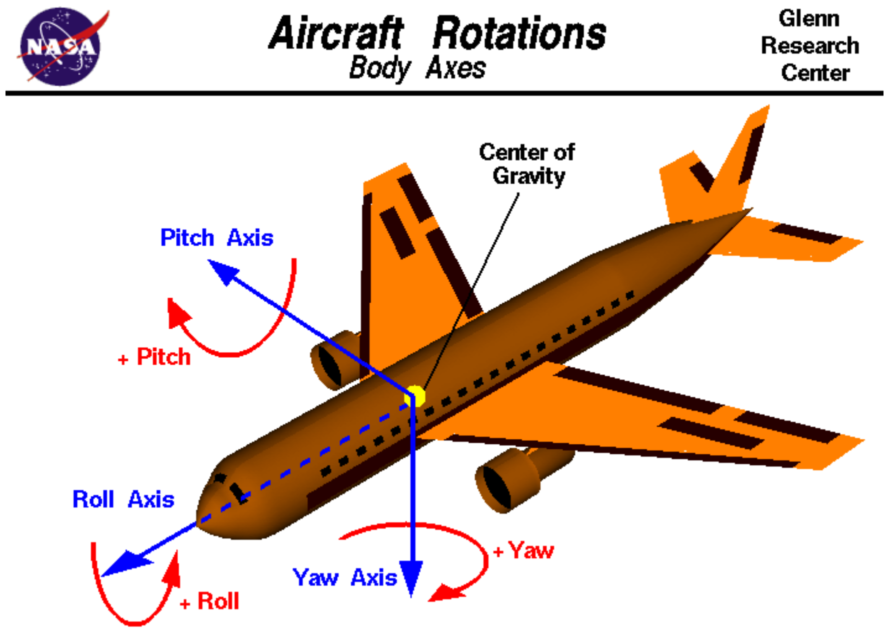


Figure 5.5: Yaw, Pitch, and Roll [NASA]

5.5.2 Navigation

An autopilot needs to be able to navigate the aircraft in the area it is operating. An operator should be able to tell the aircraft to fly above a specific area, and the UAV will thus have to navigate accordingly. The most common navigation method is accomplished by using GPS, which is mentioned in Section 3.3, and the user can then simply provide the UAV with GPS waypoints that the UAV should follow. It is possible to append other information to GPS waypoints such as speed and angle of flight [Fossen, 2011]. This will be useful if the UAV needs to get footage underneath the branches of a tree, which cannot be accomplished from directly above the tree, but would have to be captured from an angle.

GPS navigation is a simple and well-known system, but as most systems it has its limitations. For instance, a GPS receiver located on a UAV needs to be connected with at least four GPS satellites to get a precise position reading. If the weather is bad or the UAV is located in certain parts of the world, it could be that the GPS receiver does this connection and thus it will not be able to tell its own position accurately. This would result in that the autopilot would not be able to navigate properly. A combined approach using GPS and Inertial Navigation System (INS) can be used to provide additional safety to a navigation system. An INS utilizes accelerometers and gyros to calculate/estimate the position of the aircraft without the need for external references. If a flight is being performed in a valley where the GPS signal could possibly be lost, the INS can be used to estimate the planes position until the GPS connection is re-established, thus making navigation possible even without a GPS signal. Spoofing is another issue regarding the use of GPS [GPS World]. This was mentioned in the pre-study and the problem can be solved by include a INS in the system.

5.6 Communication

The communication capabilities of a SAR UAS are essential for both control and notification purposes. If a UAV is used for searching for objects and makes a detection, it needs to have the capability of notifying its operators of the detection and perhaps the location of the detected object. The operators also needs to have the option of telling the UAV what to do, hence control communication is necessary. The used communication technology needs to provide a high enough bandwidth, and long enough range for a UAS to function properly. How much, and how long this is depends on the scenario and application the UAS is used in and for. If the UAS is used as an autonomous detection system, the bandwidth does not need to be that high as the only message necessary to transmit, is a message telling the operator that a person has been located and giving the position of that person. If the UAS is used for providing video feedback to its operators, the bandwidth demands are much higher. It could be necessary to utilize several different communication technologies on board the UAV and use the one with highest bandwidth whenever possible. If the SAR crew wants an image confirmation of the person that is detected by the UAV and the communication link is to slow to be used for transmitting this image, the UAV could circle back over the GCS and transmit the image using a high-bandwidth communication link.

In a SAR UAS, a communication link needs to be present between the sensors and the autopilot on board the UAV, and as mentioned, a communication link needs to be present between the operators and the UAV itself. The communication between the UAV and operators can be accomplished in many ways and the most common way is to use RF communication. There are many options within RF communication and for instance, using the mobile phone, GSM network, would be an option to use for SAR UAS. The communication between the components on board the UAV will not be evaluated in depth during this Thesis, but will be accomplished using serial communication.

When discussing UAS communication there are two communication links in interest; the control communication link and the data communication link. These two links could be realized as one, but a reason for having two separate communication links is that it might be necessary in order to obtain the required bandwidth and range for each of them. In addition, having two communication links provides the safety of redundancy if one link is lost.

5.6.1 Control Link

The control communication link is where all commands and information regarding the flight are sent between the UAV and its operators. This is the link where all information needed to control and monitor the aircraft is transmitted to and from the operators. Flight commands are transmitted via the control link and the range is the most important feature of a control link as the data amount sent over this link is not necessarily that high.

5.6.2 Data Link

The data link is the link handling all data messages that are sent to and from the UAV. This could for instance be video, images, and location of objects. All data that is not for control purposes is usually handled by the data link. The bandwidth demands of a data link are higher than that of the control link. The range of the data link, depending on what the UAS is used for, does not need to be as long as for the control link. If images are needed to be transmitted from the UAV to the GCS, the UAV could circle back towards the GCS and when close, transmit the images. Thus the range of the data link does not necessarily need to cover the entire search region.

5.6.3 Methods

There are several ways of accomplishing the data and control communication links in a SAR UAS. Perhaps the most obvious choice is to utilize the cellular, GSM network. This is a well-established network in Norway, but unfortunately, since Norway is a relatively large country and some areas are somewhat remote, the GSM coverage might not be present in all the areas that a SAR UAS would have to be used. Therefore, basing the UAS solely on the GSM network will not be an option. Of course, having the opportunity of utilizing GSM will not be a disadvantage and could mean that the UAS can be used differently depending on the location of the SAR operation. For example, in areas where there is no GSM coverage, the UAV has to follow a searcher a fly above their heads.

The key challenges for UAS communication is the long communication range needed and the fact that the UAV are moving at very high speeds [Jain and Templin, 2012]. The communication method used will possibly also need to be integrated with communication systems of commercially manned aircrafts, as they could end up sharing the same airspace and thus needing to know of each other to prevent collisions. Creating a local ad-hoc network is solution that can be used during SAR operations. The authorities could make sure the airspace around a SAR operation is cleared of other airplanes and thus the need for incorporating the communications systems with manned aircrafts might not be needed. Establishing a local network that can be used anywhere will be advantageous during SAR operations and could make it possible to utilize the system in any location.

Regarding the topology of the communication network of a SAR UAS, a combined approach using point-to-multipoint and mesh network could be a good solution, see more information on network topologies in Section 3.8.2. Having the capability of using a network that allows mesh topology to be used, will allow the UAS to be used in areas where valleys and hills block the LOS from the GCS to some parts of the search region. By utilizing several UAV, communication can be maintained even though there is not a direct LOS between the UAV and GCS. Using a mesh network could extend the range of the communication link, thus allowing a lower range communication system to be used and possibly save costs.

5.6.4 Issues

When discussing UAS communication, a distinction can be made between VLOS and BLOS communication. VLOS describes the scenarios where the operator always has visual contact with the aircraft. This will of course set limitations to range and obstacles in between the operator and the UAV. Depending on how the UAS is designed, it could be that the UAV does not need to have a communication link with its operator present at all times during a SAR operation. If the link is lost, flying back to where the link was last present is a way of re-establishing the connection. This would make it possible to fly around mountain-tops and in valleys during a SAR mission without the need for a BLOS communication system. As mentioned, mesh networking could also be used to maintain the communication link beyond LOS when a VLOS communication system is used. BLOS describes the scenario where the operator does not necessarily see the UAV during the entire flight. This will increase the achievable range of the flight and make flying behind obstacles possible even when not utilizing mesh networking. BLOS would probably be needed in most SAR scenarios as the search regions can be large and the terrain might make it so that some areas are not visual from the GCS. As mentioned in Section 3.1, there are currently regulations that prohibits the use of BLOS UAV operations in Norway. This will have to be allowed before a system could take advantage of a BLOS communication system.

When using radio signals, blocked or distorted signals due to obstacles in the environment is an issue. A mountain might for instance stop the radio signals that travels in the direction of the UAV, resulting in the loss of the communication link between the UAV and GCS. A radio signal can also get distorted resulting in the loss of the information within the signal. A communication system will need to be robust enough to handle these issues and provide a safe and secure link between the operator and UAV in order to be used as part of a SAR UAS.

A search region of 1km^2 was mentioned in the SAR example in Section 4.2.1. This is a relatively small search region and the regions can be much larger than this. A tend-off between range and data throughput will have to be made when deciding which communication system to use. Usually, the longer the range of a communication system, the lower the data throughput. Having a system that can choose which communication system to utilize based on the distance to its receiver would be advantageous. For instance, as mentioned earlier if the UAV needs to transmit some pictures of an area, it could fly back towards the GCS and transmit the images once a better, higher bandwidth communication link is possible to utilize.

5.7 Sensors

Depending on the application a SAR UAS is used for, different sensors can be utilized and mounted on the UAV. Sensor technology can be so much and during SAR operations, the most obvious choice would be sensors that allow the detection of humans, or human objects possible. Other sensors that could be used in a SAR scenario are light, smoke, fire, or radio signal sensors. Having the opportunity of choosing the sensor system that best fits the needs of a specific SAR situation would be advantageous.

Cameras are perhaps the most obvious option when thinking of human detection, because images or vision is one of the main ways humans detect objects. A camera detection system can be used for autonomously searching for people on the ground beneath a UAV, or the image stream could be transferred back to a GCS and be analyzed by the UAS operators. An approach using thermal cameras can be used for detecting humans as the human heat signature usually stands out from the environment they are located in [Rudol and Doherty, 2008].

The sensors used in a SAR UAS have to light enough so that the UAV is capable of supporting their weight during flight. The sensors also have to use as little power as possible as to not reduce the flight-time of the UAV, since the UAV probably needs to provide the power to these sensors. The temperature ratings of the sensors might also play a vital part in the choice of which sensor to use as they will have to function in SAR weather conditions.

5.8 Ground Control Station

The GCS of a UAS is the interface between the operator and UAV. In a SAR UAS, the operator should be able to give commands to the UAV pre-, and during flight using the GCS. Waypoints and flight modes should be programmable from the GCS and it should provide the possibility of in-flight sensor reading so that the operator has all the flight information available. Having the opportunity of controlling several UAV from one GCS will be advantageous during SAR operations, as utilizing multiple UAV could possible speed-up the search.

The most important feature of a GCS is that it needs to be user-friendly. If the system is hard to use and not intuitive, the SAR members might be forced to use time on fixing problems with the system instead of doing other more important tasks. User-friendliness is therefore very important to keep in mind when designing the system. For instance, having a clear separation between the configuration and the actual mission execution is very important as to not confuse the user more than necessary. The configuration is normally not done as often as the mission execution and it is therefore good to have a separate interface for it.

Making it possible to operate the GCS with as little training as possible, will be important for a SAR GCS. Training of personnel is costly and in many cases not possible to accomplish. The SAR personnel are people with other jobs and they do not have the time to learn and train as much using and learning a system. Thus, having an intuitive, easy to use application for interfacing with the UAV is important. Point-and-click configuration of missions will be desired as this is a very intuitive approach for planning a mission, and most persons familiar with a computer will quickly understand how to use such a system. Touch screens nowadays are also very common and having a touch screen interface could make it very easy to understand how to plan a flight mission for the UAS.

5.8.1 Maps

When planning a SAR operation, maps are essential for determining in which areas to search. Having a GCS that incorporates a map into the planning and operation stage of the flight mission will thus be important. Many map services are freely available online and some can even be downloaded such that offline view is possible. If a good enough free-map service is not possible to obtain, the purchase of such a service might have to be considered for the GCS of a SAR UAS.

5.9 Summary and Discussion

This chapter has explained each part of a SAR UAS, and discussed the features and capabilities that each part needs to possess in order to have a complete and working system. The type of aircraft used will depend on the each particular SAR situation and the energy source, propulsion, and actuators will depend on the design of the finalized SAR UAS. The autopilot should be able to perform autonomous flight and should be as easy to use as possible. The autopilot needs to be capable of navigating in a specified area and the most common method to enable this is to utilize GPS. The communication links for the UAS are called the data and control link, and each link has its own requirements. The data link probably needs a higher bandwidth and could possibly have a shorter range than the control link. The control link needs long range, but possibly not as high bandwidth as the data link depending on how to UAS is designed. The higher level of autonomy which the system possesses, the less range is needed by the communication system. There are many types of sensors that can be used as part of a SAR UAS and the choice of which to use will depend on the application the UAS is intended to be used for. During SAR operations, the most obvious sensor to use would be a sensor that can detect humans. The GCS of a UAS needs to provide the user with mission planning capabilities, and it should be very user-friendly. The GCS should also be able to control multiple UAV at the same time if that is desired.

Two methods were introduced as to how a UAS could be used during SAR operations. One method was to allow the UAS to scan a region and report if anything was detected. The other method was to allow the UAV to follow a searcher and fly above their heads, reporting the findings to either the searcher or a GCS. If the UAV is programmed to follow a SAR searcher, it will probably stay within VLOS during the entire flight, thus not violating any of the current laws and regulations. If the UAV is programmed to scan a region, the UAV will most likely fly BLOS from the GCS in order to scan the entire region. Thus, the design of the UAS and how it is intended to be used will have to be taken into account when using the UAS, and perhaps the operators needs to apply for a permission to fly the UAV BLOS in order to utilize the full potential of the SAR UAS.

The laws and regulations regarding UAS usage in Norway are touched upon in Section 3.1. By releasing a new regulation regarding footage taken from a UAV, the government proves their willingness to incorporate UAS into today's society. It also amplifies the assumption that the technology is up and coming, and will thus be enabled for use in more applications than previously. Having the capability of getting aerial footage is the most common perception of utilizing UAV, but they can be used for so much more than that. Many commercial solutions utilizing UAS has been mentioned in the media recently amplifying the assumption that the usage areas for UAS are many. The usage of UAS for delivering pizza or books are examples of this. As explained earlier, developing a SAR UAS will make it easier to develop UAS for use in other applications; hence, the SAR UAS development will not only benefit the SAR community.

Chapter 6

Object Recognition System

The possibilities UAS applications in SAR operations are as mentioned many. Perhaps the most obvious application is to utilize the UAS as an object recognition system, or more particularly, a system for detecting humans. The basic concept behind SAR is to locate and retrieve persons, thus having an autonomous system that can detect them will be highly desired.

By having an object recognition system on board a UAV makes it possible to program the UAV to fly above a desired region and autonomously search for objects in that region. When an object is discovered, the UAV can notify its operators with the location of that object, and send back an image for further analysis if desired. The reason for doing the detection autonomously is to free the operators to handle other tasks that are necessary and perhaps more important to do. The UAV can also be used to feed a video stream back to a GCS where an operator would have to interpret the images and search for the objects themselves, but that would require a massive communication link with high bandwidth and long range, and would also occupy an operator in the process. By doing the detection autonomously, the bandwidth of the communication link between the UAV and GCS does not have to be high. All that is needed to be transferred is the location of the objects that are detected. In addition, the communication link range does not need to cover the entire search area, because the UAV can fly on its own and when a discovery is made, return to where the communication link was last present and notify the operators from there.

As the main goal of the UAS elaborated in this Thesis is UAS utilization during SAR operations, and also assuming the SAR operations are only taking place outside cities, i.e. in the wilderness, this chapter will present and test a wilderness human recognition system for use on board a UAV.

6.1 Human detection

Human detection is a difficult task to perform, and especially when performed from the air. In a SAR UAS, several techniques involving thermal cameras, pure camera vision, mobile phone tracking, human speaking frequency detection, or light tracking can be used for detecting humans on the ground beneath a UAV [Hammerseth, 2013]. This section will describe some detection techniques that are relevant for the SAR scenario in more detail.

6.1.1 Computer Vision

Using a computer for interpreting images is a very intuitive approach for detecting objects as vision if one of the main ways humans recognizes objects. An object can be detected in an image based on many feature such as shape or texture. In Figure 6.1, a train and frog is detected in an occluded environment using what is called SIFT features [Szeliski, 2011]. Even when the object is not entirely visible, it is possible to detect it using this method. There are two trains in the image that are detected and marked with the color green and yellow in the illustration to the right in Figure 6.1. A frog is detected and marked with the color red in the same illustration. Images analysis is a processing extensive task and doing it in real-time puts high demands on the system performing the analysis. The example in Figure 6.1 would probably not be able to perform in real-time on an embedded computer.

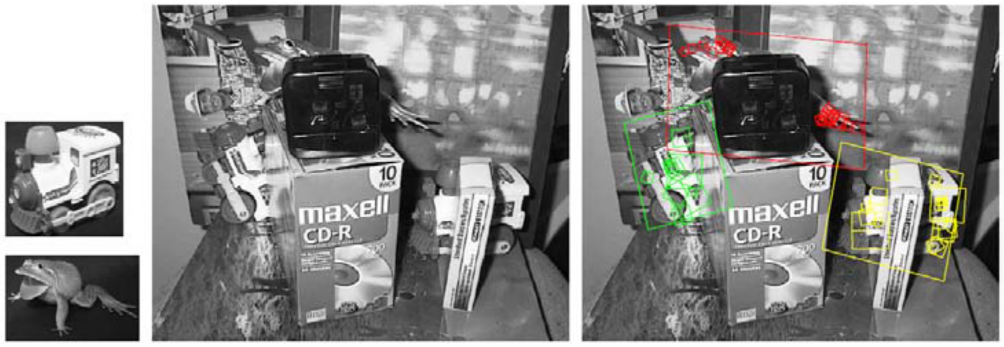


Figure 6.1: Object detection [Szeliski, 2011]

The image processing performed during SAR operations will most likely need to be performed on board the UAV itself because of the bandwidth and range issues regarding the communication systems that are possible to use in a SAR UAS [Gamnes, 2013]. The computer executing the image processing might not meet the demands needed to perform the desired processing in real-time, and therefore using techniques and writing efficient code to save computation time is very relevant for the SAR scenario. A system called *KybMo* was created to fit the precise needs of a computer vision system used on board a UAV [Mohammed Ali Koteich, 2011]. This particular system has not been tested and can thus not be evaluated. However, computer vision tasks might be too hard even for this system to function in real-time and hence, computational saving methods could still be needed.

6.1.2 Camera Alternatives

When discussing the use of computer vision for detection purposes, the camera itself has to be evaluated as well as the system doing the processing. There are many types of cameras and several can be used for the same purpose. The two types of cameras evaluated in this report are thermal and color cameras.

Before an image is sent from the camera to its recipient, some sort of compression is usually performed by the camera to reduce the size of the image. Reducing the size allows the image to be transferred in a reasonable time over the connection to its recipient, usually a USB host. The transfer time is usually limited by the bandwidth of the connection and thus having a smaller image size will reduce the time it takes for the transfer to be completed. When processing an image, a compression might remove features that are necessary for the detection to be successful and it might be difficult to know in advanced which features a camera removes. Consequently, it might be important to know what kind of compression the camera is performing and maybe choose a camera based on which compression is better for the particular system or detection scenario.

Thermal camera

Thermal cameras are passive sensors that detects infrared radiation emitted by all objects that has a temperature above absolute zero [Gade and Moeslund, 2014]. Originally, thermal cameras were developed for surveillance and as a military night vision tool. Common usage of thermal cameras has been limited due to the costs of such a camera, but as technology has progressed, thermal cameras are becoming better and cheaper. As a result, this enables the technology to be used in more applications and thus the technology is used more commonly and not only by the military. Application areas for thermal cameras includes detection of animals, building heat analysis, gas detection, and human recognition.

Utilizing thermal cameras is most likely the best solution for detecting humans from a UAV. Using a passive sensor excludes the need for any external energy source, like the sun or artificial lighting, for detecting objects. This feature enables finding objects in complete darkness. However, there are still issues using thermal cameras that needs to be addressed before it can be used for detecting humans during SAR operations. In the summer, the surroundings of an object might be warmer, or equally as warm, as the object one is trying to detect, thus making the detection difficult [Jo et al., 2013]. In addition, occlusions of clothing might introduce difficulties during the detection as the cloths might hide the heat signature of the person and consequently, the features that are used to detect humans might be hid from the view of the camera.

Even though thermal camera technology is developing, resulting in better and cheaper cameras, they are still expensive. A thermal camera that could be used for detecting humans during SAR operations, called *ThermalEye-3600AS miniature infrared camera*, has a price between $2 707 - $3 795 depending on the features of the camera [Thermal-Eye 3600AS]. The resolution of this thermal camera is a modest 640 x 480 pixels. Using a high-resolution thermal camera for SAR applications is therefore somewhat unlikely, at least in the near future.

Because the resolution of a thermal camera that is used in a SAR UAS is likely to be low, the resolution might not provide enough information to determine if an object is a human or not. Taking advantage of a solution combining both thermal and color cameras is possible if this is the situation. Using the thermal camera for detecting heat blobs (collections of heat signatures in the thermal image) and afterwards processing the same area in the color image for humans is a possible solution to this problem [Rudol and Doherty, 2008]. Once a heat blob has been detected in the thermal image, the corresponding region in the color image can be analyzed, thus speeding up the detection process because the entire color image does not have to be processed for human features, only the blob region.

Color Image Camera

Color images are digital maps consisting of pixels containing color information of the points in the image. These pixels maps the light from the world into discrete values of some color space representation such as RGB or HSV [Szeliski, 2011]. Using the RGB format, which is also the standard in the open-source computer vision library OpenCV, any pixel in the image contains three values, one for R, one for G, and one for B. R represents the red value of this pixel, G represents the green value, and B represents the blue value of this pixel. The HSV, hue, saturation, and value, color space can also be used for image processing and might provide a better or worse result than using RGB depending on the application. The RGB color space will be used during this Thesis.

Using color images for detecting humans is a non-trivial task. Normally this approach requires a lot of processing power and is therefore hard to perform in real-time. This becomes even more challenging when the processing is to be executed on a small, embedded computer on board a UAV. Utilizing the combined solution that was mentioned with thermal and color cameras will contribute to lowering the processing time of the detection application and making a real-time application possible to function even on an embedded computer. Obviously, processing a smaller part of an image will take less time than doing the same processing on the entire image.

Since color image cameras are more common and cheaper than thermal cameras, having a color camera with higher resolution than a thermal camera is a lot less costly and thus more plausible. Using a color camera that has a higher resolution than the thermal camera for the final detection could yield a better result because more features will be visible in the color image.

Video vs. Still Images

Whether the human detection should be performed on a video stream or on still images is a question related to which detection approach that is used. Detecting objects in videos are often done by examining a stream of images for movement and thus be able to recognize objects based on the movement that object does from frame to frame [Ogale]. The idea is to separate the objects from the background by classifying everything that is not moving as the background. This type of detection is often used when the camera is not moving, which makes it easier to separate the background from the actual objects that enter the frame. It is also possible to use this approach even if the camera is moving, but that requires more processing to be performed. It is not easy to subtract the background when the camera is mounted on a UAV that flies around in the air constantly changing the view of the camera. In addition, during SAR operations, the missing person might be located in an area that does not allow them to move around, or they might be incapable of moving, hence they will possibly not be possible to separate from the background and thus not detected in the video stream if such an approach is used. Doing the detection on still images might be a better approach in the SAR scenario, since detection on still images does not depend on an image stream and will thus not have any problems with a moving camera or objects that does not move around.

6.1.3 Tracking Radio Signals

Another approach for detecting humans in a SAR scenario is by tracking radio signals, such as those emitted by missing person's mobile phone. Almost everyone in today's society has a mobile phone and most people bring it along when they go hiking. Having a mobile phone signal receiver on board a UAV might be a solution for discovering where a human could be located. Even if the area where the missing person is located does not have mobile phone reception, their mobile phone is still probably turned on and still transmitting radio signals, possibly Wi-Fi signals, which could be intercepted by a UAV [Wang et al., 2013]. This approach would of course not work if the missing person's mobile phone ran out of battery time, which is a possible scenario if the SAR operations takes a long time.

6.1.4 RECCO

The *RECCO* system is a well-known snow avalanche SAR equipment for detecting people who are buried underneath a snow avalanche [RECCO]. The RECCO device is fastened in many clothes and sportswear worn by people that are frequently in the mountains. A possible approach for detecting humans from an UAV could be done by detecting the RECCO reflectors fastened in the missing person's clothes. The range of a *RECCO* device is 200 m through air, and 30 m through snow, but this is highly dependent on the snow amount and some other parameters such as the condition of the snow, etc. The *RECCO* system is a two part system where the rescue personnel has an active searching device, and the devices that are search for are passive reflectors and usually fastened or sewn in the clothes of the people who are buried by the snow. Having a *RECCO* detector on board a UAV would be a way to detect humans underneath the aircraft and not necessarily just in a snow avalanche scenario. If the missing persons are buried by snow, the UAV might not detect them due to the altitude between the UAV and the ground, but if they are close to the surface or not buried by snow at all, it could be a quick and easy way of discovering them.

An idea a UAS called *Airborne Avalanche Rescue System* is an attempt to utilize the capabilities of the *RECCO* system on board a UAV for finding people who are buried by a snow avalanche [Yanko Design and Rolle].

6.2 Detection System

The following sections will explain the parts and ideas behind a computer vision human detection system that can be used as part of a SAR UAS. A general algorithm for human detection performed from a UAV will be proposed, and afterwards a proof-of-concept system is developed and tested to prove that such a system is feasible to develop.

6.3 Architecture

The features mentioned in Section 4.2.2 are important for a SAR UAS and they will consequently influence the development of a detection system that is going to be used as part of this UAS. Since the detection processing is to be performed on the aircraft, the weight and power consumption of the detection system will be limited since the aircraft needs to provide the power and carry the devices that are used. The processing should be performed on the UAV as not to put such high demands on a possible communication link used between the UAV and its operators, and to allow the detection system to work autonomously without the need for a connection with a GCS.

6.3.1 Algorithm Design

As mentioned previously, a combined approach using thermal and color cameras is a likely solution for detecting humans from a UAV during SAR operations. An activity diagram of an algorithm taking advantage of this combined solution is illustrated in Figure 6.2. The algorithm take advantage of having two cameras, both a thermal and a color camera. The color camera is likely to have a higher resolution and is therefore more suited for determining if an object is actually a human or not.

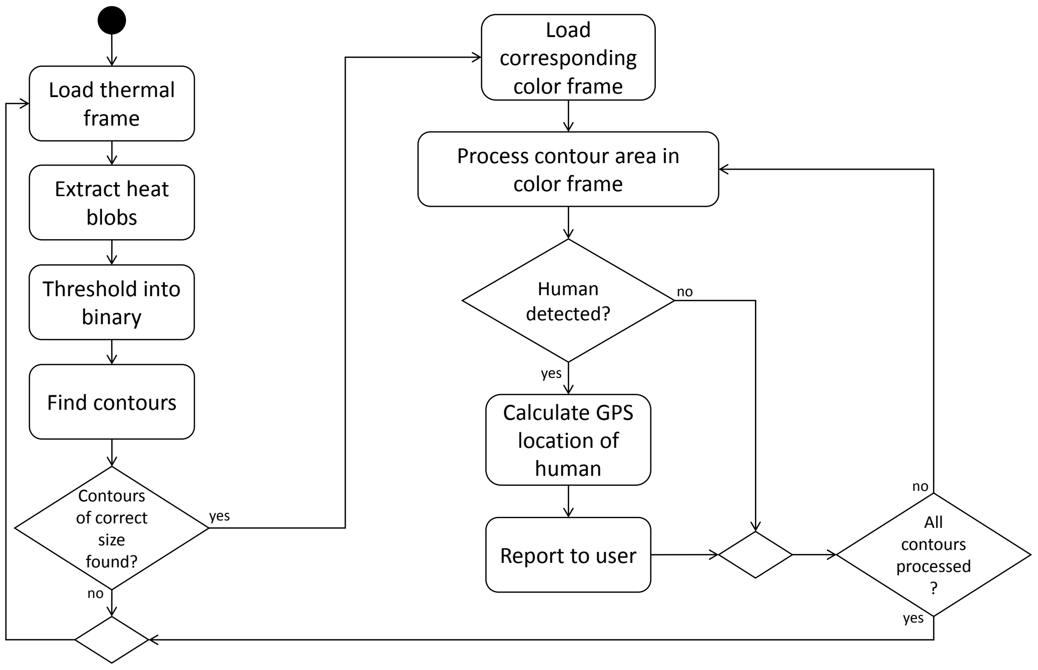


Figure 6.2: Activity diagram of algorithm

The algorithm illustrated in Figure 6.2 can be summarized as following. A thermal image is loaded and segmented based on the heat signatures in the image frame in order to extract heat blobs (collections of heat signatures in the thermal image). The frame is thresholded based on a temperature limit and transformed into a binary image. Everything below the temperature chosen will be excluded from the frame. The contours of the heat blobs in the binary frame are extracted and if any contours of the correct size are found, meaning the contour could be a contour of a human given the altitude of the UAV, the corresponding location in the color image is determined and processed for human features. If a human is detected, the GPS location of this human is calculated based on the location and orientation of the UAV, and the result is sent to the GCS or an operator.

This combined approach will not work in the dark because the color camera will have no visual contact with the object. However, if the resolution of the thermal camera is good enough, the region in the thermal image could be analyzed instead of the color camera region. The altitude of which the UAV covers an area might have to be altered in order to fit the needs of the detection system. If a lower resolution camera is used, the UAV will have to fly closer to the ground than if a higher resolution camera is used in order for the features needed to do a human detection to be visible. The use of both thermal and color cameras will decrease the system cost as the thermal camera can have a lower resolution because it does not need to detect the features of a human, and hence the thermal camera can be cheaper resulting in less cost for the entire system in total.

6.4 Prototype Detection System

This section will present and discuss the development of a prototype detection system based on the architecture described in Section 6.3. This system is only developed to prove the feasibility of such a detection system and will thus include many assumptions and simplifications regarding the situation that the system will be used in.

Since a thermal camera was not available during this Thesis, an ordinary webcam was used and instead of searching for heat signatures in a thermal image, the color red was searched for in a color image. Searching for red intensities in an image will be approximately the same as searching for heat intensities in a thermal image and the algorithm will therefore be fairly similar if an actual thermal camera is acquired thus making the new camera easy to utilize. Searching for the color red could possibly be more difficult than looking for heat signatures since the color red is also present in other colors, but simplifications regarding the detection scenario will be made during this Thesis, which will make the detection easier.

In order for the detection system to determine the position of the objects it detects, it needs to be connected with the autopilot, or in some other way, get the orientation, heading, altitude, and position of the camera. The localization of objects will not be part of this prototype, but a separate proof-of-concept system is explained and tested in Section 6.6.

During the development in this Thesis, the open-source computer vision library OpenCV, mentioned in Section 3.5, was used along with all its capabilities. Since C++ is the standard programming language of *OpenCV*, and the favorite of the developer developing the application, C++ was used for programming the detection system.

6.4.1 Prototype Setup

The prototype human detection system created in this Thesis uses a low cost, low-energy consumption embedded computer called *BeagleBone Black*. The embedded computer is connected to a *Logitech c270 HD* webcam via USB, which provides the embedded computer with color images that can be analyzed using the computer vision library *OpenCV* that is installed on the *BeagleBone Black*.

The detection application is written as an approach to the algorithm illustrated in Figure 6.2.

Because of the limitations of using an embedded computer to perform the processing, some techniques to increase the efficiency of the application are necessary to obtain the required detection speed. Since the detection system needs to gather the images, process the images, and communicate with the user, having the ability of programming concurrently as mentioned in Section 3.7, will be advantageous and thus be used in this system. The processing power on board a UAV will be limited and thus fully utilizing the processor is highly desired.

6.4.2 Prototype Algorithm

As mentioned, in order to create a prototype system based on the algorithm illustrated in Figure 6.2, a few assumptions are made. The assumptions had to be made because of equipment availability and to allow a workable system to be created in the time for which this Thesis was written. The human, which is to be detected, is assumed to be covered completely in the color red, except for their face, and there is only one human in the image at the same time. No other objects in the image are covered in red or they are assumed to have a much lower red intensity than that of the human as to not disturb the detection. Instead of detecting a human body, the face of a human is detected in this prototype application for illustration purposes. The detection of a face is used due to the opportunities that the *OpenCV* library provides. The calculation of the GPS position of the detected object is as said not part of this prototype system, but a different proof-of-concept system will be presented in Section 6.6.

The activity diagram of the modified algorithm used for this prototype is illustrated in Figure 6.3. The actions and decisions with the color yellow are the ones that are altered and thus different from that of the original algorithm from Figure 6.2.

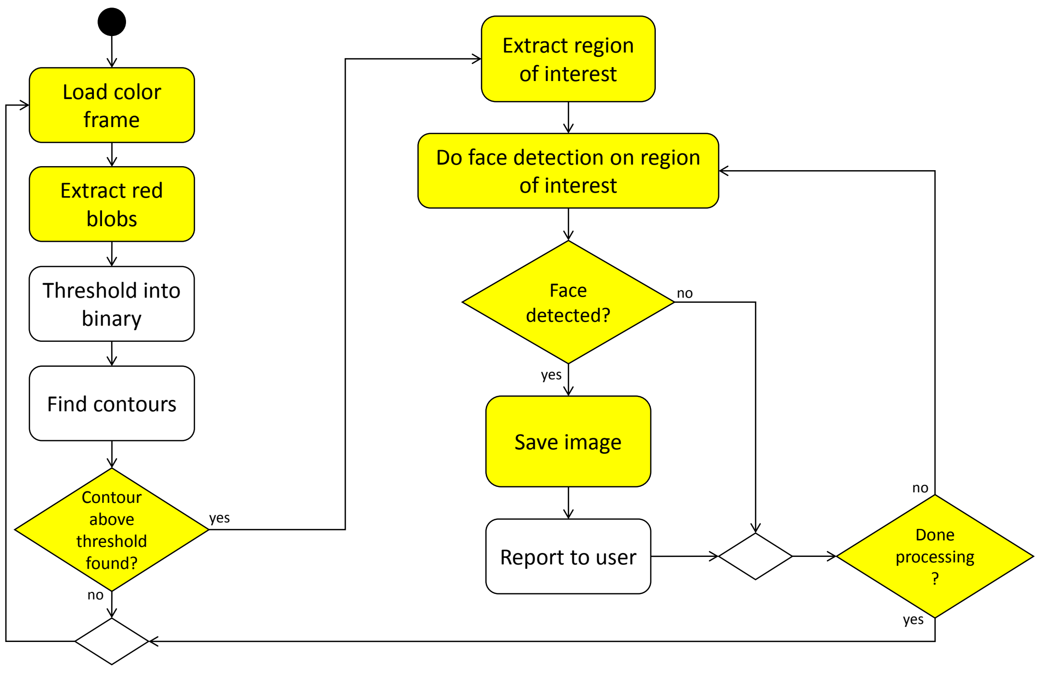


Figure 6.3: Activity diagram of prototype algorithm

The prototype algorithm illustrated in Figure 6.3 can be summarized as following. A color image is loaded and segmented based on the intensities of the color red in order to extract the red blobs in the image. This segmented image is then thresholded into a binary image based on a certain red intensity value. If something is below the limit, it is removed from the frame and one is left with a binary image. The contours of the red blobs are then found in this segmented image and if a large enough contour is not found, a new frame is loaded. If a large enough contour is found, a slightly larger region than that of the contour is extracted and a face detection is run on this extracted frame. The reason for extracting a slightly larger region is that the face of the human will not be covered in red during the tests and will thus not be part of the contour. Extracting a region that is somewhat larger than the contour will most likely also include the face of the human, hence making face detection possible. If a face is detected, the face is marked on the image and the image is saved. Lastly, the user is notified about the detection. The detection of a face will probably not be a solution that can be used during SAR operations, but will show that such a detection is possible to perform using this setup, hence detecting human bodies will probably also be possible to detect.

6.4.3 Prototype Application

The detection application is run on a *BeagleBone Black*, which is setup as explained in Appendix A. The code assumes that a webcam is connected to the *BeagleBone* via a USB cable. The application works by taking an image from the webcam input stream and performing the detection steps as illustrated in Figure 6.3 on as many of these images as possible. The newest image is always used and the others are discarded. The face detection step in the algorithm is accomplished using a function that *OpenCV* provides. This *OpenCV* function detects objects by using Haar feature-based cascade classifiers [Viola and Jones, 2001]. This is a machine learning based approach and the *OpenCV* library provide its users with some already created classifiers, which are used during this Thesis.

The prototype application code is listed in Appendix A.4, Listing A.4. The benefits of utilizing a computer vision library such as *OpenCV* is illustrated in Listing 6.1 where code for detecting a face is shown. The provided function takes an image as input, along with a vector called found, where the detected faces positions are stored. The other inputs are scaling factors and algorithm specific parameters, which will not be explained here. The *OpenCV* community provides a guide for using the library and the functions are well explained there. The compact implementation of advanced functions are a huge benefit from utilizing a computer vision library such as *OpenCV* which results in ready-to-use applications much faster than without.

Listing 6.1: OpenCV Face Detection

```
face_cascade.detectMultiScale(img, found, scaleFactor, minNeighbors, flags, minSize);//
    , maxSize);
```

The detection application in Appendix A.4 utilizes two threads of execution. One thread collects images from the webcam stream and the other thread performs the detection steps on as many of these images as possible. Because of the particular webcam used, it was necessary to divide the application into two threads in order to always process the newest image. If the retrieval of images was done by the detection thread, old images were processed because of a buffer storing old images, and this was not desired.

Because the detection application consists of two threads, inter-task communication was necessary to be used. Shared memory was chosen since this is probably the easiest method to use. Line 131 to 136 in Listing A.4 shows the variables that are shared between the two threads, which are a memory location for the image, a test integer, and an algorithm mode selector integer used to choose how the algorithm is run. The most recent image was saved to a shared memory location that both threads had access to. This shared memory was protected by a mutex so that the image was not accessed at the same time by the two threads. Condition synchronization was also used during the startup of the application in order for the webcam retrieval thread to retrieve one image before the detection thread started its execution. The retrieval thread collects one image and then sleeps for some time in order to allow the detection thread to run and get the image that is stored in the shared variable. During testing, this sleep had to be introduced because the retrieval thread was starving the detection thread by not allowing the thread access to the image. If the detection thread was scheduled to run when the retrieval thread was accessing the shared image the detection thread could not start its processing of that image and was thus forced to wait for the image to be released, resulting in a very variable waiting time for the detection algorithm thread. This could have been solved in other ways, for instance by having multiple mutexes or using two different shared memory locations for storing the image.

6.5 Prototype Detection System Testing

To establish the usability of the prototype detection system explained, some tests using the detection application was performed on a couple of still images. Instead of utilizing the webcam to gather the images, a mobile phone was used to capture some still images and the processing was performed on these. The code for this still image detection application is listed in Listing A.3, and is almost the same as the code that utilizes the webcam. The path of the image is added as an argument to the application, and there are no threads running. The detection is done inside the main loop and only performed once on the image specified. Doing the detection on still images will be the same as performing the operation on the stream of images captured by the webcam. If a webcam is used, a small delay will be introduced because the images will have to be transmitted by the webcam to the unit performing the detection, but this test is simply done to show the usability of the system and hence still images will be sufficient.

The images used to test the prototype detection application was captured from the roof of a building, roughly 15 meters above the ground, as to simulate the image being captured by a UAV flying in the air. The images were of high quality with a size of 3264 x 2448 pixels, and one of the images used is shown in Figure 6.4. As explained, the application is used to detect faces in the images. When a face is detected, the face is marked with a green circle on the image and this image is saved to memory. The time spent on doing the detection is recorded so that an analysis can be made regarding the run-time of the application. The human in the image is wearing a red sweater so that the segmentation steps of the prototype algorithm illustrated in Figure 6.3 are possible to perform.

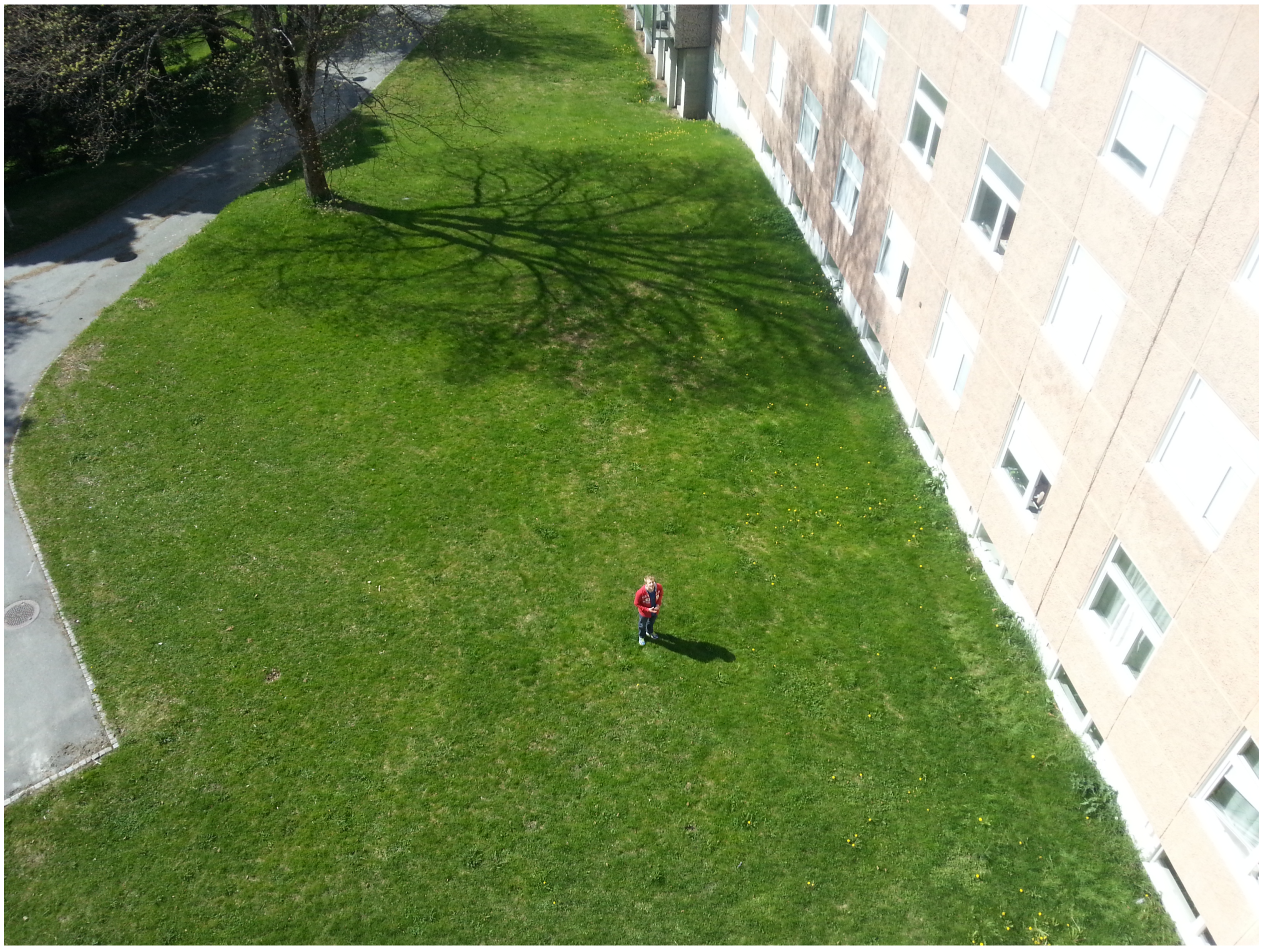


Figure 6.4: Picture of human taken from roof of a building

6.5.1 Results

The application was run on the embedded computer *BeagleBone Black* which was setup as explained in Appendix A. Figure 6.5 shows the result of the face detection performed on the image from Figure 6.4. The face is detected in a Region Of Interest (ROI) that is extracted around the largest red blob in the image and the face is marked with a small green circle. This proves that a detection is possible to accomplish using this algorithm and equipment.

In order to prove the advantage of using the combined approach with thermal and color cameras, the face detection on the image in Figure 6.4 was performed in two different ways. First, the face detection was done on the image using the *OpenCV* algorithm with no segmentation or blob extraction of any regions. The second time, the detection was done utilizing the algorithm steps from Figure 6.3 by extracting red blobs and performing the face detection on the ROI around the biggest red blob. The results are clear and the run-times in seconds of performing the detection on four different images are shown in Figure 6.6 where the x-axis is the time axis. The orange lines are the detection run-time when the *OpenCV* face detection function was run directly on the image itself, and the blue lines are the detection run-times when utilizing the

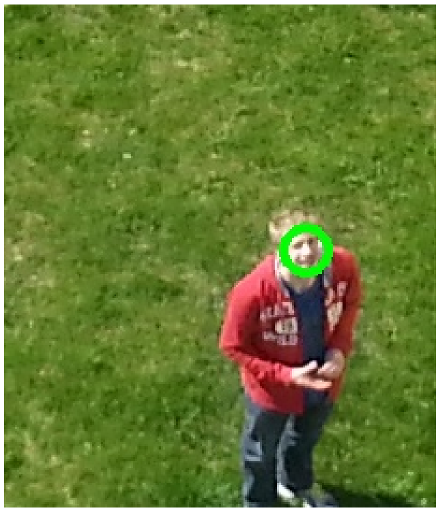


Figure 6.5: Face detected in image

algorithm as explained earlier, using blob extraction.

The algorithm was also tested on an image capture by the webcam from the same location. This image had a resolution of 640 x 480 pixels and the face was not possible to be detected in the low-resolution image. The comparison between a high- and low-resolution image is shown in Figure 6.7.

6.5.2 Discussion

The time saved by utilizing the blob approach was massive. By utilizing the algorithm illustrated in Figure 6.3, i.e. extracting a ROI based on color segmentation, the run-time is cut from an average of 167.5 seconds, to 2.4 seconds when performing the face detection on the *BeagleBone Black*. That translates to a decrease in run-time of 98.6% for the particular images that were used. The images were quite large and doing the detection on smaller ones would probably result in a different decrease than this test showed, but still the test shows that the time saved by utilizing the explained algorithm are large.

Although the described algorithm takes part in the detection speed-up, other parameters influence the run-times as well. The images used were as explained, large images with many pixels in order to have the features of a human face visible. The same algorithm and code was run on a laptop and the results came out slightly less obvious. Utilizing the algorithm, i.e. extracting a red blob region before performing the face detection, the application used roughly 0.5 seconds in order perform the face detection. When the detection was run without extracting any region, simply using the *OpenCV* face detection algorithm, the application used roughly 25 seconds. The difference between this test and the one performed on the *BeagleBone Black* was probably affected by the differences in memory capabilities. The laptop used has 8 GB of RAM compared to the 512 MB of the *BeagleBone*. The laptop also runs on an Intel based processor, which the *OpenCV* library is optimized for, which is not the case for the standard compilation of *OpenCV* regarding arm processors, which is what the *BeagleBone Black* possesses. However, the results

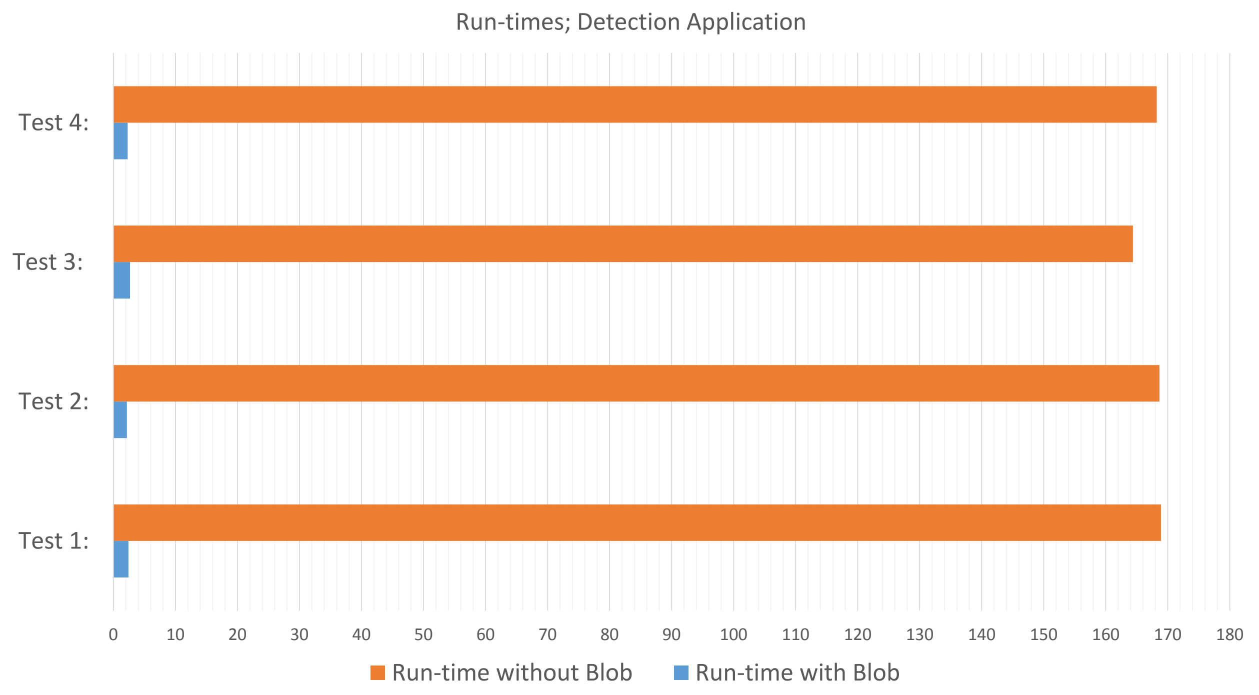


Figure 6.6: Graph of Detection Application run-times

from the tests are clear and performing feature detection on a smaller part of an image, for by example utilizing thermal images for segmentation purposes, is a very good solution to use during SAR operations.

The detections were performed running the CPU of the *BeagleBone* at 300 MHz. A test was performed where the frequency was altered, as shown in Appendix A.2, to 1000 MHz, but that did not increase the run-time of the application by much. The result of the tests performed on the *BeagleBone Black* might have been improved if some optimization steps regarding the *OpenCV* library had been performed. As mentioned in Section 3.5, the increase in performance of the *BeagleBone Black* could be as much as doubled by utilizing the optimization features of the *OpenCV* library for ARM processors. Although the detection took 2 seconds using the *BeagleBone Black*, face detection is a difficult and time-consuming task and utilizing other detection techniques could lower the run-time. If the optimization steps were taken and the run-time halved, a detection taking 1 second would probably be sufficient in a SAR scenario depending on the altitude and speed of the UAV. The detection on board a UAV during SAR operations cannot take too long as some regions might be missed if the images are not processed fast enough. In the report by Vegard Hammerseth, a calculation example showed that having a system that could process the images in half a second would be sufficient for use during SAR operations [Hammerseth, 2013]. This could possibly be achieved on the *BeagleBone Black* by optimizing the code and algorithm explained in this chapter.

It is possible to train the face detection algorithm of *OpenCV* to discover other objects such as bananas or pencils, and the algorithm could be thought to detect human bodies from the air. This would be suitable for use during SAR operations as a face detection is probably not very

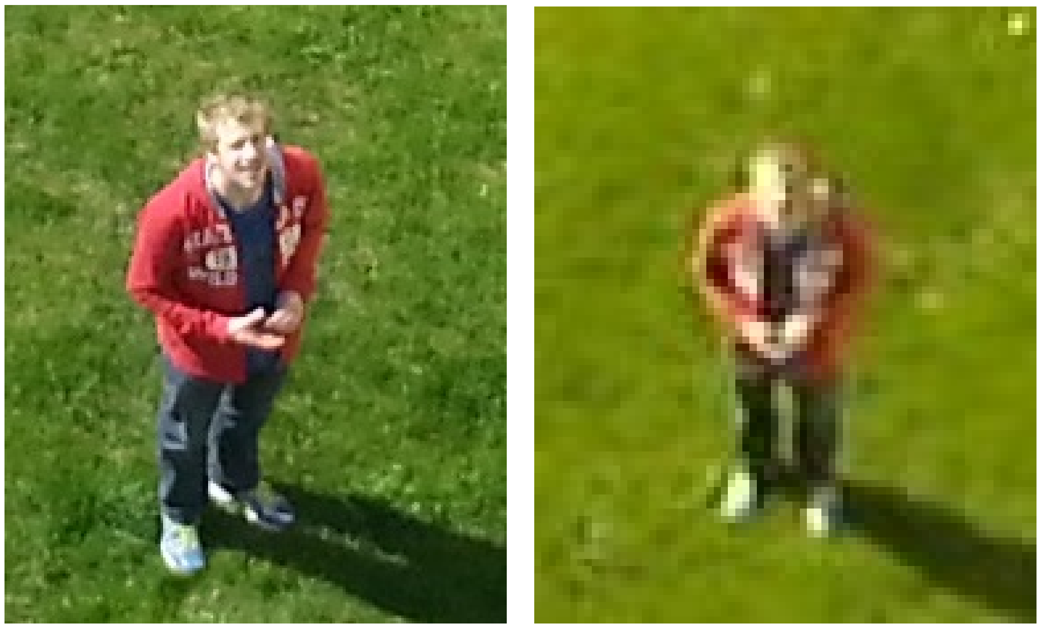


Figure 6.7: High- and low-resolution images

useful. Teaching the system to detect humans from above will be a good solution in order to make the detection possible.

The detection application was tried on a low-resolution image and the face was not detected in this low-resolution image. This enhanced the assumption that detecting a face from the air might not work very well when a low-resolution camera is used, or the altitude of the UAV is high that the face features will not be visible on the image. The images used during the test ware taken fairly close to the ground and a UAV is probably going to fly higher than the altitude used in the test. This altitude would probably make the facial features of a human in the image disappear. In the low-resolution image, illustrated to the right in Figure 6.7, the features of the face are clearly not visible and the face was therefore not detected by the application. This low-resolution test enhanced the reasoning for utilizing a higher resolution color camera during the last steps of the detection, but will depend on which features that are used for performing the detection. If other features than facial features are used, a low-resolution image might be sufficient for detecting an object. For example, by extracting the contour of an object and classifying the object based on its shape, a lower resolution image could possibly be used. A simple ellipse detection might be sufficient for determining if the object is likely to be a human or not.

6.6 Localization

During a search, it will be important to get the location of the objects that are detected by the UAV. One approach is simply to provide the location of the aircraft when an object is detected [Hammerseth, 2013]. This approach will introduce some error to the reported location of the objects and if the plane is turning or flying relatively high, the reported location could be inaccurate and possibly mislead the SAR personnel to go towards the wrong world location. However, providing information regarding the orientation of the UAV to the SAR crew, and using the knowledge of the location of the UAV, the SAR personnel could estimate the most likely location of the recognized object. This localization step of an object recognition algorithm is the "Calculate GPS location of human" step illustrated in Figure 6.2 between the human detection and user notification steps. This was as explained not part of the prototype detection application.

As mentioned, if information regarding the position and rotation of the camera doing the detection is used, a camera detection application can estimate the position of the object it detects [Johnston Jr., 2006]. This position can be used by the SAR personnel to give them a more accurate estimate of the objects position thus increasing the possibility of finding them. In Section 3.6.3, a method for estimating the location of an object seen in an image is roughly explained.

If a UAV is flown by an autopilot and controlled using GPS waypoints. The UAV will necessarily know its position, latitude, longitude, and altitude, and also its orientation relative to a world coordinate system with GPS latitude and longitude axis. The orientation will be in forms of compass bearing (yaw), and also pitch and roll around the axis. These measurements can be used by a localization system in order to determine the position of the objects seen by the camera which is mounted on the UAV and thus has the always has the same orientation relative to the UAV.

This section will describe the setup, implementation, and testing of a proof-of-concept localization application utilizing the information explained in Section 3.6.3.

6.6.1 Test Setup

In order to estimate the world location of objects seen in an image, the orientation of the camera coordinate system relative to the world coordinate system needs to be known. In addition, the camera matrix of the used camera needs to be known in order to determine the world coordinate of a given image pixel.

Camera Calibration

As explained in Section 3.6.4, a camera calibration can be done using the *OpenCV* library. The library provides its users with a camera calibration application, which makes finding the *camera calibration matrix*, **K**, possible. This matrix is as mentioned specific for each individual camera, and the one found for the camera used during this Thesis is listed in Listing 6.2, which is extracted from the localization application in Listing B.1 in Appendix B.3. An *OpenCV* provided matrix structure is used for storing the *camera calibration matrix*.

Listing 6.2: Camera Matrix

```
Mat K = (Mat_<double>(3,3) <<   8.1331102645202134e+02, 0, 3.1950000000000000e+02,
                   0, 8.1331102645202134e+02, 2.3950000000000000e+02,
                   0, 0, 1 );
```

Coordinate Systems

By using the world's latitude and longitude axis as the basis for the world coordinate system, the rotation of the UAV and hence camera coordinate system can be described relative to this.

During the localization testing in this Thesis, the GPS position of the camera location along with its height above ground and orientation was hard-coded into the application.

The localization application code is listed in Listing B.1 and the formulas used in the application for calculating the bearing and distance between two GPS locations, and also calculate a new GPS location given an old one, are all listed in Appendix B. In order to translate a pixel position to a world GPS location, the camera coordinate systems rotation relative to the world coordinate system was needed to be known. Because of the way *OpenCV* interprets images, the camera coordinate system is not as defined in Figure 3.2. Since *OpenCV* starts counting pixels in the top left corner of an image, the coordinate systems can instead be illustrated as shown in Figure 6.8, where the latitude/longitude axis of the world frame are shown as X_w and Y_w. The altitude axis of the world coordinate system is rotated down into the ground to make the rotation between the camera and world coordinate systems easier to determine.

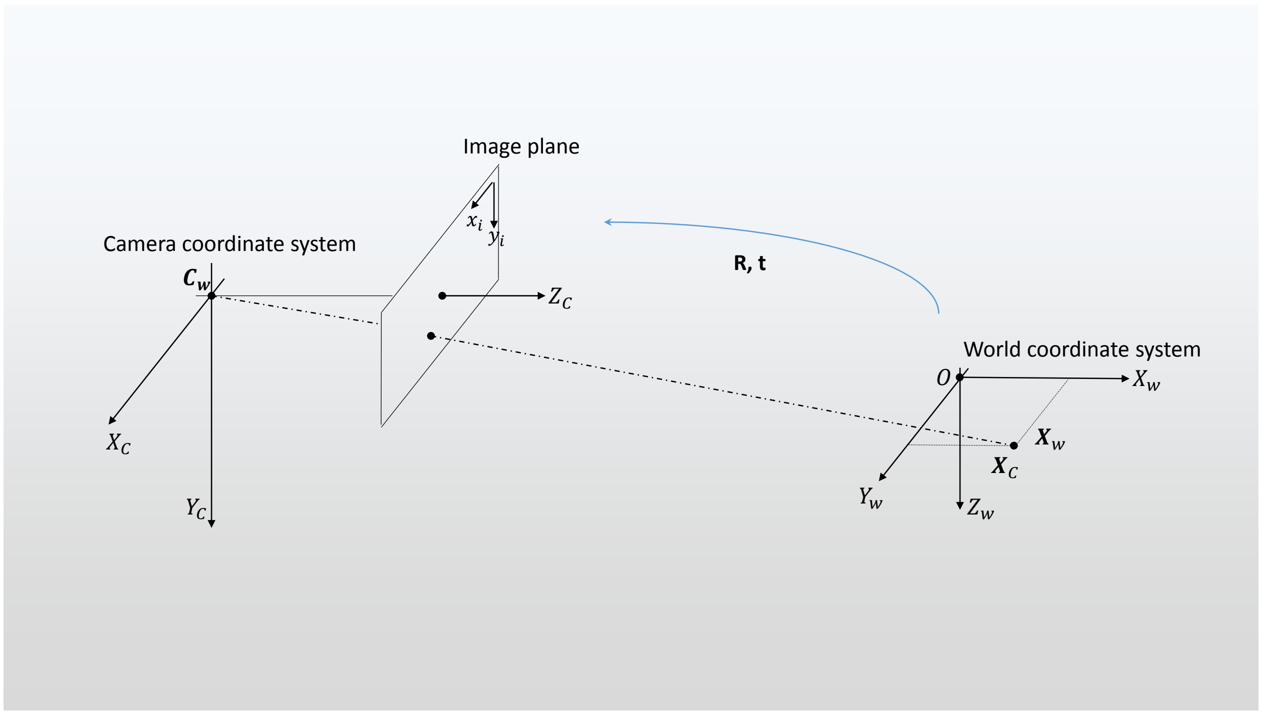


Figure 6.8: Camera coordinate system and World coordinate system

In Figure 6.8, the camera center $\mathbf{C}_w$ is the latitude and longitude GPS world coordinates of the cameras position including the negative altitude of the camera, meaning the negative height it was placed above ground level. The negative value is used since the world coordinate system is defined with the Z-axis downwards. The ground plane beneath the camera is assumed to be level (flat) during the testing, which facilitates the localization part. The world coordinate system is the GPS latitude/longitude system. X_w points in the north (latitude) direction and Y_w points in the east (longitude) direction.

In order to facilitate the localization calculations, the center of the camera coordinate system was always set to $(0, 0)^T$ in world coordinates. Hence, the illustration in Figure 6.8 is slightly incorrect, as the position of the camera should be placed directly above the origin in the world coordinate system. Always defining the camera center at the $(0, 0, Z)^T$ in world coordinates makes it possible to determine the location of a detected object in terms of movement in the X and Y world frame directions. When using the world latitude/longitude axis as the world coordinate system, the movement in X and Y directions equals' movement in the latitude and longitude axis. The application result of the transformation from pixel to world coordinate was a movement in meters, starting from the GPS location of the camera, along the world coordinate system axis, i.e. the latitude and longitude axis. The next part of the localization application in this Thesis has not been studied in detail. The formula used to estimate the new GPS position given the movement in meters along the latitude and longitude axis from a starting position, is listed in Appendix B.2. This formula was found on the Internet and it was not evaluated properly, thus its credibility was not verified. However, the formulas was tested and proven to work well, at least in the location this test was performed. If the testing had been performed closer to the poles of the earth, the equation might not have worked that well, but the distances in these tests are so small that any error in these equations will be less than any measurement errors done during the setup of the system. Thus making the errors in the equations used close to non-relevant for the results.

Camera Rotation Determination

The rotation of the camera coordinate system relative to the world coordinate system in Figure 6.8 can be described by the following rotations using the right-hand rule; starting with the two coordinate systems aligned, the camera coordinate system is first rotated $\pi/2$ about its Z-axis. After this rotation, the top of the image plane is pointing north, and the right of the image plane is pointing east. Then the camera coordinate system is rotated $\pi/2$ around the X-axis to get the Z-axis of the camera coordinate system pointing north, as shown in the figure. The rotation matrices used are shown in Section 3.6, Equation 3.3. In Figure 6.8, the camera center is also moved with a translation compared to the world coordinate system. This translation was in the testing only consisting of movement in the negative Z-axis direction since the camera center was always defined as $(0, 0, Z)^T$, as explained earlier.

The rotation around the cameras Z-axis after the top of the image plane had been aligned north (the camera coordinate system rotated $\pi/2$ around its Z-axis from the same starting position as the world coordinate system), is the same as the bearing of the camera, or equivalently the bearing of the UAV if the camera is mounted on the UAV facing forward. This means that if the image center (i.e. the Z-axis of the camera coordinate system) is pointing straight east, the bearing of the camera is $\pi/2$, or 90 degrees. Since the camera was not mounted together with any autopilot or AHRS during the localization testing, it was difficult to determine the correct rotations of the camera. In order to determine the rotations, the GPS locations of the camera center and of the spot the center pixel in the image was directed towards, was used. The center pixel was directed towards a familiar GPS location and the bearing, i.e. rotation around the camera coordinate systems Z-axis, between this location and the camera location was determined using Equation B.1. The angle of rotation around the cameras X-axis was determined by using the distance between the two GPS locations, found by using Equation B.2, and the altitude above ground at where the camera was placed. This transforms into a geometry problem where Pythagorean Theorem, ARCTAN(distance/height), was used to get the rotation around the cameras X-axis. The rotation around Y-axis was assumed to be zero during the tests.

Localization Test Setup

The camera was set up at the GPS location (63.418560, 10.401090) in latitude/longitude decimal degrees. The center pixel was pointed at location (63.418856, 10.402342) and the height of the camera was found using a barometer to be 16 meters above this point. Using these locations, the bearing of the camera was calculated to be 1.084712 radians, or 62.1495 degrees rotated clockwise from north. This made the rotation around the cameras Z-axis to be $\pi/2$ (to make the top of the image plane be towards north) + the bearing 1.084712. The rotation around the X-axis was then found using the distance between the camera position and the center pixel position and the altitude of the camera. The rotation was determined to be 1.347489 radians, or 77.2054 degrees in positive X rotation. The rotation around the cameras Y-axis was set to 0 as mentioned previously.

6.6.2 Results

After providing the application with the GPS location and rotations of the camera, a person was placed in a location that was in the view of the camera, i.e. in the image frame. The pixel position of this location was extracted and the GPS position estimated using the localization application.

Two screen shots of the application image are shown in Figure 6.9. The blue circle in the middle of the images is a marking of the center pixels in the image, which was as explained, pointed towards a known GPS location in order to be able to calculate the cameras rotation and bearing. The cursor was clicked at the locations where the person was viewed in the image, and the application wrote out the estimated GPS coordinate of that pixel in the terminal. The screenshot to the left in Figure 6.9 illustrates when the person was located at what will be called test location A, and the screenshot to the right is when the person was at what will be called test location B. The person is quite difficult to make out in the images because of a somewhat low camera resolution used during the tests, but the person would using a thermal camera be more separated from the environment and thus easier to locate.

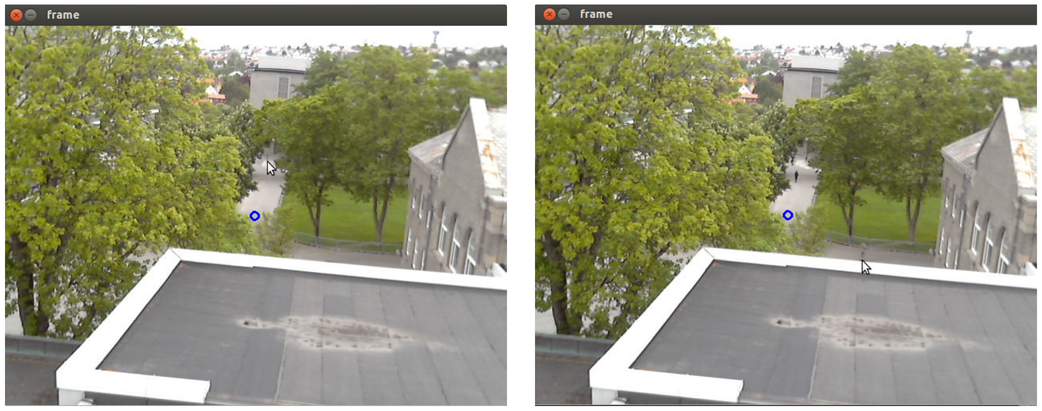


Figure 6.9: Screenshots from localization Application

The GPS position of the test locations, A and B, were determined by using Google Earth [Google Earth]. Table 6.1 shows the real GPS location and the applications estimated GPS location in decimal degrees. The error in meters, i.e. distance between the real and estimated locations, are shown to the right in the table. The locations used during the localization test are illustrated on a map in Figure 6.10. This map, in addition to the test locations A and B, also includes the camera center location (C) and the location which the center pixel was pointed at and used for determining the rotations that were used in the application program (P). The estimated locations of the detection application are also shown in the map with the yellow tags.

Table 6.1: GPS Locations of test points A and B

Location	Real GPS Location	Estimated GPS Location	Error (m)
A:	(63.419068, 10.403247)	(63.419016, 10.403123)	8.45
B:	(63.418722, 10.402130)	(63.418729, 10.402082)	2.5

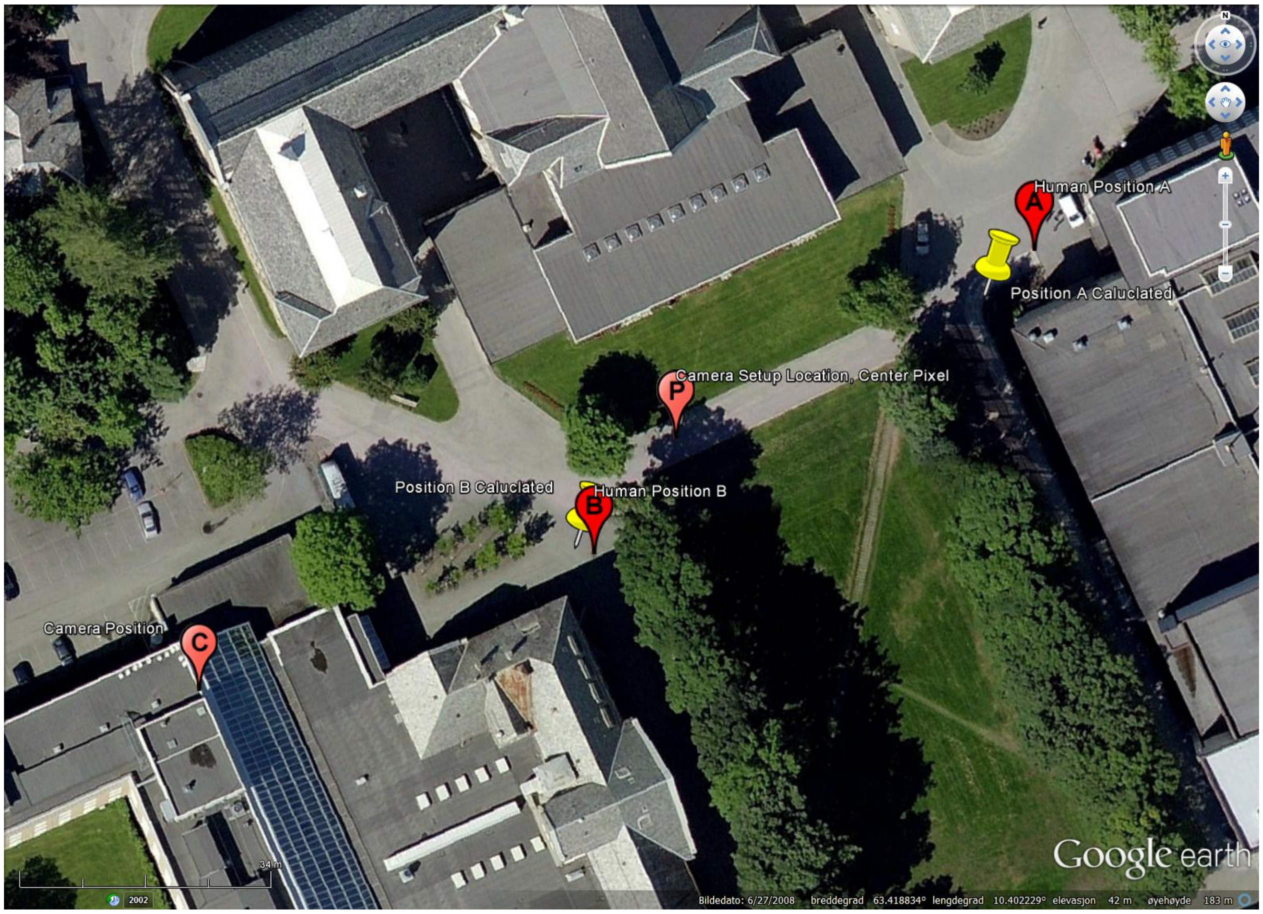

Human Position A
Position A Caluclated
Camera Setup Location, Center Pixel
Position B Caluclated
Human Position B
Camera Position
Google earth
Bildedato: 6/27/2008 breddegrad 63.418834° lengdegrad 10.402229° elevasjon 42 m øyehøyde 183 m
2002
34 m

Figure 6.10: Localization Test, GPS Positions

6.6.3 Discussion

The localization application worked fairly well and provided a good estimate on the world locations given the pixels of the image. Table 6.1 shows that the error was only 2.5 meters when estimating the GPS position of test location B, and 8.45 meters for location A. Considering that the camera GPS position and the GPS position of location A is roughly 120 meters apart and that all the setup was done manually, i.e. the errors in the values of the camera rotations used could be significant, 8.45 meters is not that much. A SAR searcher would have been able to locate the missing person if the GPS position was defined within a radius of 10 meters. The ground was assumed to be flat during the test and the height difference between the camera setup location and test location A might have influenced the results. The error for the estimation of location B was only 2.5 meters, which is a good result. The distance between the camera center and test location B was only 55 meters and thus any error sources might not influence this result as much as for test location A.

As seen in Figure 6.9, the rotation around the cameras Y-axis was probably not zero as it was assumed to be during the tests. The hill in the horizon of the images that is following the top edge is actually a slope down towards the city center and thus the rotation around the cameras Y-axis was probably not zero during the tests. The localization test should be done once more with the actual measurements of the camera rotation used in the estimation. The test performed during this Thesis had many error sources and was thus not able to be used for verifying the accuracy of the developed application. However, the tests did prove the feasibility of a single camera localization application.

A problem using the localization approach explained in this section is that the contour of the ground during SAR operations will probably not be flat as assumed by this application. This could lead to a significant error in the estimation of the objects location. If the localization application is used from a UAV flying in a valley, and the application discovers an object, which is actually on top of the valley ridge, the estimated position of this object would be on the ground at the other side of the ridge because this is where the assumed ground level would have been [Johnston Jr., 2006]. Providing the application with map data that includes the contours of the ground could possibly solve this problem. Another problem is if an object is "detected" in the sky. The position of this object would be undefined and such a scenario should be handled by the application.

6.7 Evaluation and Discussion

Overall, an autonomous human detection system is proved feasible by the tests that were performed in this chapter. Utilizing sensors on a UAV for detecting humans from the air is an approach that could work for the SAR scenario. Utilizing computer vision is the most obvious sensor solution to use and using a combined approach with both thermal and color cameras will lower the total cost of a detection system. This combined approach could possibly provide a better result than using solely one of the cameras.

Since the focus of this Thesis is SAR UAS for use in the wilderness, many assumptions can, and was made regarding the surroundings of, and regarding the object that was searched for. There are no buildings emitting heat that disturb a possible thermal camera image, and there are most likely no other humans in the search area disturbing the detection of the missing person. These, and other assumptions might be difficult to determine and they are most likely not the same for all SAR scenarios. This will make creating a general detection system difficult and thus having a UAS where the detection system can be chosen based on the information gathered, will be advantageous. For instance, if the person one is searching for is covered with clothes from top to toe and is severely cooled, the thermal camera might not get a significant heat signature reading from them and the computer vision system will thus probably not work. Having the opportunity of utilizing a mobile phone detection system instead of the computer vision system could in this scenario mean the difference between finding the person or not. The person could be having a mobile phone turned on and thus a mobile detection system could perform the detection easier than a camera system.

If a computer vision detection system is used on board a UAV, and the UAV flies above the heads of the SAR personnel that is searching in the same region as the UAV, the UAV will most likely detect the searchers as well as the missing person. This might lead to problems as the UAV will report the searchers as detections and the UAS operator might miss the detection of the actual missing person because he is overloaded with "false" detections. This problem can be solved by providing the UAV with information regarding the location of all the SAR personnel, or doing this filtration at the GCS. When the UAV detects something at the same location of a member of the SAR crew, it knows that this is the not the missing person and therefore thus not report the detection. The UAV could also cover the area before the SAR personnel enter it, making this particular issue not relevant.

The application code written for the detection part of this Thesis is far from optimized, and studying the code and optimizing it might increase the processing speed substantially. In addition, the optimization steps mentioned in Section 3.5 can be applied to the *BeagleBone Black*, thus possibly increasing the processing speed of the application even more. The code written for the localization application and the equations used for determining the objects location needs attention. The application was solely written as a proof-of-concept system and the credibility of the sources used for calculating the new GPS location given the old one, was not evaluated properly. This should be fixed before the application can be considered functional. The localization also assumed that the ground beneath the camera was flat, thus making the estimation of the objects position easier. The flat ground assumption will probably not hold in SAR operations, but the issue can be solved by include map data containing the grounds contours in the application. Nevertheless, the application proved functional and provided well enough result to prove that the localization concept was working. This application can thus be used for providing future researchers with a development starting point for a localization application to use as part of a SAR UAS.

Chapter 7

UAS Prototype

A prototype system can be used as a development tool and can be used to make decisions regarding how a final system should be designed. This chapter will explain the setup and discuss the different components of the prototype UAS developed in this Thesis. The setup used in the pre-study is also used, with some changes, for this prototype UAS. The prototype UAS is thus almost the same as the one developed by Vegard Hammerseth in his Master Thesis, and the prototype UAS elaborated in this report, is a further development of the developed system from his work [Hammerseth, 2013].

7.1 Setup

The design of a SAR UAS was elaborated in Chapter 5. The setup of the prototype UAS in this chapter was done as an approach towards a system similar to this UAS. Some assumptions, simplifications, and modifications are done due to time and component availability. An overview of a prototype UAS vision is illustrated in Figure 7.1.

The prototype UAS does not include more than one UAV and the ground beneath the UAV is assumed to be flat to make the detection process easier. All testing was performed within VLOS since the current regulations in Norway prevents any other operations without a special permit. There was only a communication link present between the GCS and the UAV, not the UAV and the SAR personnel walking below, but incorporating a separate system for this using either *Wi-Fi* or an *XBee* module should not be too difficult. The lost or injured person is assumed to be a red clothed person, or red sweater placed on the ground in the search region.

Each part of the prototype will be explained and discussed in the following sections. Figure 7.2 illustrates an overview of the UAS components and the setup of these components. The ground control station is an application called *Mission Planner* run on a laptop. This application has a wireless connection to the UAV using a RF module, which is called a *Telemetry* module. The laptop also has a wireless connection to the detection system using another RF module called *XBee*. The UAV itself contains an autopilot system called *APM 2.6*, and a detection system run on a *BeagleBone Black* embedded computer. Although the *BeagleBone Black* (BBB) is illustrated on board the UAV in the figure, it was not placed here during flight due to safety concerns. The GPS

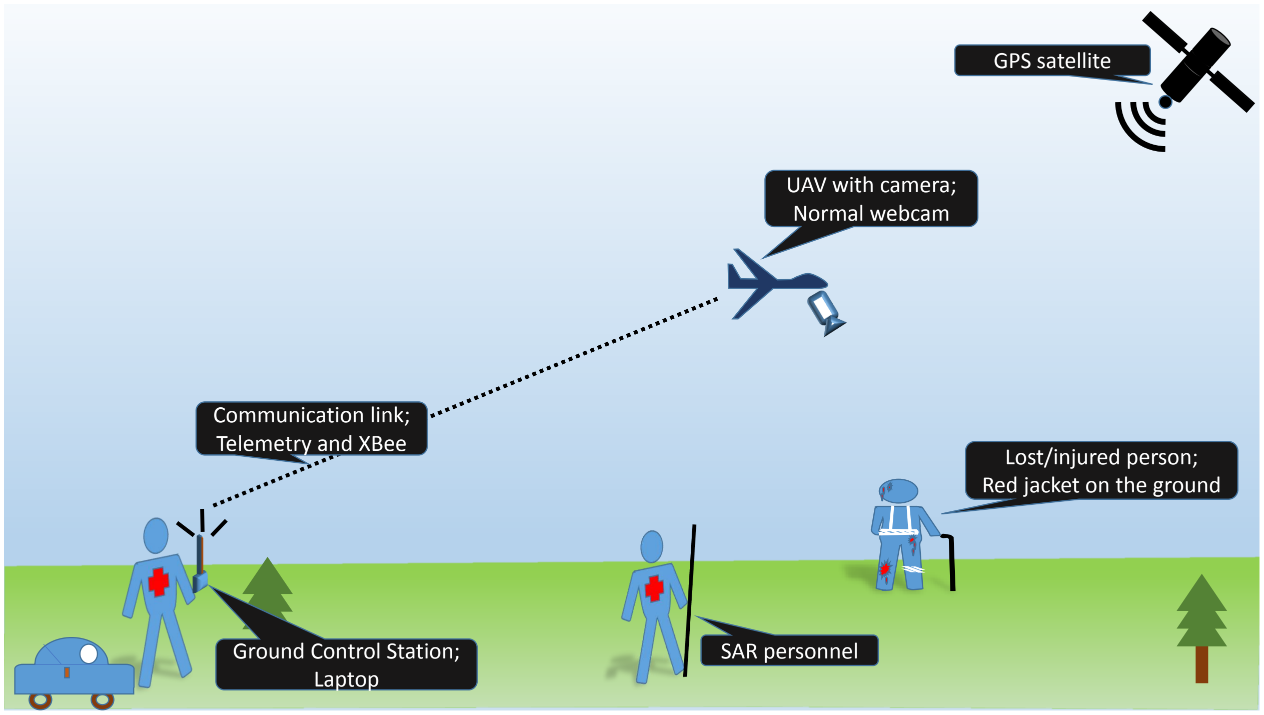

GPS satellite
UAV with camera; Normal webcam
Communication link; Telemetry and XBee
Lost/injured person; Red jacket on the ground
Ground Control Station; Laptop
SAR personnel

Figure 7.1: Prototype UAS Vision

satellite provides the autopilot with navigation information and the red jacket on the ground in Figure 7.2 illustrates an injured or missing person.

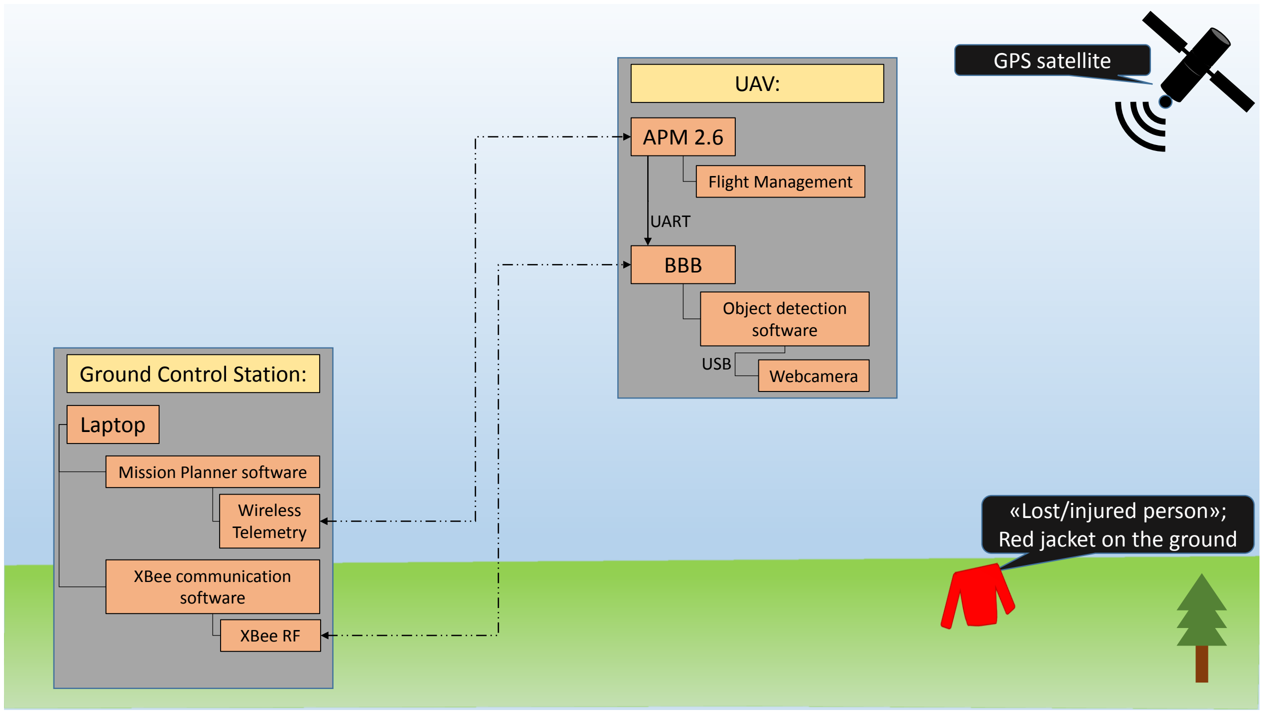


Figure 7.2: Prototype Components Overview

7.2 Aircraft

The choice of aircraft for this prototype UAS was made based on what was available at the time and it was decided to use the same aircraft as in the pre-study of this Thesis [Gamnes, 2013]. The aircraft with most of the components connected is shown in Figure 7.3. The components and their testing setup are explained in Section 8.4.

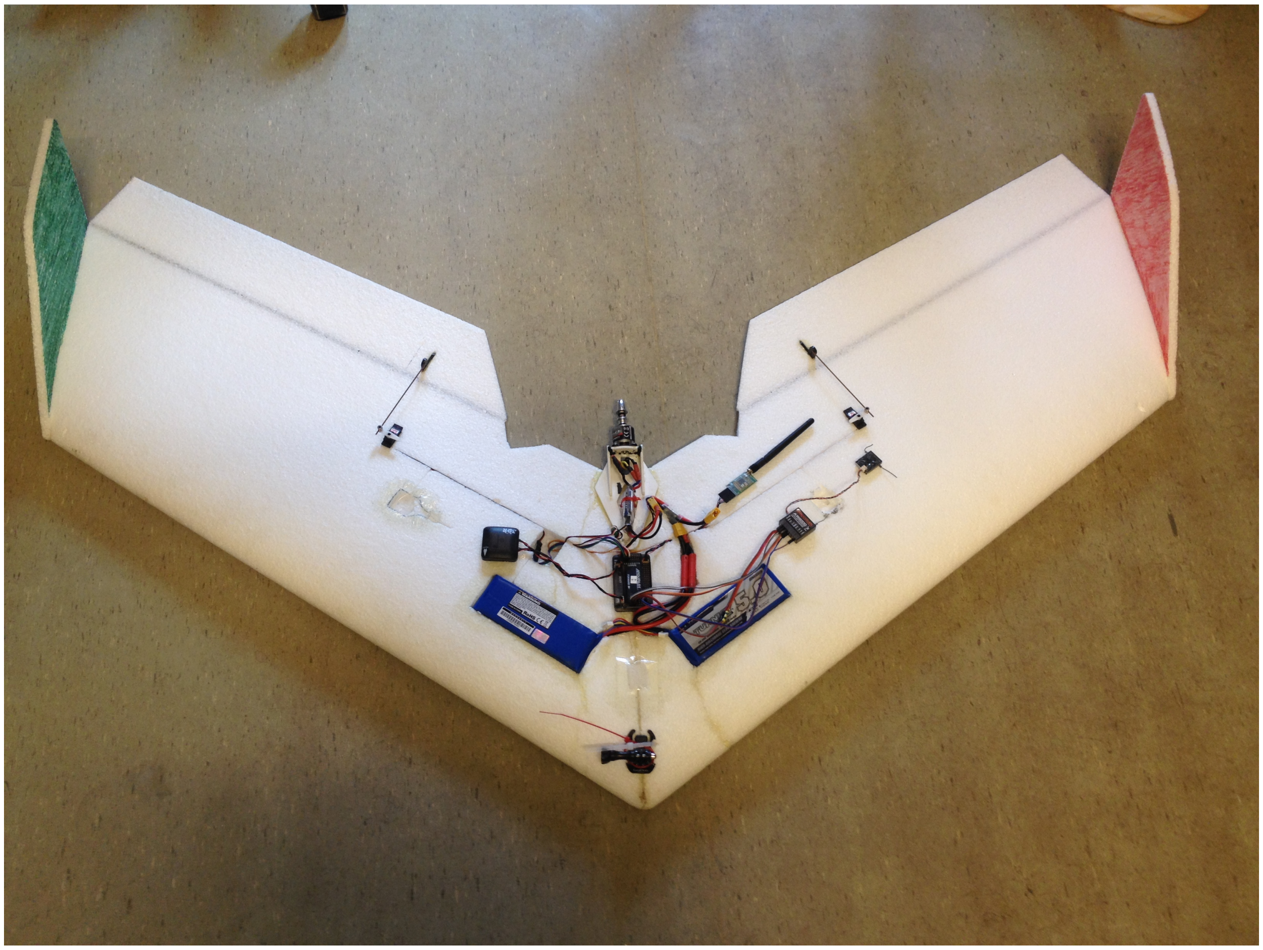


Figure 7.3: Prototype UAV

7.2.1 Airframe

The airframe is a *MaxiSwift* flying wing made of a foam-based material called Expanded Polypropylene (EPP), and the plane has a wingspan of roughly 1400 mm with a weight of 700 g [Max]. The plane was setup to carry a camera and the components of a detection system.

7.2.2 Energy Source

The aircraft is powered by two "Turnigy 5000mAh, 3S, 20C, Lipo Pack" batteries connected in parallel [HobbyKing, c]. These batteries together supply 10 000 mAh at 11.1 V and will allow the plane to fly approximately 45 minutes [Gamnes, 2013].

7.2.3 Propulsion

An engine called "Turnigy Park450 Brushless Outrunner 1050kv" was used as the UAV propulsion system [HobbyKing, a]. It has a weight of 66 g and runs on 7.4-11.1 V, which will work with the battery setup that is used.

7.2.4 Actuators

The actuators used for controlling the flaps on the plane, which in turn controls the plane, are two "HD-1810MG Digital Servos" [HobbyKing, b]. Their operating voltage is from 4.8 V to 6 V and they provide a speed of 0.13 sec/60° and a torque of 3.9 kg/cm. The size of the servos is 22.8 x 12.0 x 29.4 mm and their weigh 16g.

7.2.5 Aircraft Evaluation

The aircraft used for a prototype UAS should be even more robust than an aircraft used during an actual SAR operation. During development and testing, the aircraft will be prone to hard impact landings and thus needs to be robust. During this Thesis, the aircraft was landed using belly landing and the chance of a rough landing is therefore somewhat likely.

The aircraft used is quite large and is able to carry bulky, possibly heavy components for testing. The aircraft, being a plane includes many of the desired features for an aircraft that is going to be utilized during SAR operations. A plane has long flight range and flight time, and handles windy conditions fairly well. Because the plane is made of a foam-based material, it is easy to repair and if broke, it can be repaired using solely super glue. It has good flight stability and is easy to fly, as was proven in the pre-study where the UAV was flown by somewhat beginner pilots with no significant problems. Some problems occurred during the take-off of the aircraft, but that will be explained in Chapter 8. A multi-rotor vehicle might be more suitable for use during development as it is much easier to control and can often be flown indoors as well. However, a multi-rotor vehicle might not be able to carry as heavy loads as a plane depending on the size of the units, and hence a plane could be more suited to use during prototyping.

Batteries were chosen over fuel since they are safer and more environmental friendly than fuel. The benefit of not needing as much maintenance is also taken into consideration and is a huge benefit during the development.

The propulsions system was simply used because it was previously chosen and thus already attached to the plane. The propulsion system should, as explained in Section 5.4, be evaluated at a later stage when the plane is completely developed as the weight and shape of the plane will influence the choice of the engine.

Normal RC servos were used on the aircraft and they will be sufficient for use during prototyping and probably in an end system as well.

7.3 Autopilot

The autopilot system that was chosen for this prototype is called *APM 2.6*. This autopilot system is discussed more in Section 3.2 and in the pre-study of this thesis [Gamnes, 2013]. It is a low cost, well tested autopilot system with autonomous flight capabilities matching many of the criteria for an autopilot system used as part of a SAR UAS, as mentioned in Section 5.5.

The *APM* system contains its own AHRS and its own GPS receiver. The navigation features are already implemented providing the user with waypoint flight capabilities. The *APM* provides the possibility of autonomous flight, which includes take-off, GPS waypoint flight, and landing. A simple to use GCS application is provided to program, configure, and tune the autopilot. The control of the aircraft's pitch, yaw, and roll angles are accomplished using PID controllers. These PID controllers requires some tuning before the flight can be considered stable and autonomous flight considered possible. This tuning is performed by the user using the provided GCS application.

As mentioned in Section 5.5, the autopilot should have the capability of providing fail-safe actions to its users, and the *APM 2.6* does exactly that. If programmed correctly, the *APM 2.6* can execute safe actions based on many different scenarios such as low battery voltage, loss of communication link, or exiting a pre-defined GPS grid. For example, in the case that the communication link to the GCS is lost, the autopilot can be programmed to fly the UAV back towards a predefined "home" or launch location that was set during the start-up of the flight. When this location is reached, the UAV will fly above the location at a predefined height until the communication link is re-established.

7.3.1 Autopilot Evaluation

The autopilot used was chosen based on several features. *APM 2.6* is an open-source autopilot, which makes prototyping easier as the source code is available for review and alterations can be made if desired. The community surrounding *APM* is relatively large and thus development issues are often solved with information posted on the Internet regarding these issues, which helps save time if the same issues are encountered during setup and use. The *APM* is a relatively small device and comes with a casing that provides protection from impacts. This could be essential during prototyping as impacts may very likely occur. The autopilot is easy to set up and is well tested by its community and thus proven to function. The device only allows one external serial communication link to be used at a time using the provided firmware. This is possible to alter in the source code, making two external serial links available, but this requires the code to be studied and is not done as part of this Thesis. This is a disadvantage if other units needs to be able to communicate with the autopilot.

A small test using a quadcopter was performed to test if the *APM 2.6* worked for other frames as well as planes. The quadcopter was able to lift off and hover in the air. The quadcopter flew around for a bit and then crashed into the ground. The crash was later discovered to be cause by several of the Electronic Speed Controller (ESC) failing. They were tested and proven not to be working correctly. However, the system worked for a little while, and the changeover of the autopilot from one frame to another, did not take very long. A new firmware was loaded onto the autopilot using the provided GCS application, which made the autopilot able to control a multi-rotor vehicle instead of a plane. Although the quadcopter crashed, it flew fine for a little while, and thus showed the flexibility of the *APM* system, which will be useful both during prototyping and during SAR operations. Having the option to choose which airframe to use in one of the features desired from a SAR UAS.

7.4 Communication

During this Thesis, the communication capabilities of the *APM* system have been utilized as well as using two *XBee-PRO S2B* RF modules. A common RC controller called *DX7s*, was also used for providing manual flight control input to the UAV [DX7s].

The radio module that is implemented and sold together with the *APM 2.6* autopilot is an air-to-ground data and control link between the UAV and GCS [3DRobotics, b]. The radio module is called a *Telemetry Radio*, which is a common term within RC technology. The term *Telemetry* has the meaning of "Tele" = "remote", and "metron" = "measure", meaning that it is used for collecting data from remote locations. This particular *Telemetry* module utilizes either the 915 MHz or 433 MHz frequency depending on the country that the system is used in, and since this Thesis is written in Norway, the frequency used is 433 MHz. The *Telemetry* module is easily connected to the *APM* via a provided cable, and another *Telemetry* module is connected to a laptop running the GCS application via a USB to serial converter. Everything is setup to work without any programming by the user and the link appears as a transparent UART link for the *APM* and the GCS. The protocol used by the modules is the *MAVLink* protocol, which is mentioned in Section 3.2. The supply voltage of the *Telemetry* module is 3.7-6 VDC and the unit uses 100 mA when transmitting and 25 mA when receiving data. The serial interface is 3.3 V UART, but the device is also 5 V tolerant. The maximum output power is set to 100 mW, but can be adjusted by the user to stay within regulations. The range of the unit is not specified and is highly dependent on the antenna and other parameters of the setup. A range listed on the community website is 4.5 km and some even got more.

The *XBee-PRO S2B* module used in this prototype is explained further in Section 3.8.3. The unit has a range limit of 1600 m and is easy to setup and use.

The RC controller *DX7s* is a common remote control for RC planes and helicopters. The controller is often used for hobby projects and is a commercial product that works as expected.

7.4.1 Control Link

As the control link for the prototype UAS, the *APM*s own radio module (*Telemetry* module) along with the RC controller was used. The Telemetry module allows commands to be sent from the GCS to the UAV and also allows information to be read from the autopilot for illustration purposes in the GCS. The *DX7s* controller was used for flight control during manual, and other flight modes requiring manual input. The controller was also used to choose between the different flight modes of the autopilot.

7.4.2 Data Link

The data link of the prototype UAS was present to allow information to be sent back to a computer, or possibly mobile phone, from the detection application run on the *BeagleBone Black*. The data link used in this Thesis is as mentioned the *XBee PRO S2B*. The *Telemetry* module of the *APM* system also functions somewhat as a data link, by sending the flight information from the autopilot to its GCS.

7.4.3 Communication Evaluation

The *Telemetry* module of the *APM* system provides a good range for use during SAR operations. The unit is a commercial product and works "out of the box". The serial interface is 3.3 V UART, which will make it possible to interface with the *BeagleBone Black*. The module provide the user with a control link as well as a data link for transferring flight information. This flight information is displayed on the GCS and this functionality is already implemented in the GCS application.

The *XBee-PRO S2B* is used in the prototype UAS for connecting the *BeagleBone Black* to a laptop and providing the UAS with a separate data link. The interfacing between the devices are simple because they both utilize 3.3 V serial communication. The device is run with a supply voltage of 2.7 - 3.6 V making it possible to supply the power directly from the *BeagleBone Black*. The range of the device is 1500 m LOS, but mesh networking can be utilized making it possible to extend the range some more. 1500 m will probably not be sufficient during most SAR scenarios, but if the UAV is programmed to follow a SAR searcher and fly above their head, 1500 m will probably be more than sufficient. If the UAV flies autonomously, and performs the detection autonomously, 1500 m could be sufficient as the UAV can fly back towards the GCS to make the transmissions if necessary.

The common RC controller *DX7s* was used as a separate control link apart from the *Telemetry* module of the *APM* system. Having a separate controller during SAR operations is something that should be avoided and can be avoided if the UAS is completely autonomous. However, during prototyping and tuning, it is necessary to have such a controller in order make flight possible and to assure a safe flight.

A communication link that was not investigated in this Thesis was a Wi-Fi connection between the *BeagleBone Black* and a SAR searcher's mobile phone. If established, this link could be used for providing control and data information between the UAV and SAR personnel, which in turn could make it possible to control the UAV using a mobile phone.

Module Comparison

In order to choose which radio module to use as the data communication link between the laptop and the detection system, two modules were evaluated. The *XBee PRO-S2B* module and a *Radiocrafts* module called *RC1280HP*, both modules are mentioned in Section 3.8.3 and the specifications of the two modules are listed in Table 3.5. Both of these modules were available for use and therefore evaluated in this Thesis.

The price difference between the devices is not that high, but the range capabilities of the devices are. The *Radiocraft* module is superior when it comes to range with its 6 km over the *XBee*s 1.5 km. The *Radiocrafts* module uses more power when transmitting data than the *XBee* device does, but the *Radiocrafts* module makes up for it by having a lower receive and idle consumption. The idea of the communication link these devices are evaluated for is being a data communication link where the transmission from the UAV to the GCS will be the dominant factor. However, using the UAS as an autonomous detection system might mean that the transmissions from the UAV to the GCS are not happening constantly, but only occurs when a detection is made; therefore having a lower idle consumption might be desired. The data-throughput is clearly an advantage for the *XBee* with its 250 Kbps, but depending on the application the RF modules are used for, the 4.8 Kbps provided by the *Radiocrafts* module could also be sufficient. The capability of utilizing a mesh network topology will be an advantage during a SAR scenario happening in rough terrain, which makes the *XBee* better suited, as it possesses this capability. Their sizes are similar and they both run on 3.3 V, which makes them both possible to use together with the *BeagleBone Black*.

Even if the *Radiocrafts* module has a better range than the *XBee* module, the *XBee* family consists of several other module utilizing almost the same setup and interface. Another device that was found and evaluated is called *XBee-PRO 868* and has a range of 40 km LOS. This is much more than the *Radiocrafts* module and the module uses about the same amount of power during transmissions. The prices are almost the same for the two modules and therefore, utilizing the *XBee-PRO S2B* will be the better option during this Thesis because it will facilitate the possible changeover to this longer-range *XBee* unit if desired in the future.

7.5 Sensors

The sensor system used on this prototype UAS is the detection system discussed in Chapter 6. This system consists of an embedded computer called *BeagleBone Black* connected to an ordinary webcam. The *BeagleBone* is also connected to the *XBee-PRO S2B* in order to make communication with a laptop running a GCS possible, and it is connected to the *APM* autopilot in order to retrieve flight information. A Wi-Fi dongle could have been used to connect the device to a mobile phone. Since the *BeagleBone* also has the opportunity of communicating with the autopilot, this Wi-Fi setup could have been used in order to make the UAV track a SAR searcher. However, this is not investigated during this Thesis.

There are no other sensors used nor tested during this Thesis. The setup and connections of the *BeagleBone Black* is explained in Chapter 8. The unit was not placed on board the UAV during flight because of safety concerns regarding the equipment. Several crash landings had been performed during testing and since the equipment did not have much protection, it was not placed on the UAV during flight.

7.5.1 Sensors Evaluation

A human detection system was evaluated in Chapter 6 and proved to be feasible and could be part of a SAR UAS. By mounting the system on board the UAV, it could provide the SAR personnel with an autonomous human detection system capable of locating humans on the ground beneath the UAV. The detection system is low power and lightweight, and will thus be possible to use on board a UAV.

7.6 Ground Control Station

The GCS that was used is the *APM* provided application called *Mission Planner*. This GCS is elaborated some more in Section 3.2. The GCS allows the user to perform initial setup, configuration and tuning, simulations of flight, flight data logging, mission status updating, and mission planning. The GCS is a simple to use GUI application that can be freely downloaded from the Internet. A screenshot of the application is shown in Figure 7.4.

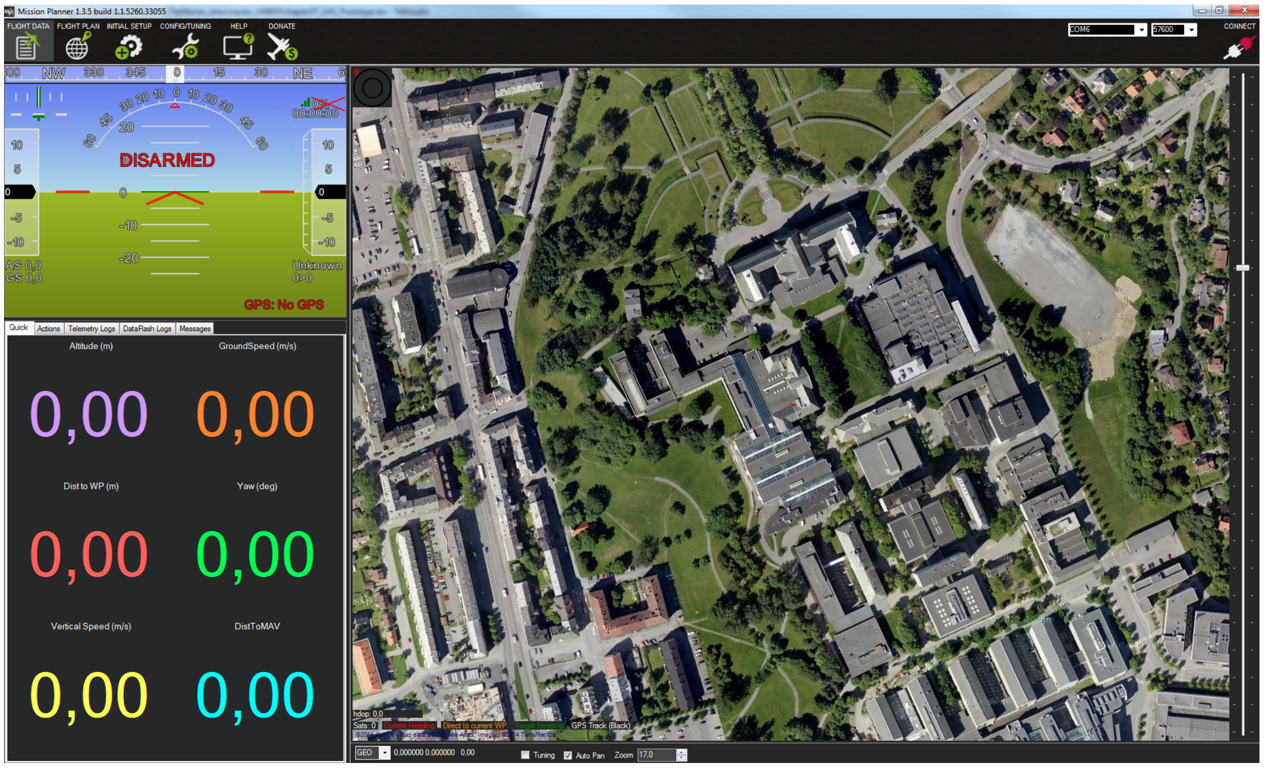


Figure 7.4: Mission Planner application

7.6.1 Ground Control Station Evaluation

Since the user-friendliness is one of the most important attributes of a GCS used as part of a SAR UAS, this feature of the *Misson Planner* application will be evaluated. The *Mission Planner* GCS is a somewhat user-friendly application. The application is easy to understand once it has been used a couple of times, but is perhaps not as intuitive at first glance. The separation between configuration tools and mission execution tools are not that obvious and sometimes it might be hard to separate them and knowing which are what. The mission planning and mission review screens are separated into two different windows, which is good in order to keep a small separation between mission planning and execution options. The system is easily used without much training and mission planning is done using GPS waypoint point-and-click configuration. Using a map of the flight area, GPS waypoints may be inserted and thus allowing the user to program an autonomous flight mission. The menus are somewhat intricate and thus the application can be a little confusing some times. The GCS application is also free to use and no cost is therefore spent on it, hence keeping the total cost of the UAS down.

The *Mission Planner* software allows pre-fetching maps of an area such that the flight system can be used even without Internet access. This is important if the SAR situation is performed in an area where no Internet access is available. However, it might be difficult to know in advance which area the search is going to be executed in and thus having a complete map service that can be downloaded in advanced might therefore be vital for SAR operations. It might therefore be necessary to implement a map service along with the GCS, thus having access to all maps even when offline.

The mapping feature of the GCS mentioned in Section 3.2.1 could be useful during SAR operations to see which areas are actually covered by the UAV during a flight. The GCS allows a visualization to be created containing the flight path of the UAV, and this flight path may be plotted in real-time during, and also after flight. This will allow the operator to see which areas that were covered by the UAV and thus plan the next flight accordingly. Unfortunately, the current version of the GCS application does not allow multiple UAV to be used controlled at the same time. A newer GCS application was mentioned in Section 3.2.1, but this application is still in the development stage and thus not properly usable. Having the possibility of utilizing multiple UAV during SAR will possibly be very relevant. Overall, the GCS is a well working, easy-to-use application once the user has gotten to know its capabilities.

7.7 Evaluation and Discussion

The prototype UAS presented in this chapter can be used for testing and thus be part of the decisions regarding the final design of a completed SAR UAS. The prototype should be tested in a real SAR scenario with the detections system on board to prove its capabilities and usefulness during SAR operations. When set up and tested properly, the prototype will be able to function as a valuable SAR tool. The detection system, discussed in Chapter 6, is developed to discover red clothed humans, but altering this system or utilizing some other detection methods, and also mounting the system on the UAV itself, the prototype UAS can be used for detecting and locating humans during SAR operations.

The way the prototype is set up allows it to be used as illustrated in Figure 5.3 where the UAV searches through a predefined area and reports its findings. By implementing a communication solution between the autopilot and a mobile phone, the UAS could be used as illustrated in Figure 5.4, where the UAV follows a member of the SAR crew and flies above their head reporting its findings. Utilizing this method, the detection system would have to include a method for distinguishing a member of the SAR crew from a missing person. For instance, since the UAV is following the searcher, it will necessarily know the location of the searcher and thus not classifying them as detections and therefore not reporting it when they are detected.

Chapter 8

Prototype UAS Setup and Testing

This chapter will present the setup and testing of the prototype UAS and its components, as presented in Chapter 7. Many of the devices used in this system are not originally intended to work together and thus some interfaces had to be created. The detection system was tested in Chapter 6 and was not mounted on board the UAV during the testing. Therefore, no testing of the detection system together with the rest of the prototype UAS was conducted in this Thesis.

8.1 Connecting the BeagleBone Black and APM 2.6

In order to have a fully functional human detection system, the *BeagleBone Black* used as the main processing unit of the detection system, needs to communicate with the autopilot, i.e. the *APM 2.6*. If the *BeagleBone* is also used for sending mission commands, this interface is also requires. Even though the detection system was not placed on board the UAV during the flights in this Thesis, the interface between the detection system and autopilot was made in order to make it possible to utilize the flight information in the detection application in the future.

This section will explain how the connection between the *BeagleBone Black* and *APM 2.6* can be setup, and a small test utilizing one of the methods mentioned will be explained.

8.1.1 Connection Possibilities

The *APM 2.6* communicates with other devices using a communication protocol called *MAVLink*, which is presented in Section 3.2.2. As mentioned in Section 7.4, the communication between the *APM 2.6* and the GCS application is accomplished using a radio module where the radio module takes serial communication from the *APM* and sends it over a wireless connection to a laptop running the GCS application. The radio module of the *APM* works as a transparent serial link, meaning that the *APM* and the GCS assumes they are communicating via directly connected serial link. The radio module accepts 3.3 V and 5 V serial communication, which will make the module work with the 3.3 V interface of the *BeagleBone Black*. The documentation of the *APM 2.6* does not list the voltage level of its serial communication, but since it is running on 5 V, it is likely to assume that the communication is also performed using this voltage level. A 5

V serial link will not be possible to interface directly with the *BeagleBone Black*, and hence the voltage level of the communication link will have to be stepped down if that is the case.

There are essentially two ways of connecting the *BeagleBone Black* to the *APM 2.6*. Having either a one-way communication link, or a two-way communication link. Methods for both approached will be presented and discussed in this section. Because of the limitations in the provided firmware of the autopilot, in that the device can only handle one external serial interface at a time, this had to be taken into consideration when deciding on which approach to use for the communication between the devices. The source code of the autopilot was not tempted to be altered during this Thesis and thus no second serial link was possible to be established.

One-way Communication

By only having a one-way communication link, allowing the *APM 2.6* to send information to the *BeagleBone Black*, the *BeagleBone* will not have the opportunity of altering the flight of the UAV since it cannot send any messages to the autopilot. However, if the *BeagleBone* only needs to get the flight information from the *APM* there is no need for the *BeagleBone* to talk to the autopilot, hence, the one-way link could be sufficient.

A one-way communication link between the devices can be accomplished in several ways. The easiest approach will be to allow the *BeagleBone* to sniff the packets sent by the *APM* to the GCS. All the information regarding the flight is sent in these packages and can thus be used by the *BeagleBone* for whatever purpose desired. A different approach would be to alter the code of the *APM* such that it can handle two serial links at the same time. This is not straightforward and could possibly be in the way of future updates of the *APM* firmware. A third approach would be to not use the radio module for transmitting the information to the GCS and solely send the information to the *BeagleBone*. Then if desired, the *BeagleBone* can forward these messages to the GCS.

The three approaches mentioned for providing a one-way communication link are illustrated in Figure 8.1. In the first illustration, the *BeagleBone* is sniffing the packets sent from the *APM* to its radio module. In the second illustration, the *BeagleBone* has a direct link to the *APM* and receives it packets over this connection. This would also allow different information to be sent to the radio module and the *BeagleBone*. The last illustration show that the *BeagleBone* is used instead of the radio module. Using this approach and only having a one-way communication link will prevent the *APM* from receiving information from the GCS, but the information could still be forwarded from the *BeagleBone* to the GCS, hence maintaining the link between the autopilot and the GCS.

Two-way Communication

By having a two-way communication link, between the *BeagleBone Black* and the *APM 2.6*, the *BeagleBone* could send commands to the *APM* and thus be part of planning the flight of the aircraft. This would be desired if the *BeagleBone* were used for tracking purposes and needs to inform the *APM* of where to fly. Another usage option can be used when a member of the SAR crew is connected to the *BeagleBone* using for example a Wi-Fi connection and informs the *BeagleBone* about their position. This position could then be used by the *APM* to plan its flight

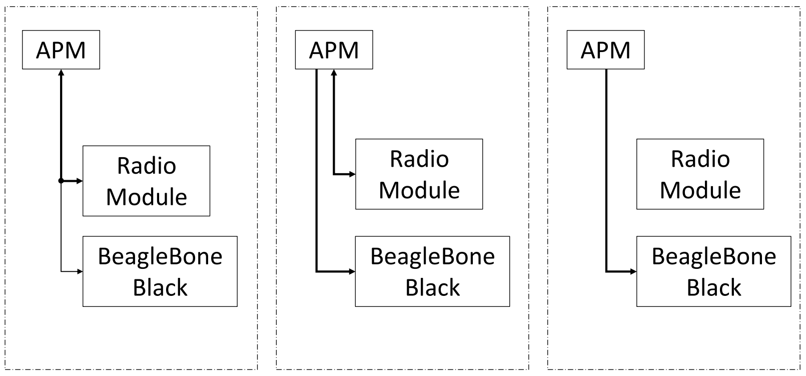


Figure 8.1: One-way Communication between BeagleBone Black and APM

and hence follow a member of the SAR crew from the air as illustrated in Figure 5.4.

Three ways can be mentioned for providing a two-way communication link between the *BeagleBone* and the *APM*. The first is to have the *BeagleBone* forward the packages from the *APM* to the radio module and the other way around. Using this approach, the *BeagleBone* can alter or add packets if desired and thus gain control over the aircraft. The second approach is to use the same method as in the one-way communication approach, which is altering the source code of the autopilot, but in this scenario also let the *BeagleBone* communicate with the *APM* as well. Finally, the third approach is the same as in the one-way method, only allowing the *BeagleBone* to send messages to the *APM* as well.

These three approaches are illustrated in Figure 8.2. The first illustration is by using the *BeagleBone* as a man in the middle and forwarding the packets sent to and from the *APM* and radio module. The second illustration is by having a separate serial link between the *APM* and the *BeagleBone*. The third and final illustration is by not using the radio module at all and allowing the *BeagleBone* to communicate with the *APM*. Information can then be transmitted from the *BeagleBone* to a GCS instead of using the radio module.

8.1.2 Connecting the devices

In order to test the communication between the *BeagleBone Black* and the *APM 2.6*, the simplest of the connection methods described previously, the one-way sniffing method was chosen and used. The *BeagleBone* was connected to the serial output from the *APM 2.6* between it and its radio module, and used to sniff the transmitted packets from the autopilot. This is the first method illustrated in Figure 8.1, and allows the *BeagleBone Black* to get the same information as the GCS application received from the autopilot, but did not allow the *BeagleBone* to communicate with the autopilot itself.

Unfortunately, the voltage level of the *APM 2.6* serial output was not known and thus had to be determined in order to prevent breaking the *BeagleBone Black*. The *BeagleBone* is only 3.3 V

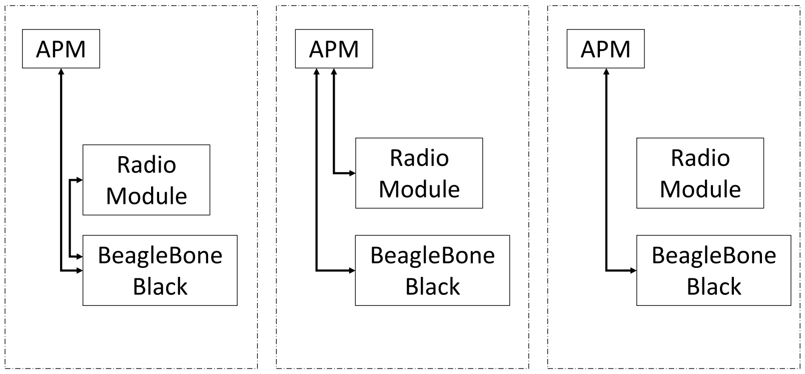


Figure 8.2: Two-way Communication between BeagleBone Black and APM 2.6

tolerant and thus would not be able to listen on a 5 V serial communication link. Because no information was found regarding the voltage levels of the *APM*s serial communication, an oscilloscope was used to determine it. The *APM* system was connected to its radio module, and the GCS was run on a laptop so that information was sent from the *APM*, via its radio module, to the GCS. When the system was up and running, the oscilloscope was used to measure the voltage levels of the *APM*s serial receive (Rx) and transmit (Tx). Figure 8.3 shows the measurement of the *APM*s transmit, i.e. the voltage level of the serial information sent from the *APM 2.6*. In the figure, the voltage per frame was set to 2 V, and the time scale was set to 500 ms per frame. The measurements showed that the voltage level of the *APM*s Tx was 5V and thus could not be connected directly to the *BeagleBone Black*.

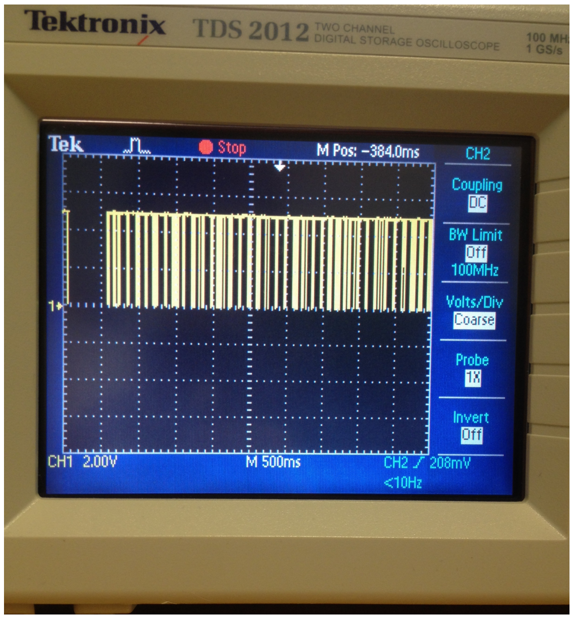


Figure 8.3: Oscilloscope Output of APM serial Tx

The transmission levels were measured to be 3.3 V for the receive (Rx) of the *APM*, i.e. the serial information being sent from the radio module, and as mentioned and illustrated in Figure 8.3, 5 V for the transmit (Tx) of the *APM*, i.e. the receive of the radio module. The serial input to the *APM* is thus 3.3 V, which makes the *APM* capable of receiving the 3.3 V serial communication of the *BeagleBone Black* if desired. Because the voltage level of the *APM*s Tx was 5 V, it had to be downscaled in order to be connected to the *BeagleBone Black*. A splitter cable was produced in order to split the serial communication cables between the *APM* and the radio module and make it possible to connect it to the *BeagleBone Black*. A voltage level converter from 5 V to 3.3 V was also made making it possible for the *BeagleBone Black* to listen on the transmissions from the *APM* without breaking. Figure 8.4 shows how the splitter cable and level converter is used to extract the transmit wire of the *APM 2.6*, perform the level conversion, and connecting it to the *BeagleBone Black*. The ground wire from the *APM* is connected to the *BeagleBone Black* to make sure that the communication is interpreted correctly. The pins used on the *BeagleBone Black* were chosen because they allow serial communication to be performed on them.

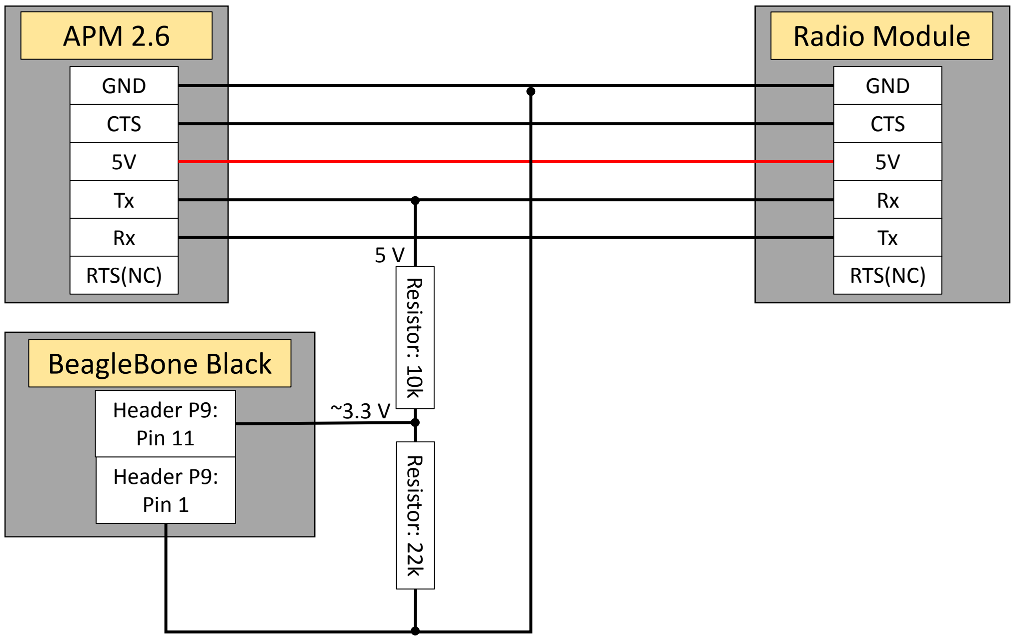


Figure 8.4: BeagleBone Black Sniffing Setup

8.1.3 BeagleBone Black Implementation

After the devices were connected, an application was written in order for the *BeagleBone Black* to interpret the serial communication sent by the *APM* to its radio module. The *BeagleBone Black* was setup as explained in Appendix A and used as this for the remainder of the tests in this Thesis. The communication application uses the same setup that a GCS uses for interpreting the information sent by the *APM*. The *BeagleBone Black* is as explained connected as shown in Figure 8.4 and listens on the packets sent from the *APM* to the GCS. In order to use serial communication on the *BeagleBone Black* the Unix API for terminal I/O *Termios* was used [Termios].

This API is used for configuring the terminal ports and thus enabling the serial communication to be performed correctly.

The messages from the *APM* are as mentioned encoded and sent using the *MAVLink* protocol and library. Therefore, in order to interpret the messages intercepted by the *BeagleBone Black*, the following steps were taken:

1. *MAVLink* header files were downloaded and saved on the *BeagleBone Black*
2. Serial communication was enabled on the *BeagleBone Black*
3. An application was written to interpret the messages
4. The devices were connected and a test was run

Including MAVLink headers

In order to interpret the packets or be able to alter them, the *MAVLink* headers needed to be included on the *BeagleBone Black*. After they were included, messages could be interpreted using the function "$mavlink\_parse\_char()$" which is provided along with the definitions of the messages in the *MAVLink* header files. Some information regarding *MAVLink* is mentioned in Section 3.2.2. The setup of this library on the *BeagleBone Black* can be found in the comment section of the serial communication application code in Appendix A.6.

Serial Communication Setup

The GPIO pins on the *BeagleBone Black* can perform many functions and the operating system has to be told what they are going to be used for. The pins on the *BeagleBone Black* was setup to enable UART4 by calling the command "*echo BB-UART4 > /sys/devices/bone_capemgr.*slots*". This commands sets up pin 11 on the P9 header to act as an UART serial receive (Rx) for the device. Pin 13 on the same header is also setup as an UART serial transmit (Tx) for the device, but is not used during this test. The P9 header is the pin header nearest to the power jack input, an illustration can be found at the manufactures website.

Application

The serial communication application code can found in Appendix A.6, Listing A.7, and the code utilizes the serial API *Termios* for performing the serial communication. The code is written as a test application and the interpretation of the messages could have been moved to a separate thread if the *BeagleBone* was going to perform other tasks at the same time as interpreting the messages. For test purposes, having the interpretation done in the main loop was sufficient.

8.1.4 Application Testing

The setup from Figure 8.4 was used and the application listed in Appendix A.6 was run on the *BeagleBone Black*. At the same time, the GCS was run on a laptop in order to get the *APM* to transmit its packets, and to verify that the messages received by the *BeagleBone* were correct. The *BeagleBone* was connected to a laptop using a USB cable in order to start and stop the application, and to read the results of the decoding.

Results

Listing 8.1 shows a single packet output from the interpretation of the application. Line 1 is a printout of the system ID, component ID, message length, and message ID. Line 2 to 6 are the bytes of the packet in hexadecimal numbers, and line 7 is a printout of the label decoded from the message ID using the common *MAVLink* message overview [MAVLink]. Line 8 shows an example of extracting the attitude information from the packet payload using the included *MAVLink* functions, which can be seen in Listing A.7, lines 159-163.

Listing 8.1: Serial Communication Application Output

```
1 Serial -> BBB: SYS: 1, COMP: 1, LEN: 28, MSG ID: 30
2 0xfe 0x1c 0xd9 0x01 0x01 0x1e 0x95 0x2b
3 0x00 0x00 0x1e 0xd4 0x00 0x3b 0x07 0x6c
4 0x4a 0x3b 0xa4 0x56 0x0c 0x40 0x66 0x65
5 0x7b 0x3a 0xcc 0x64 0x0a 0x3b 0x95 0x78
6 0xb5 0xba 0xed 0xbb
7 Attitude received
8 Pitch: 0.003089, yaw: 2.192788, roll: 0.001966
```

The *MAVLink* packet anatomy, as explained and illustrated in Section 3.2.2, is visible by evaluating the bytes that are received. By examining byte number two from the packet in Listing 8.1, which is 0x1c, it can be seen that the printout of the message length, which was 28, is the same as this hexadecimal number 1c. The message ID, which is packet byte number six, is equal to 1e, which is the same as the message ID printed by the application, 30. This shows that the interpretation of the messages is working. The attitude information printed by the application was also verified by noting the value of yaw on the GCS that was run on the computer and comparing this to the yaw value received by the serial application run on the *BeagleBone Black*. These values matched and proved that the system can be used for extracting the flight information from the *APM 2.6* to the *BeagleBone Black*.

8.1.5 Discussion

The approach used in this test example for extracting the flight information to the *BeagleBone Black* was perhaps the simplest of the communication methods presented. However, the approach was used solely to show that it was possible to get the information to a detection system and could thus be used by a detection application if desired. The test proved that the setup worked and that it is possible to extract the autopilot's flight information on the *BeagleBone Black*. Having this information available on the *BeagleBone Black* would make it possible to use it during an object localization as was missing in the localization tests performed in Section 6.6.

It was also tested to connect the *APM 2.6* directly to the *BeagleBone Black* using a USB cable. This worked fine and would be an option to use if all information from the GCS is sent to and from the *BeagleBone* instead of the *APM* itself. Using this approach would allow a Wi-Fi connection to be setup between the *BeagleBone* and a mobile phone, thus allowing the autopilot to be controlled by the mobile phone. This would as explained, be smart if the UAS was used to follow a member of the SAR crew from the air.

8.2 XBee Range Test

In the datasheet of the *XBee-PRO S2B*, the range of the module is listed to be roughly 1500 m LOS for the international version, which is the one allowed to use in Europe. More information regarding the module is listed in Section 3.8.3. In order to verify that the *XBee* module could be used as a communication link between a UAV and GCS during SAR operations, the module was tested using the manufacturer provided evaluation software. The test and configuration software, called *XCTU*, is a free, multi-platform application with a simple to use graphical interface for configuration and testing of *XBee* devices [Digi International Inc., b]. The software allows a range test to be run between two radio modules on the same network and gives information regarding the packet loss in the network and the receive signal strength. This can in turn be used for evaluating the network.

8.2.1 Setup

The *XBee* range test was performed by placing two *XBee* modules apart from each other and altering the distance between them. The first device was placed in the second floor, outside the window of a house. The other unit was placed on top of a car and driven away to get different lengths between the devices. The second floor was used to get some elevation for one module as to simulate the flight of a UAV. One *XBee-PRO S2B* was connected to a laptop using a USB interface module, and the other was placed and powered by a battery at the other location. The module that was not connected to the laptop had the Rx and Tx input/output connected to each other so that the packages were sent from this module back to the source. Thus if the *XBee* module that was connected to the laptop was told to transmit some letters, these would be transmitted back again from the other module once it received them.

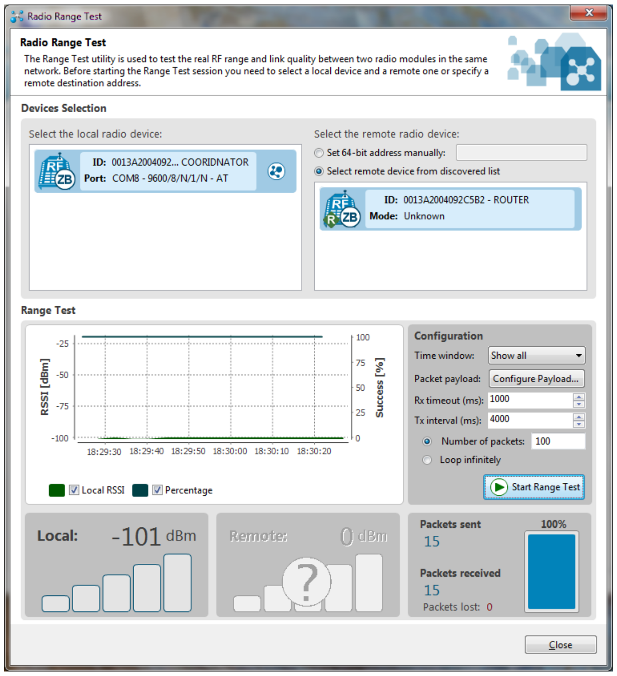
The provided *XCTU* software was used to analyze the network, which was setup as a point-to-point network between the two modules. One module was a network coordinator and the other was a router.

8.2.2 Results

Figure 8.5 shows the test result from a test performed while the *XBee* modules were roughly 720 m apart. Fifth teen package were sent and no packages where lost. The receive signal strength was estimated to be -101 dbm which is the same as 0.1 pW.

Figure 8.6 shows the test results from a test performed at 1.8 km apart. This is 300 m longer than the maximum range specified in the datasheet of the modules. As seen in the figure, at this length no packages were lost and the receive signal strength was also this time estimated to be -101 dbm.

A last and final test was performed at roughly 2 km away. During this test, about half of the packages were lost showing that the link was not stable at this distance.


Radio Range Test

The Range Test utility is used to test the real RF range and link quality between two radio modules in the same network. Before starting the Range Test session you need to select a local device and a remote one or specify a remote destination address.

Devices Selection

Select the local radio device:

ID: 0013A2004092... COORIDNATOR
Port: COM8 - 9600/8/N/1/N - AT

Select the remote radio device:

( ) Set 64-bit address manually:
(●) Select remote device from discovered list

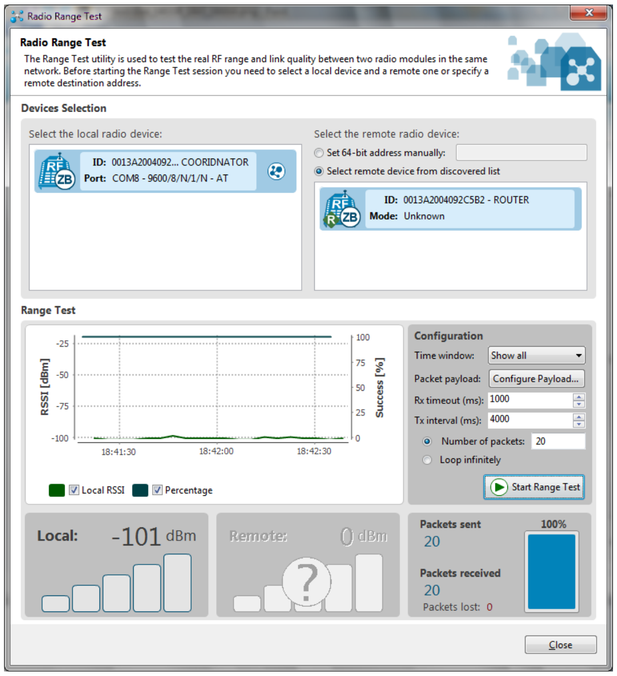
ID: 0013A2004092C5B2 - ROUTER
Mode: Unknown

Range Test

Configuration
Time window: Show all
Packet payload: Configure Payload...
Rx timeout (ms): 1000
Tx interval (ms): 4000
(●) Number of packets: 100
( ) Loop infinitely

Local: -101 dBm
Remote: 0 dBm
Packets sent 15
Packets received 15
Packets lost: 0
100%

Figure 8.5: XBee test at 720 m


Radio Range Test

The Range Test utility is used to test the real RF range and link quality between two radio modules in the same network. Before starting the Range Test session you need to select a local device and a remote one or specify a remote destination address.

Devices Selection

Select the local radio device:

ID: 0013A2004092... COORIDNATOR
Port: COM8 - 9600/8/N/1/N - AT

Select the remote radio device:

( ) Set 64-bit address manually:
(●) Select remote device from discovered list

ID: 0013A2004092C5B2 - ROUTER
Mode: Unknown

Range Test

Configuration
Time window: Show all
Packet payload: Configure Payload...
Rx timeout (ms): 1000
Tx interval (ms): 4000
(●) Number of packets: 20
( ) Loop infinitely

Local: -101 dBm
Remote: 0 dBm
Packets sent 20
Packets received 20
Packets lost: 0
100%

Figure 8.6: XBee test at 1800 m

8.2.3 Evaluation and Discussion

The testing performed with the *XBee-PRO S2B* modules proved that the range of these modules were at least what was specified in their datasheets. The modules will provide a sufficient range for some SAR scenarios, but a longer-range module might be needed for others. If that is the case, these modules will have to be replaced. The good thing about using these modules is that the *XBee* family provides other modules that can be used, and some with more range than the ones used during this Thesis. Interchanging the modules for another *XBee* family module should not provide much of a challenge. The *XBee* module mentioned in Section 7.4 called *XBee-PRO 868* could be an option to use as part of a SAR UAS due to its long range.

During the tests performed, the LOS between the modules was somewhat blocked by some trees and bushes. This might have affected the transmissions and thus the test results. A test should be performed in air on board a UAV, in a real SAR scenario to verify the usability of the modules. The range test performed at 2 km might have given a different result if one module was located in the air and the link might then have proved functional at this range as well.

To extend the range of the modules and provide a better LOS between them, a fishing rod can be used to elevate the module that is place near the GCS. This would allow for a better LOS and thus increasing the range of the communication link. Using these modules in an area with rough terrain might mean that a direct LOS is not possible to achieve for the entire region. Hence, elevating the module near the GCS could allow some regions to obtain a LOS with the GCS as well.

The data throughput of the modules were not tested and a test should be performed testing this feature as well. Depending on the purpose the modules are used for, the data throughput might be an important feature to evaluate. If they are only used for simple control commands, a low data throughput could be sufficient. The *XBee* modules used will probably work fairly well as part of a SAR UAS. Their range are not very long, but as explained, if the UAV flight is autonomous or multiple UAV are used together with a mesh network approach, the communication link range does not need reach that far.

8.3 Connecting the BeagleBone Black and XBee-PRO S2B

In order to test the interfacing possibilities between the *BeagleBone Black* and *XBee-PRO S2B*, a simple communication application was developed and tested. The application code for this is listed in Listing A.8 in Appendix A.7. The code utilizes the same serial communication API as before, *Termios*, and the POSIX thread library, which is mentioned in Section 3.7.

The interfacing between the *BeagleBone Black* and the *XBee-PRO S2B* modules is straightforward since they both utilize 3.3 V serial communication voltage level, hence the output and input between the devices can be directly connected to each other. The pin mapping done in the *APM 2.6* and *BeagleBone* connection test was also used in this application. However, instead of utilizing UART4, UART1 was used and can be setup by calling the command "*echo BB-UART1 > /sys/devices/bone_capemgr.*/slots*" in the terminal of the *BeagleBone*. The UART1 will be listed as "*/dev/ttyO1*" and the transmit (Tx) will be present on pin 24 on the P9 header, and the receive (Rx) present on pin 26 on the same header. A different UART was used so that if both applications made in this Thesis, which uses serial communication, were run at the same time, their pin mapping would not collide.

The connection setup of the *BeagleBone Black*/*XBee* interface application can be seen in Figure 8.7. The *BeagleBone Black* is connected to a laptop running a ssh session over a terminal. One of the *XBee* modules is connected to the *BeagleBone*, and the other module to another laptop running the *XCTU XBee* evaluation software.

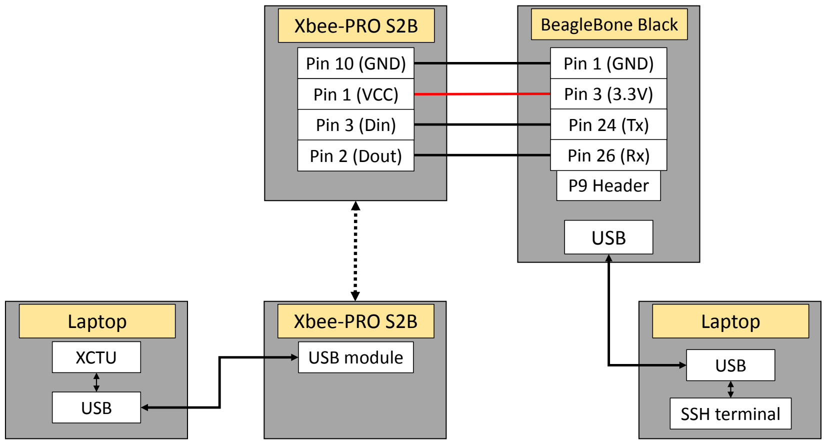


Figure 8.7: BeagleBone Black/XBee interface application connection setup

The coordinator *XBee* module was connected directly to a laptop and the *XCTU* software was run in order to use the terminal setup of this program. Using this software, input could be typed in a terminal and this would be sent to the module listening on the network. The application running on the *BeagleBone Black* was setup to have one thread handling the serial communication input bytes from the *XBee* module and writing each byte to the terminal of the *BeagleBone*. Another thread was setup to handle the transmissions from the *BeagleBone* to its connected *XBee* module, and was setup to take input lines from the terminal of the *BeagleBone* and transmitting the entire line, via its connected *XBee* module, to the other one when the "enter" key was hit. The units were placed close together so that both laptops could be operated at the same time.

8.3.1 Results

Listing 8.2 shows the output of running the application after connecting the modules as shown in Figure 8.7. The text "Hello XBee World" was typed in the *BeagleBone Black*s terminal, line 3 in Listing 8.2, thus transmitting this to the *XBee* module connected to the other laptop and getting the input displayed on the terminal there. Then the text "Hello right back at you" was entered in the terminal of the *XBee* not connected to the *BeagleBone*, and as seen in the listing, the text appears in the terminal of the *BeagleBone* one character at a time, as expected.

Listing 8.2: Output BeagleBone Black/XBee Interface Application

```
root@beaglebone~xBee# .programExe
Hello XBee world
Input Hello XBee world
 size 17
Rec_byte H
Rec_byte e
Rec_byte l
Rec_byte l
Rec_byte o
Rec_byte
Rec_byte r
Rec_byte i
Rec_byte g
Rec_byte h
Rec_byte t
Rec_byte
Rec_byte b
Rec_byte a
Rec_byte c
Rec_byte k
Rec_byte
Rec_byte a
Rec_byte t
Rec_byte
Rec_byte y
Rec_byte o
Rec_byte u
```

8.3.2 Evaluation and Discussion

The *XBee-PRO S2B* module was easy to setup and connect with the *BeagleBone Black*. Since they both use the same serial communication voltage level, no level conversion was needed when connecting them. The *XBee* module that was communicating with the *BeagleBone Black* was also powered by the *BeagleBone* as illustrated in Figure 8.7, and the *BeagleBone Black* proved to provide enough power for the transmissions to be accomplished.

The testing was performed indoors with the two *XBee* modules placed close together. Since the test was only done in order to test interfacing possibilities between the devices and verify that such a connection would be possible, the test was not performed at any longer distances. The application worked and proved that the *XBee* modules may be used together with the *BeagleBone Black* in order to provide a data link back to a GCS during SAR operations. The *BeagleBone Black* does have a limit regarding the power output that the device can provide to other devices. Usually, "capes" are used for providing the power needed by other devices as to not overload the *BeagleBone Black* itself. Using these "capes", which are placed on top of the pin headers of the *BeagleBone*, it is possible to provide the devices connected with a different power source than directly from the *BeagleBone*. This might be needed if some high power communication modules are used for realizing the data link of a SAR UAS.

8.4 APM 2.6 Setup

In order to evaluate the *APM 2.6* system for use as autopilot during SAR, the system had to be setup on board a UAV and test flights performed. The following sections will describe the setup and testing of the *APM 2.6* autopilot for use in the prototype UAS discussed earlier in this report. The corresponding results and discussions will follow directly after the different tests are presented.

The prototype UAS was presented and discussed in Chapter 7 and the setup in these tests were done accordingly. As mentioned, the autopilot was mounted on, and thus replaced the flight system used on the prototype UAS created by Vegard Hammerseth, which was also used in the pre-study of this Thesis.

8.4.1 Setup

Figure 8.8 shows the plane with almost all its components connected. Mounted on the plane were the *APM 2.6* autopilot with its radio module, the autopilot GPS/compass module, an ESC, a brushless motor, two batteries, a power module that powered the *APM* and measured the remaining battery voltage, two servos, a camera mount, and a *DX7s* radio controller receiver called *AR8000*. As seen in the image, the winglets (the outermost part of the wing) were colored to facilitate the users experience during flight [Hammerseth, 2013]. The right winglet was colored with green, and the left with red. This approach is adapted from boats and made it easier for the pilot to know the orientation of the plane whilst in the air. Lights could also have been mounted to make it stand out in the dark, but this was not done during these tests.

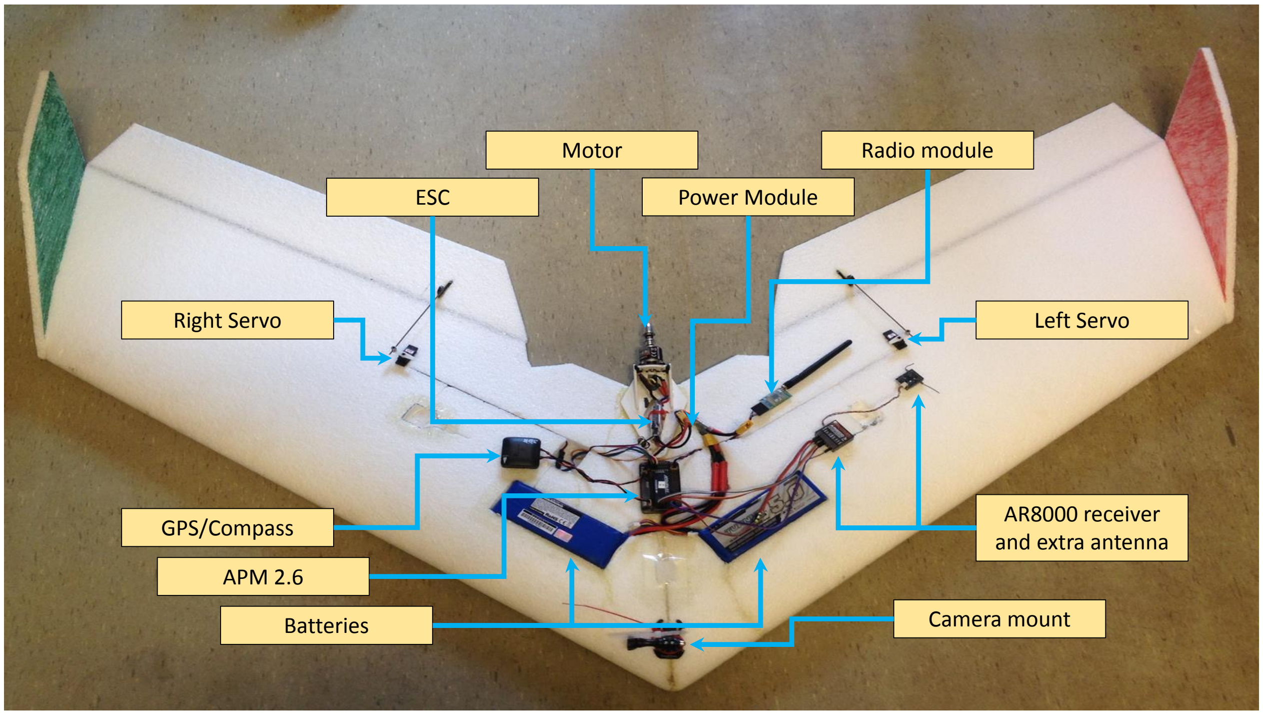


Figure 8.8: UAV with APM setup

All cables were twisted around themselves in order to reduce the electromagnetic interference by other equipment. The GPS/compass module was placed as far away from the ESC as possible because of the electrical noise created by the ESC. The *APM* itself was placed as near the center of gravity as possible in order to reduce the vibrations on the device. The wire antennas of the *AR8000* receiver was directed such that the three different wires pointed in all axis directions, that is in the x, y, and z directions. Most of the components were fasted using double-sided tape and some super glue. The batteries were mounted in their place using Velcro to facilitate the procedure of their removal and insertion when charging was necessary, and the camera mount was fastened using cable ties. The motor is fastened to a motor mount with some screws and glue, and the mount itself was glued to the airframe. The servos were also kept in place by using super glue.

Figure 8.9 illustrates the connections on board the UAV and shows the components that were present on the UAV during the flight tests. The only components not shown are the camera and its mount, and the airframe itself.

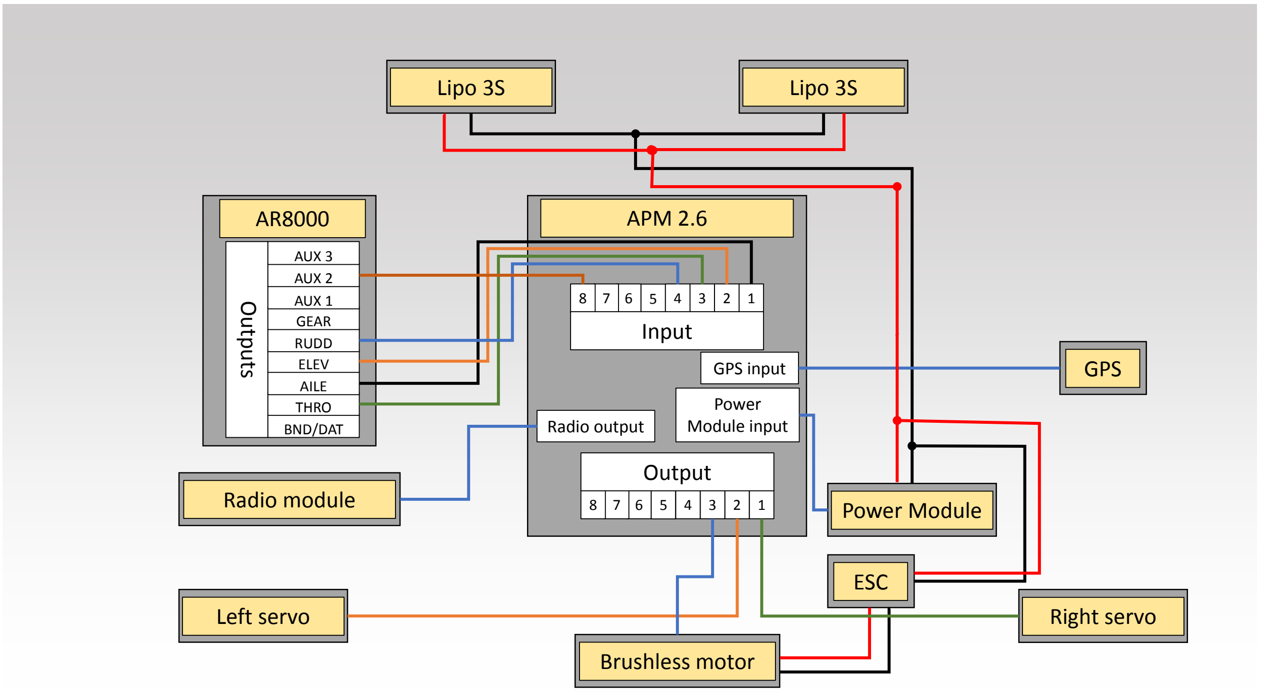


Figure 8.9: UAV Component Connections

Pre-Test Flight

After the autopilot was installed on the plane, it was setup using the calibration wizard provided by the GCS application of the *APM* system, *Mission Planner*. Performing some steps following a well explained wizard sets up the plane for flight. This setup will have to be redone if the autopilot is repositioned. The tuning parameters of the PID controller that controls the aircraft when in autonomous modes were chosen based on a similar model found online. The *APM* community provides some configuration files for commonly used airframes which simplifies the process of tuning the aircraft. One of these configuration files, created for a plane called *TELINK Toro900 flying wing*, was used during the testing.

Before each flight, the pre-fetching capability of the GCS was used to get a map of the area that the flight would be performed in. This allows the map to be used in the field even when the laptop did not have an Internet connection. The GCS allows the user to specify a map area that is temporarily downloaded and saved on the computer for use during the mission. When powering the autopilot at the flight location, the plane was left motionless on the ground as level as possible until the three LEDs on the autopilots had stopped flashing. This allowed the autopilot to calibrate the sensors and be ready for flight. It was also necessary to wait for a GSP lock if autonomous flight was going to be executed. The "home" GPS locations will be set once the autopilot is turned on and achieves a GPS lock, and it is therefore important to turn it on in a location that is desired to be the "home" location. The "home" location is where the plane will return to if a loss of communication occurs and the "return-to-launch" safe action is activated.

8.5 Flight Test 1

The primary goal of the first flight test was to verify that the plane was indeed capable of flying. Since the autopilot had never been used before, the test would also be used so that the pilot could get familiar with the systems capabilities. Following the guides of the *APM* community, the behavior of the control surfaces in Stabilize or FBWA mode during the first flight should be tested and observed. The control surfaces should move the plane back to level after pitch or roll input has been applied and released. If this is not performing well, tuning the gains of the autopilots PID controllers needs to be done. Take-off should be performed in Manual mode and whilst in the air, the mode should be switched to either Stabilize or FBWA and the behavior observed. The take-off was accomplished using hand launching, and landing using belly landing on a soft surface during these flight tests. Another thing to determine during the first test was if the plane was capable of carrying a load such as the *BeagleBone Black* along with a webcam, i.e. if it was capable of carrying a detection system. This was proven for this airframe by Vegard Hammerseth, but since a new autopilot system has been introduced, it should be retested.

The radio module was not connected to the *APM* during Flight Test 1 and thus transmitting flight data in real-time to the GCS was not possible.

Test summary for Flight Test 1:

- Test if the plane is capable of flying
- Learn to utilize the autopilot system and fly properly
- Test either stabilize or FBWA mode and observe the behavior of the UAV
- Determine if the UAV is capable of carrying the *BeagleBone Black* and a webcam during flight

8.5.1 Results and Discussion

Pre-test Flights

The first couple of pre-test flights verified the reasoning made in the pre-study of this Thesis that the need for an automated take-off routine is essential for a SAR UAS. The plane was launched several times with the result being either a broken propeller or broken frame. The need for an autonomous launching system is essential for a SAR UAS, and it needs to be developed for such a UAS in order for it to be used properly. If the plane is to be used as part of a SAR mission, failed take-offs are not an option.

The fact that the take-off sequence should be automated is backed-up by a video released by the Portuguese navy [METRO]. In this video, some persons from the Portuguese navy are trying to launch a drone from a pier and the result being that the UAV flies straight into the water. Even though the video does not tell, the most likely cause of this failure was probably a human error during the take-off, and the video shows that even with trained personnel, the human factor is still possibly the most likely error source during a UAV take-off. Having automated the take-off will result in a much more reliable system and will thus be more applicable for use during SAR operations.

The positive sides of having performed some failed take-offs is that the robustness of the frame was tested and proven capable of handling some rough treatment. The result of one of the failed take-offs can be seen in Figure 8.10 and shows the aircraft broken into two pieces. Special super glue had to be purchased so that the plane could be glued back together. If normal super glue was used, it would melt the foam of the frame and it is therefore not applicable. The super glue used, called *UHU Por*, is very strong and the plane was able to fly even after having been broken in half, two times.

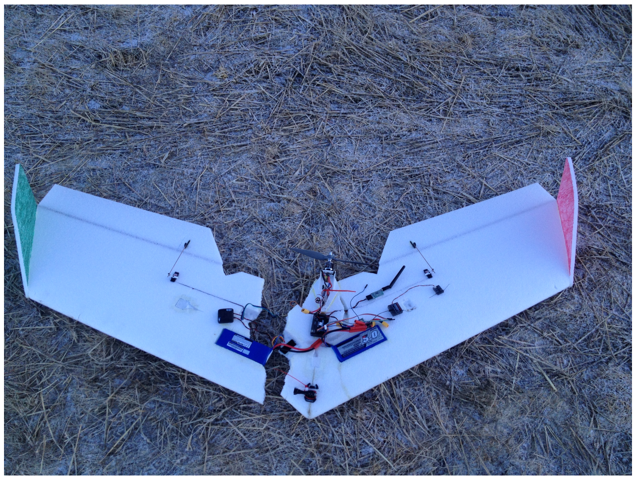


Figure 8.10: UAV After Failed Take-off

First Flight

After the first couple of miss-flights, the plane was successfully launched and hence proved capable of flying with the new autopilot installed. The launching of the UAV was accomplished using the hand launching method and landing was performed doing a belly landing on a soft spot on the ground. No extra equipment was fastened on the plane except that which was needed for the *APM 2.6* to function. Manual control mode was used and the plane was flown for a little while and then landed in order to test the controllability of the system.

Second Flight

A goal of Flight Test 1 was to determine if the *BeagleBone Black* and a webcam could be fastened on the UAV and see if the UAV was still able to fly. It was decided that fastening a *GoPro Hero 3* in the front of the plane and doing the image processing afterwards was a safer approach than mounting the *BeagleBone Black* and webcam directly on the UAV [GoPro]. Since several crashed had occurred, the robust casing of the *GoPro* provided a very sought after protection from impacts.

The *GoPro* casing was mounted in the front of the plane and the plane was successfully launched using hand launching. Instead of having the actual *GoPro* installed in the casing, a small bag with matching weight was used. This was done because if the plane were to crash, the camera would not be harmed. Since the UAV was successfully launched, flown, and landed, it was decided that it would be safe to mount the *GoPro* camera on the plane for the next test.

This second flight proved that the UAV was capable of carrying a payload such as the *GoPro*. The combined weight of the webcam and the *BeagleBone Black* used in this Thesis is roughly the same weight as the *GoPro* with its mount and fastening screw. The plane will therefore probably not have any problems carrying the devices. The weight of the webcam is 227 g and the weight of the *BeagleBone Black* is 40 g, combined to 267 g. The weight of the *GoPro* device used, with its casing, is 170 g, but also including the *GoPro* mount and fastening screw makes it somewhat heavier than the *BeagleBone* and webcam combined.

Third Flight

Since the autopilot was configured with a standard configuration file, its behavior had to be tested to see if any tuning was necessary. The final autopilot tuning needs to be performed after all the components that are supposed to be on the plane are actually mounted on the plane. However, in order to test the capabilities of the autopilot it was decided to try to tune the plane with only the *GoPro* mounted as extra equipment on the UAV. The *GoPro Hero 3* was thus mounted in its casing and attached to the camera mount on the plane. Then the plane was once again launched using hand launching and flew fine. The plane did not display any different behavior than before the mounting of the *GoPro* as was suspected and proved that the plane would be capable of carrying this payload.

During this third test flight, the Stabilize mode of the *APM* autopilot was tested. Since the *APM* radio module was not connected to the system, the RC controller (DX7s) was used to change the flight mode of the system, and once the Stabilize mode was entered, the plane displayed some porpoise (dolphin like) behavior and the mode was quickly aborted. Figure 8.11 shows a plot of the pitch angle extracted post-flight from the flight log of the autopilot. The labels along the "Line Number" axis explains which mode was used at that particular time. The porpoise behavior can clearly be seen in the graph, and it starts when the Stabilize mode is entered and stops once the mode is switched back to Manual.

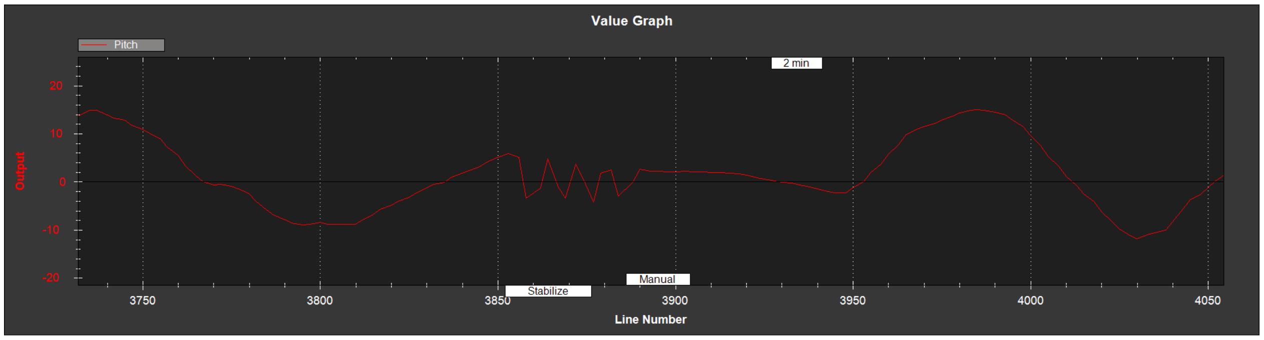


Figure 8.11: Pitch Plot, Third flight of Flight test 1

The graph in Figure 8.11 suggests that the parameters of the PID regulator for the pitch angle of the autopilot system were not good. The value of the P gains was set to 1.9 pre-flight, as given by the standard configuration file, and halved to the value of 1 post-flight following guidance from the *APM* community.

8.5.2 Flight Test 1 Summary

The autopilot performed well and the UAV was able to get airborne without too much hassle. Three successful flight were performed during Flight Test 1 and all the goals of the test were met. The plane was capable of flying. The autopilot system was utilized and the UAV was flown somewhat properly, although the take-offs and landings were a bit rough. The stabilization mode of the autopilot was tested and its behavior observed. The plane was proved capable of carrying a load with a weight similar to that of the *BeagleBone Black* and a webcam.

During these first couple of flights, the autopilot was not connected to the GCS via the radio module provided by the *APM* system. The log files were collected for analysis using a USB cable and transferred post-flight from the onboard memory of the autopilot. The plot in Figure 8.11 shows the capabilities of utilizing a system such as the *APM 2.6*. All the analysis tools are already provided by the GCS application and no programming had to be done in order to get this plot. It was simply a matter of downloading the log files from the autopilot, and choosing which plot to graph. By not having these tools, it would have been difficult to know if the porpoise behavior actually was cause by the switching between the flight modes or not. The plot clearly shows that the behavior started once the Stabilize mode was entered, and stopped once it was changed back to Manual.

8.6 Flight Test 2

The goal of Flight Test 2 was to get video footage of an "injured" dummy place on the ground beneath the UAV to see if it would be visible from the air, and possibly do some computer vision processing on the video afterwards. The radio module capabilities of the *APM* system should also be tested. The video was to be collected using the *GoPro* camera that was mounted during the first flight test. The plane was launched by hand and take-off was performed in Manual mode since the autopilot system was not set up for autonomous flight. Landing was done using the belly landing approach on a soft ground spot. The stabilization mode was tested again to see if any changes had occurred when altering the PID parameters.

The *APM* radio module was connected during this test and the GCS application used to observe the flight data in real-time.

Test summary for Flight Test 2:

- Get images from the air of an "injured" dummy placed on the ground beneath the UAV
- Test the radio module of the *APM* system
- Get more comfortable flying the aircraft
- Test stabilization mode and observe if any changes has occurred since altering the parameters of the PID controller

8.6.1 Results and Discussion

First Flight

Unfortunately, as many times before, the take-off during the first flight failed and resulted in a broken propeller mount as well as breaking the propeller and the testing this day had to be aborted.

The reason for this failed take-off was not known at the time, but may have been due to a damaged motor mount or simply due to a human error. Everything appeared to be correct with the UAV, but the plane did not gain altitude after being tossed during the hand launching.

Second Flight

A new propeller was acquired prior to the second flight and the motor mount was glue back together and fastened tightly to the airframe. The purpose of this second flight was solely to test if the plane could get airborne after the previous failed take-off and component breaking. The plane performed well and was able to fly satisfactory and land with no problems. Flight Test 2 could thus be re-continued.

Third Flight

After a successful second flight, the third flight was done trying to meet the goals of Flight Test 2. A laptop was connected to the plane using the *APM* radio module and the *GoPro* was placed in its casing and mounted on the aircraft. The plane was hand launched into the air and flew as expected in Manual mode. The stabilization flight mode FBWA was tested in-flight, but the response was absent or too slow to be noticed. This is probably because the gains were set to low for the system to react and should thus be increased to get a response.

Flight logs were collected by the GCS during flight, and post-flight from the internal memory of the *APM*. A dummy was placed on the ground beneath the flight area and was captured on video by the *GoPro*. The video showed that a human would be visible on the ground beneath the UAV, and a UAV can thus be used for detection purposes during SAR operations. An image from the video displaying the dummy is shown in Figure 8.12. The pilots are also visible in the background of the image. As one can see, the red jacket is clearly visible and stands out from the environment similar to what a human heat signature would do in a thermal image.

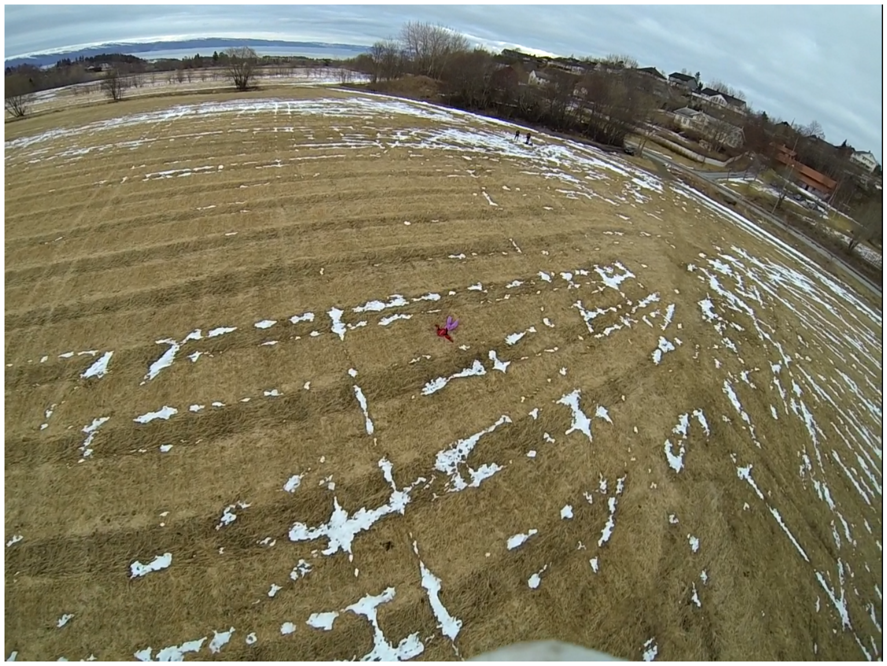


Figure 8.12: Image extracted from GoPro Video, camera view of dummy

Since the flight logs of the autopilot were collected both during flight, and downloaded post-flight, the capabilities of the *APM* systems mission analysis tools was tested once again. Figure 8.13 shows the flight path collected from the flight log of the third flight. The path plotted is the final part of the flight just before the landing was performed. The list on the left side gives the opportunity of selecting which part of the flight that should be illustrated. The *APM* system provides the user with a ".kmz" file which can be used for plotting the flight path in *Google Earth* and the flights can thus be reviewed post-flight [Google Earth]. The different flight modes are marked with different colors on the map given that several modes are viewed at the same time, and it is possible to see the rotation of the aircraft at all points of the flight path by highlighting the folder named "Planes" in the selection menu.

8.6.2 Flight Test 2 Summary

During Flight Test 2, only two successful flight were performed. Due to the failed take-off of the first flight, some time was spent on repairing the plane and thus less time on the actual testing. Images were collected of the "injured" dummy beneath the UAV and showed that a human could be visible in an image captured from a UAV. The radio module of the *APM* system was also tested and proved to be functional. Flight data was viewed during flight and the GCS could have been used during flight for changing the flight mode of the autopilot. The operator got more familiar with the aircraft whilst in the air, but take-offs and landings were still troublesome to perform. The stabilization mode FBWA of the autopilot was tested and proved that the porpoise behavior

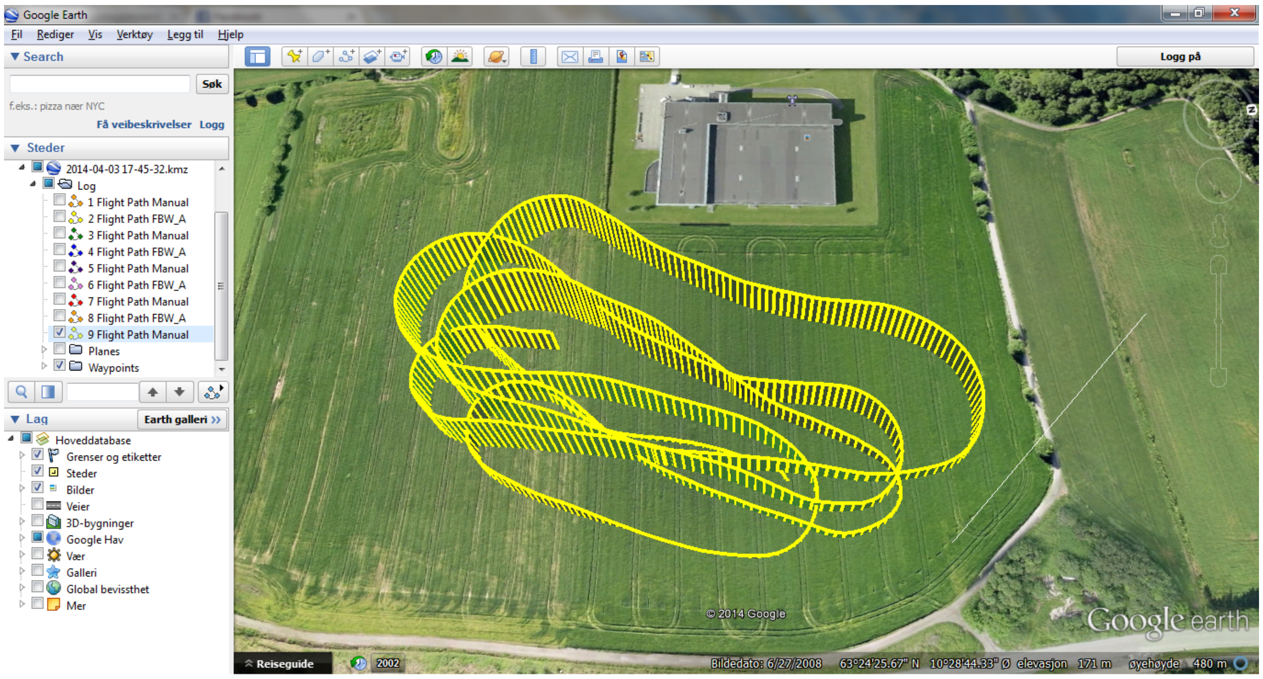


Figure 8.13: Flight Test 2, Flight Path Plotting

from before had disappeared. However, this time, the response from the aircraft was absent or at least too slow to be noticed by the pilot.

The autopilot should be tested and tuned some more in order to get the full autopilot capabilities working. Before the flight tuning is completed, the autonomous GPS waypoint tracking provided by the *APM* system is not possible to use since the autopilot needs to be able to control the aircraft during flight for this feature to be enabled. It should not be too difficult to get the PID controllers tuned, thus enabling autonomous flight as was not during this Thesis. The *APM* system is a well-tested, working autopilot system and should therefore not provide any challenges of getting it to work properly. Some more tuning and testing will make the system capable of performing autonomous flights and thus be feasible to use as part of a SAR UAS.

8.7 APM Test Summary and Discussion

The testing performed of the *APM 2.6* autopilot proved that once the system is setup up correctly and tuned, it will be a possible autopilot to use in a SAR UAS. The autopilot performed well and was easy to setup on the plane. As mentioned in Section 7.3.1, it was also tested to mount the autopilot on a quadcopter and thus proving its capabilities of controlling multiple airframes. The autopilots features and capabilities will be good to use during SAR operations and inexperienced pilots might also use the system once autonomous, or assisted flight has been set up and made possible to utilize.

The testing of the *APM* system went fairly well and no significant challenges, which would make the system incapable of being used during SAR operations, were noticed. The experience made during the flights was that take-off should not be performed manually since there are too many things that could go wrong. A launching ramp or some other approach should be used in order to get a successful take-off every time. The *APM* system does possess the capability of performing the take-off and landing autonomously, but in order for that to work the system needs to be tuned to make autonomous flight possible. A more experience pilot might be necessary to pilot the plane before the tuning is complete. Once the tuning is completed, the plane will be easier to fly, thus possibly enabling inexperienced pilots to use the UAS without any problems.

A different issue that was noticed during take-off and landing, was that this particular flying wing has the motor mount in a very exposed location which makes it prone to impacts. The plane has no wheels and thus the motor is very exposed during landings and failed take-offs. The motor might have to be mounted in a more protected area of the plane in order to increase the robustness of the system. Figure 8.14 illustrates a similar plane design as used in the prototype UAS of this Thesis. The illustration is from a system called *Trimble*, which is mentioned in Section 2.3.2, where the motor and propeller is somewhat protected by the wing frame itself [Trimble]. The system also has a propeller that fold backwards on impacts, which makes it less prone to breaking if impacts occur. A similar solution could be adopted for the UAS discussed in this Thesis and it might improve the durability of the UAS and prevent it from breaking during landing. Vegard Hammerseth mentioned in his Master Thesis that the motor mount of the UAV, which is the same as used in this Thesis, is made of plastic and thus not very durable [Hammerseth, 2013]. The mount should therefore be replaced with some other material at a later stage of the development.

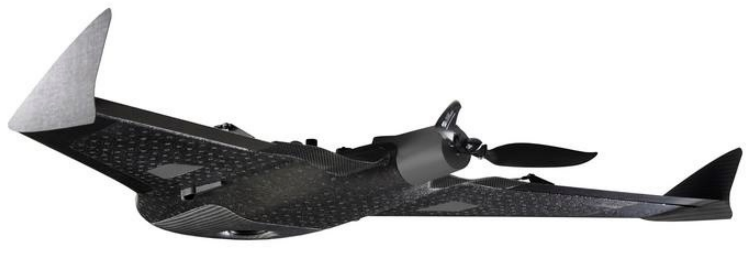


Figure 8.14: Trimble UAV [Trimble]

The GCS application of the *APM* worked very well, both during and after flight. It was easy to see which mode was selected on the aircraft during flight by viewing the flight information on the GCS application. The GCS also displayed a map showing the flight path of the aircraft, which was good to use in order to know which areas were actually flown in. The flight logs were easily accessed post-flight and could then be used for analysis of different flight aspects. The flight path extracted from the log, as illustrated in Figure 8.13, can be used by the SAR personnel for determining which areas that were actually covered by the UAV during its flight and they can use this information for further planning the mission. The UAV could perhaps be ordered to cover some area one more time, or cover an area that was not covered properly the first time.

Overall, the *APM* autopilot system proved very feasible for use as part of a SAR UAS. Many of the needed features are already implemented with the system and using them was not challenging.

In the vision of a SAR UAS as illustrated in Figure 5.2 in Chapter 5, multiple UAS are used and this will probably be a highly desired capability of a SAR UAS. The GCS application used in this Thesis does not currently possess the capability of allowing multiple UAV control. However, there is a newer version being developed that will allow this, thus making it possible to control multiple UAV using the *APM* system.

To show the capabilities of the *APM* GCS application analysis tools, the following example is included in this Thesis.

8.7.1 Take-Off Crash Analysis

The reason for reviewing the flight logs post-flight might be many. The most obvious reason is in order to tune the parameters of the PID controllers enabling autonomous flight, or use the map feature for determining were the UAV has actually flown during the mission. When tuning the PID controllers of the *APM*, the autopilots navigation set points can be plotted by the GCS along with the actual response of the aircraft in order to look for undesired behavior such as slow response, overshoot, or porpoise response as illustrated earlier. During this Thesis, the reason for review the flight logs was to determine why the plane had not taken off successfully during the first flight of Flight Test 2, causing the propeller and mount to be damaged.

Scenario

During the failed take-off of the first flight in Flight Test 2, the plane hit the ground immediately after being tossed into the air without gaining any altitude. The mount holding the motor to the plane, and the propeller itself, broke, and the reason for this happening was not known. Neither the pilot nor the "tosser" noticed anything wrong with what they were doing and the crash could therefore not be explained. The plane was glued back together and a new propeller was acquired, and as mentioned, the plane flew successfully during the next flights of Flight Test 2.

A reason for the error that occurred during the first flight might have been that the motor mount had been damaged during the landing of the last flight in Flight Test 1. When trying to take off, vibrations might have caused the mount to loosen and as a result, the plane did not gain enough speed during take-off and thus hit the ground immediately. Another possibility for the error was discovered when reviewing the log files of the non-successful flight and comparing this to a successful flight. The review showed some discrepancies between the two flights and this might be the reason for the crash.

Flight log analysis

Figure 8.15 is a plot from the flight log of the failed, first flight of Flight Test 2. The plot includes the planes acceleration along the ground, along with throttle, pitch, and roll input given by the pilot. Figure 8.16 plots the same data from the flight log of the successful, third flight of Flight Test 2. The x-axis of the plots is the time axis, the left axis is the unit for acceleration values, and the axis to the right is the axis for the control input values.

The acceleration of the plane is the green line in both plots and the throttle input is the yellow line. The light blue and red lines are elevation input from the pilot. When the red line goes up and the blue goes down, it means that the pilot want the flaps of the plane to tilt upwards, hence causing the plane to fly upwards. By reviewing both plots, two things were particularly noticeable. The plane was in both cases launched by hand and the point at which the plane was tossed can be found at the point where the green acceleration line has a significant spike upwards. This happens in between 38:22 and 38:27 in Figure 8.15, and between 47:26 and 47:41 in Figure 8.16. It is clear that when the plane was tossed during the failed take-off, the throttle input was only slightly elevated and not at its maximum. This can be seen by the yellow line (throttle input) not being elevated before the green spike (acceleration) occurs. When reviewing the plot from the successful take-off, it can be seen that at the point when the plane was tossed, the throttle input is almost at its maximum. Throttle input was in the successful case given before the green spike occurs, i.e. before the plane was tossed. Because the throttle input was given later in the case of the failed take-off, this might have been the reason causing the plane not to gain enough speed and thus crashing into the ground. The second thing that was noticed was that during the failed take-off, the elevation input occurred sometime after the plane had been tossed. This can be seen by noticing that the red and blue lines separate sometime after the largest green spike starts to elevate in Figure 8.15. During the successful take-off, the elevation input was applied directly, or simultaneously as the plane was tossed into the air, as seen in Figure 8.16. This might have cause the plane to point its nose downwards during the failed take-off and thus not gaining any altitude before hitting the ground.

This analysis, if correct, showed that the reason for the failed take-off was a human error, and thus amplifies the assumption that an automated take-off routine is needed during SAR operations. It cannot be expect that a member of the SAR crew will be able to perform a hand launch of the aircraft successfully every time the UAS is used. As mentioned in Section 8.5.1, even with trained personnel like the Portuguese navy, errors do happen.

Flight log analysis Discussion

This small analysis example show the benefits of utilizing a system like the *APM*. All the tools that were used comes pre-implemented with the system and the focus can thus be on correcting problems rather than making the tools themselves. The result of this analysis is probably not that useful as it only shows the pilot to be a bad one. However, it did show that the failure was probably not caused by a mechanical or system failure, but a human error, and thus the system can be continued to be used without any repairs or changes.

8.8 Final Prototype UAS Test

The final test of the prototype UAS was planned to be a test were the UAS was used during a real SAR operation. Perhaps coordinating this with the Norwegian Red Cross and thus being allowed to participate in a mission is an approach that can be used. The UAS could then have be evaluated based on its performance during the operation in order to determine what is good and what is bad regarding the system design, and thus valuable feedback could have been acquired. In order for this test to be possible, the entire UAS has to be developed and operational. A detection system should be present on board the UAV and autonomous flight needs to be enabled.

The status of the prototype UAS developed in this Thesis is that the autonomous flight capabilities of the aircraft are not usable, and the human detection system is not completed nor mounted on the UAV itself. Hence, the prototype UAS will currently not be able to be used properly during a SAR operations and the final test was therefore not performed.

8.9 Evaluation and Discussion

The setup and tests that have been performed in this Thesis can be summarized as following:

1. Interfacing the *BeagleBone Black* and *APM 2.6* in order to have a detection system that can perform a localization of objects

2. Testing the range of the *XBee-PRO S2B* modules in order to determine if they are feasible for use during SAR operations

3. Interfacing the *XBee* module and *BeagleBone Black* for having a data link that can be used by the detection system

4. Setting up the UAV with the *APM 2.6* and its components to evaluate if this could be part of a SAR UAS

5. Performed flight tests to test the autopilots capabilities and evaluate if the prototype UAS is feasible for use during SAR operations

6. Testing the *APM* system analysis capabilities

Most of the setup and tests during this Thesis went well and proved that a UAS consisting of the components used will be possible to develop for use during SAR operations. As mentioned, the status of the prototype UAS developed in this Thesis is that the UAS is not fully developed and thus cannot be used successfully during a SAR mission. The autopilot is not tuned properly, which means that the systems autonomous flight capabilities are not enabled. The detection system is not mounted on the aircraft and the UAS cannot for human detection until it is. The detection system is also setup simply as a proof-of-concept and would thus have to be altered in order for it to function properly as a human detection system during SAR operations. The data communication link is not interfaced with the detection application and will have to be in order for the detection system to notify the operators of any detections it performs.

When comparing the prototype UAS developed with the prototype UAS overview from Figure 7.1, which was the idea behind the prototype, all parts in the illustration are somewhat implemented and somewhat functional. The communication links between the GCS and UAV are present in the *APM* radio module, and the *XBee* modules tested. The UAV was equipped with a camera, and this camera could have been exchange for a completed detection system if desired. The GCS is already provided by the *APM* system and works as it is. So, the prototype UAS contains all components needed to be used during SAR, but these components have to be setup, and some further developed and tested properly before the system is feasible to use during a SAR operation. All components used during this Thesis are listed in Appendix C, and the total cost of the prototype UAS comes at $661, which is reasonable for a complete SAR UAS.

During this Thesis, many of the flight tests had to be postponed due to rough weather conditions. Rough weather is very likely in the case of a SAR operation and thus the UAS should manage to operate in relatively harsh weather conditions. The reasons a lot of the testing was postponed was that the testing personnel was inexperienced regarding RC planes, and thus did not want to break the equipment. The plane would probably have performed well in somewhat rough weather, but safety was put first and the UAS was not used when the weather was bad. A complete and fully developed UAS should remove the flight control from the user's hands and provide automatic take-off, flight, and landing, thus reducing the need for any experienced flight personnel and creating the possibility of using the system in somewhat rough weather. A test performed in windy and harsh weather conditions should be performed to evaluate if the UAS can be used in such conditions.

Depending on how the UAS is developed, the UAV could either be programmed to cover an area and report any findings to the SAR crew, or the UAV can follow a member of the SAR crew that is searching along a track and fly above their heads, as illustrated in Figure 5.3 and Figure 5.4. This Thesis has not focused on developing a system that can follow a searcher, but that would be possible if for example a Wi-Fi connection between a mobile phone and the *BeagleBone Black* is established and in addition making the *BeagleBone Black* capable of sending commands to the *APM 2.6*. The prototype UAS tested in this Thesis could be used during SAR operations to cover an area and report its findings when something is found if the detection system is further developed and mounted on the aircraft. If the autopilot in addition is tuned properly, the UAS could perform this detection autonomously without the need for a pilot controlling the system.

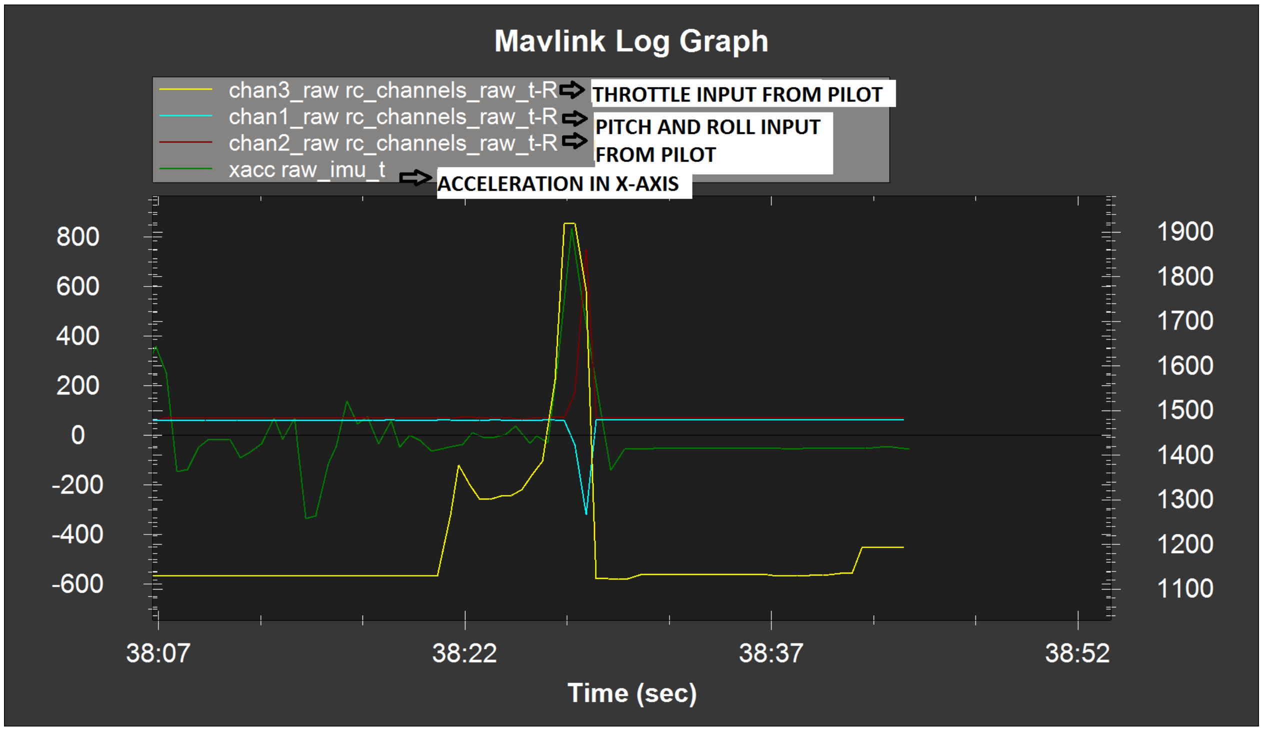


Figure 8.15: Log plot of Flight Test 2, Failed take-off

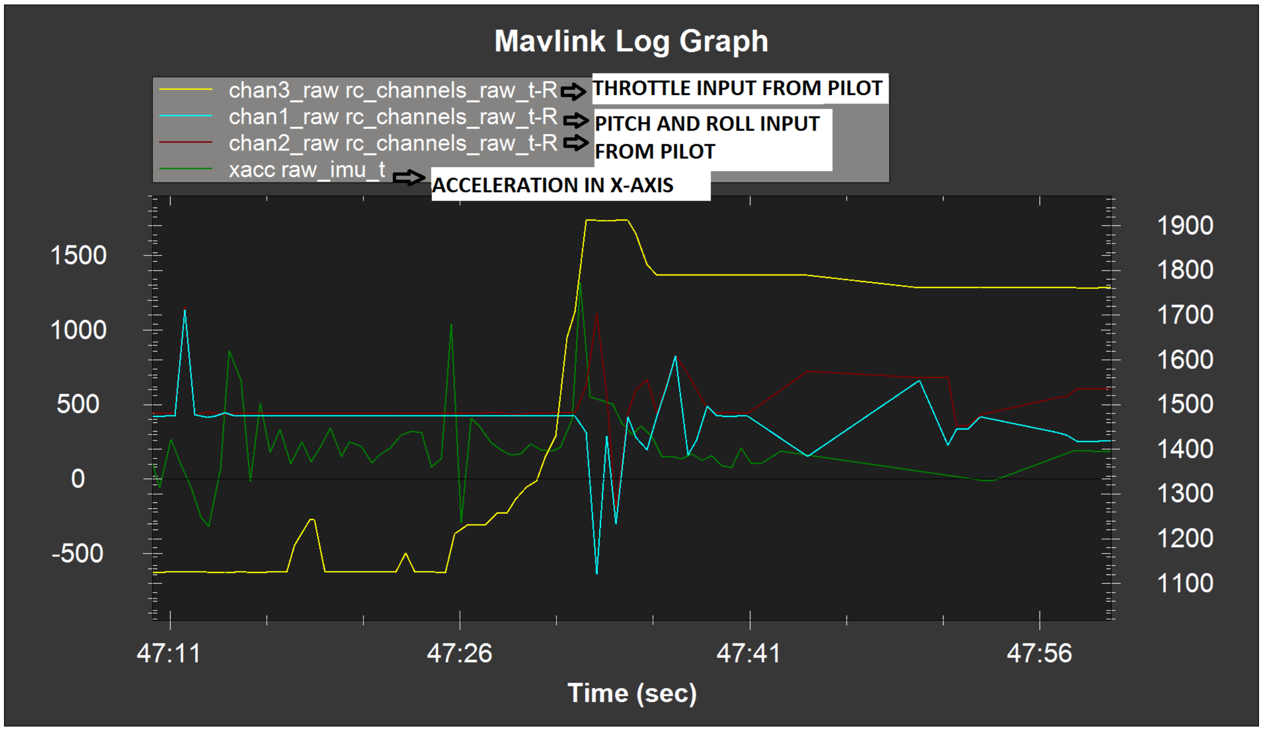


Figure 8.16: Log plot of Flight Test 2, Successful take-off

Chapter 9

Further Work

In order to develop a complete and functional SAR UAS, the prototype UAS created and tested in this Thesis can be used as a starting point for further development. This chapter will discuss some possible improvements and additions to the prototype UAS, and in addition explain what needs to be done to the prototype in order to create a working SAR UAS. The intention is that others may use this chapter as a starting point for their work in developing a complete and functional SAR UAS.

9.1 Prototype UAS Status

The prototype UAS created in this Thesis is, as mentioned in Section 8.9, not a working UAS that can be used during SAR operations. Some issues needs to be resolved for the UAS to be considered functional enough for being utilized by SAR personnel. This section will provide some information regarding the current state of the prototype UAS.

The UAS will be discussed as a whole and some parts will be mentioned in more detail than other parts. The prototype UAS is currently not set up and functional enough to be used during SAR operations and the following list mentions the parts of the UAS that needs attention before it can be considered used by SAR personnel.

- Autopilot
- Take-off and landing procedures
- Detection system

9.1.1 Flight System

Autopilot

The autonomous flight capabilities of the autopilot system used on the prototype are not enabled due to lack of tuning and the prototype has to be flown manually until they are enabled. The autopilot is not properly tuned and will need to be in order for its autonomous capabilities to work. The autopilot system used is proven to work through a massive community, and should thus provide no problems in being fixed and set up for the prototype.

The autopilot used for the prototype was during this Thesis borrowed from another project and unfortunately, it was needed back after this Thesis had been completed. Buying such a system or a similar one is a solution to this problem. The components of the prototype UAS are listed in Appendix C and they are marked as present or missing parts of the system.

Take-off and landing procedures

The take-off and landing procedures for the prototype UAS are to use hand launching and belly landing. The plane is flown manually and thus the pilot has all the control during take-offs and landings, as well as the flight. The autopilot used does possess the capability of providing the operator with autonomous take-off and landing, but that only works if the autopilot is tuned and able to fly autonomously. However, the procedures used for take-off and landing, i.e. hand launching and belly landing, will also be used by the autopilot system if those features are enabled. The autopilot will perform the take-off and landing autonomously, but the operator has to hand launch the aircraft, and it lands by using the belly-landing method. Implementing some other approach might be necessary for a SAR UAS. Perhaps having some sort of ramp for take-off, and net for landing will be advantageous. Some methods for take-off and landing are mentioned and discussed later in this chapter.

The take-off and landing procedures are not essential to implement before the system can be tested in a SAR scenario. It is fully possible to launch the plane by hand, and experienced RC pilots does this all the time. Hence, if an experienced test operator is used, hand launching and belly landing are methods that can be used during a test performed in a real SAR scenario.

9.1.2 Detection System

The detection system developed in Chapter 6 was not mounted on the UAV itself and thus not tested together with the UAS. The prototype UAS will not be able to be used for detection purposes before this is accomplished. The detection application is made simply to prove the capabilities of the devices used and the detection will probably not work in a real SAR situation. This is because the application detects humans based on detected facial features, which are not certain to be visible in a SAR scenario, or possible to detect from the altitude the UAV probably will fly at. A different algorithm would have to be implemented in order for the system to detect humans in a SAR scenario. Currently, only humans wearing red sweaters and that has their facial features visible to the camera will be detected. This is probably not very likely in all SAR scenarios.

The detection application is not developed to include the data communication link between the processing unit and the GCS. A data communication link application was created between the detection system's processing unit and the GCS, but this was not included in the detection application itself. The communication link was only set up to show that the communication could be achieved and will have to be implemented with the detection application in order for the system to be capable of notifying its operators regarding a detection.

A one-way communication application was created between the detection system's processing unit and the autopilot, but the communication capabilities was not included in the detection application, hence the detection application can not use the flight information. This will have to be implemented with the detection application in order for the localization part of the detection to work because the detection application needs the location and orientation of the aircraft to perform the localization of objects. Having a two-way communication link between the devices would also allow the detection system to control the flight if that was desired.

The localization application created in Section 6.6 is not fully developed. The application only proved the feasibility of such an application and will have to be studied to make it function correctly. As explained, the formulas used for calculating the GPS position of objects were not evaluated properly and would possibly need to be changed for some other, more correct formulas if the system is further used.

9.2 Further work on the prototype UAS

This section will list some recommendations for short-, and long-term further work that needs to be performed in order for the prototype UAS to be considered feasible for use during SAR operations. The short-term recommendations are work that needs to be done in order for the UAS to be operational and thus possible to use as a demonstration SAR UAS. The long-term recommendations are work needed to be performed in order for the UAS to be considered a complete, fully functional system, which can be used during SAR operations. Having performed all the long-term recommendations, the UAS could possibly be considered an efficient and important tool to be utilized during SAR operations.

9.3 Short-Term

As explained, the short-term recommendations for further work, are the tasks needed to be performed in order for the prototype UAS to be considered possible to utilize during SAR operations.

9.3.1 Autopilot tuning

The autopilot system currently used, the *APM 2.6*, needs to be properly tuned in order for its autonomous flight capabilities to be enabled. Once tuned, the autopilot will provide the user with mission planning and autonomous flight capabilities, thus making it possible for the plane to operate on its own without the need for human input during flight.

In order to tune the autopilot, an experience pilot might be needed, or at least several people understanding the system needs to be present during the tuning. One person has to fly, and the other needs to be on the computer altering the gains of the PID controller for pitch, roll, and yaw. Doing this alone is somewhat troublesome.

As explained, the autopilot was borrowed and has now been returned. Further work on this project would require a new autopilot to be acquired.

9.3.2 Autonomous take-off and landing

Whether the current autopilot or some other system is used, autonomous take-off and landing is probably essential to have during a SAR operation. The current autopilot used in the prototype UAS does possess these capabilities and once tuned, they will be enabled. However, the autopilot still take advantage of hand launching and belly landing. More clever approaches might be necessary during SAR operations and some will be mentioned here. The methods are considered for planes only, as multi-rotor vehicles possess VTOL capabilities.

Launching ramp

Creating a launching ramp is something that is possible in order to make the take-off of a UAV easier. The ramp can be used to provide the UAV with a surface in order to obtain enough speed for a take-off to be possible. A solution utilizing a ramp is a bungee-launcher, which is illustrated in Figure 9.1 [Bungee Launcher]. This solution will provide the UAV with enough speed before the ramp is left and make the UAV capable of taking off. If a ramp is constructed long enough such that the UAV gains enough speed by itself before leaving the ramp, the need for a bungee might not be necessary. These bungee launchers can be constructed quite small and with the possibility of assembling the ramp in the field. The construction material used can be relatively light, thus making the ramp ideal for transportation. This will make them very usable during a SAR operation for launching the plane.

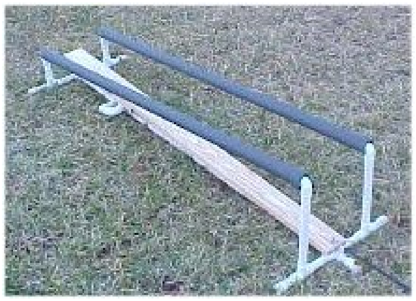


Figure 9.1: Bungee-Launcher [Bungee Launcher]

If a bungee-launcher is constructed and the autopilot system is tuned, the operator only needs to place the plane on the ramp, set the autopilot on automatic take-off, then release the bungee and the plane would take-off by itself. This approach will be feasible to use during SAR operations.

Car ramp

Mounting a ramp on top of a car and putting the plane on top of this ramp is a possible take-off approach to use during SAR operations. The car can be driven to gain speed and then the plane is released when enough speed is obtained to accomplish a take-off. The ramp could also be mounted on a snowmobile if needed and thus the same approach could be used in wintertime. This approach will require a somewhat straight road with no overhanging trees, which might not be the case in all SAR scenarios, and thus the approach is perhaps not a good method to utilize for a SAR UAS.

Computer Vision

Computer vision can be used for both UAV landing and take-off. If used for take-off, a camera can be directed towards the ground and the system could notice when the plane has been tossed into the air by detecting that the ground is moving. Once the system detects movement, the UAV can start its engines and applying upwards control output to its actuators, hence making the UAV fly up into the air. Several elevation approaches can be used during take-off, for instance circle elevation and straight elevation. In circle elevation, the plane circles around a point until a certain altitude is reached. This would be desired if the plane was launched in an area where there are hills or other objects in all directions, thus making a straight elevation approach infeasible. The straight elevation approach is exactly that, the plane takes off and flies in a straight line until a certain altitude is reached.

Using computer vision for landing is an approach where a camera can be used for detecting some pre-defined objects on the ground. By having for instance a grid defined by some red objects or lights, the plane could be guided by the camera into this grid and once there, know that it is fairly close to the ground and thus cut its engines and glide towards the ground. This would be a similar approach to landing as using radio tubes. Radio tubes can be used for landing by providing the plane with radio signals. Once the plane is aligned with these radio tubes, all signals are detected and thus the course of the UAV is known. Holding this course until the signals disappears, and then stopping the engine could be used as an approach for landing the plane. A problem using these approaches would be in the scenario where there is side-winds and the plane has to fly somewhat sideways towards the ground. The grid or radio tubes would then not be visible or detected.

Catching net

In order for the UAV to land, a net can be used to capture the UAV in flight. The UAV can be programmed to fly into the net and thus it will be caught and stopped without harming the UAV. The design of this net can be done in several ways and an example can be taken from a system found during the pre-study of this Thesis. A system called FULMAR utilizes a catching net for landing the aircraft [AEROVISION VEHICULOS AEREOS, S.L.]. The UAV is flown into the net and thus not damaged by any impacts with the ground. This net is however quite large, but so is the UAV they are utilizing and thus making a smaller net for the prototype UAS developed in this Thesis is possible.

Parachute

A different approach than the previous ones would be to use a parachute for landing the UAV. The UAV could be programmed to fly towards a location, and once this location is reached, the engine stops and the parachute is released. The parachute will thus make the plane descend with a speed of which will not cause harm to the UAV once the ground is hit. Landing the UAV this way will of course introduce issues regarding where the UAV will actually hit the ground. If the area where the UAV is used has many trees, the UAV might end up in the top of a tree, which is not desirable. Having a system that takes the wind and other parameters into account when deciding the position to release the parachute related to where it wants to land on the ground, could make it possible to land the UAV even in windy conditions.

9.3.3 Detection System

The further work that needs to be performed on the detection system is implementing an algorithm that can detect humans in a SAR scenario. The current application is only created to demonstrate that the system units are indeed capable of performing such a detection. The same algorithm approach can be used, but some changes has to be made. The current application recognizes faces instead of human bodies and this might not be feasible to use during SAR operations.

The current detection application utilizes, as mentioned in Section 6.4.3, cascade classifiers for Haar features for detecting faces in the image. By creating a different classifier for a human body viewed from the air, this could be used for detecting the humans during SAR operations from a UAV. This is possible and examples online have used the computer vision library *OpenCV* to make such files for detecting different objects such as for example bananas [Ball].

The current detection application does not include a data communication link back to a GCS, nor a communication link to the autopilot. The communication links will have to be included in the detection application for notification purposes and localization purposes. The localization application will also have to be combined with the detection application. The test application created in Section 8.3, which uses the *XBee* modules, would be possible to incorporate into the detection program and thus enabling the communication between the GCS and detection application. The testing done in Section 8.1 would also be possible to be incorporated with the detection application to make the flight data available and thus object localization possible.

9.4 Long-Term

The long-term recommendations for further work are the tasks needed to be performed in order to optimize and make the UAS better than simply a prototype system. By performing the recommendations listed here, the UAS should end up possibly being considered a very efficient and relevant tool for use during SAR operations.

9.4.1 Thermal Camera

The camera used for detection purposes in the current prototype UAS is a normal webcam. As explained in Chapter 6, a thermal camera might be a better option to utilize for human detection in a SAR scenario. Using a combined approach with both a thermal and a color camera is also possible. By including a thermal camera as part of the UAS, humans could be detected easier and more accurately than when using a color camera and would make it possible to perform the detection even in complete darkness.

The detection application created in this Thesis uses a normal webcam, and the application is written to detect red clothed humans with their faces turned towards the camera. If a thermal camera is acquired, heat signatures of a human could instead be detected, which was the idea behind the algorithm used. In the future, the cost of thermal cameras will be lower and thus it might be possible to utilize such a camera in a low cost UAS.

9.4.2 Wi-Fi Connection to User

The embedded computer used for the detection system in this Thesis could also be used in order to connect a user or searcher and the autopilot together. The searcher could then simply use his smart-phone, which most people have nowadays, and connect to the UAV using a Wi-Fi connection. Sending commands over this link and ordering the UAV to perform certain tasks would be a simple and easy approach for utilizing the UAV during SAR operations. The smart-phone of the SAR searcher could then be used to control the entire UAS. If the Wi-Fi range does not cover the entire search region, the UAV could fly autonomously around and when something is detected, return to a location and then transmit the information gathered.

9.4.3 Velocity Sensor

A velocity sensor should be included as part of the autopilot system. The current solution used, the *APM 2.6* autopilot, has its own velocity sensor and this sensor could simply be connected to the autopilot in order to be used. This sensor is possible to utilize during landings and take-offs in order to maintain the correct speed in these situations. The sensor was not connected during this Thesis, as its usage was not needed.

9.4.4 Path Planning

The mission planning of the prototype UAS was done using the *APM*s GCS application. This application allows for point-and-click configurations of GPS waypoints and a mission can thus be planned using this capability. The path taken by the UAV during these missions are chosen by the autopilot itself, but having a system that would make the UAV take the shortest path given some GPS waypoints could be a possible tool to use during SAR operations. The system, which could create these paths, could also be programmed to include wind information as to save energy during the flight. The path system could also provide the most optimal flight path for the UAV such that the detection system has optimal conditions, perhaps by always flying in a way that makes the camera point downwards.

9.4.5 UAV Communication Network

Developing a UAV communication network is a technology that could be very useful for SAR UAS. Having more than one UAV in a UAS is beneficial for several reasons, and having a communication system that automatically handles the network between these UAV could be essential for such a system to function. Including several UAV in the same SAR operation creates the opportunity of utilizing a mesh topology, thereby extending the range of the network and providing redundancy. Such a communication network could be constructed and used as part of a SAR UAS.

9.4.6 Mid-air Collision Avoidance System

Having a MID-air Collision Avoidance System (MIDCAS) might be needed as an essential part of a SAR UAS. The possibility of a UAV hitting something or somebody else is very small, but still, situations where this might occur is possible and having a system that handles this is therefore desirable. The need for MIDCAS could be higher if several UAV are to be in the air at once.

In addition, if the UAS is to be used in the same airspace as manned air-traffic, the need for MIDCAS might be vital for the UAS to be allowed to be used. This collision system might also have to be incorporated with the manned aircraft systems in order for them to work together.

Computer vision can be used to provide a UAS with MIDCAS capabilities. The system can be developed on a small, embedded computer and thus be used on board a UAV [Kalvå, 2014].

Radar is also possible to use for avoiding other objects such as UAV, as well as trees and mountains. However, rules of conduct, similar to what is used on car roads, would have to be standardized for this to work. If one UAS is programmed to turn left when an object is right in front of it, and the other UAS is programmed to turn right, they will most likely hit each other instead of turning away from each other.

9.4.7 Gimball camera mount

Having a camera on board a UAV for capturing images of the ground will be hard to accomplish if the camera is stationary and the plane is turning. If the weather conditions are windy, the UAV might have to fly a somewhat sideways to correct for the winds direction and will thus not point its underside directly towards the ground. A stationary camera would in this case perhaps not point towards the ground at all. A Gimball camera mount can be used for changing the view of a camera during flight. It can be used to correct for the angles of the UAV, thus pointing the camera towards a desired location even when the plane is turning, and it can be used for stabilizing a video feed if that is what the camera is being used for.

9.4.8 Image Mapping

If the SAR UAS is used to send back images to a GCS as to provide the operators with a visual tool, having single, non-connected images might not be the best solution. Having a system that creates a mosaic image, or image map, of the received images could be useful. By combining and stitching the images together, a more completed view of an area is possible to create. Of course, the separate images can be viewed if desired, but also having an image map will be a useful visualization tool for the SAR personnel. This map could be used as a "real-time" replacement map for the one which the SAR personnel probably has of the search area.

9.4.9 Fault detection

A SAR UAS needs to be able to detect faults that occurs with and to the system. If the system is to be considered safe, fault detection mechanisms needs to be implemented. The current autopilot does possess features for detecting the loss of a communication link and low battery voltage, and is able to perform safe actions and warn the user in the event that this occurs.

Having a UAS that will be used during a SAR operation, will necessarily mean that the UAS needs to be as safe as possible. A SAR operation include many people and the safety of these people are essential to maintain. Having a UAS that is not reliable, hence, jeopardizing the safety of these people is not an option.

9.4.10 Make another UAV

Of course, a long-term recommendation is to create a second UAV for testing and development purposes. A SAR UAS should provide the option of utilizing multiple UAV during an operation, thus possibly increasing the efficiency of the search. Having several UAV with different capabilities is also a possible scenario during SAR operations. Making a second UAV would allow such features to be developed and tested.

9.4.11 Collaborate with other projects at the institute

There is currently being done a lot of work regarding UAS and related technology at the Department of Engineering Cybernetics at NTNU. Some projects involves creating algorithms for dropping packages from the air at the exact location specified even with wind and other conditions, and some involves creating control systems for UAV flight control. Whatever is being done, much of the development is most likely relevant in some way for use as part of a SAR UAS as well. Therefore, collaborating with other projects at the institute is a good approach to get a SAR UAS as good as it possibly can be.

Chapter 10

Discussion and Conclusion

The feasibility of developing a low cost UAS for use during SAR operations has been shown throughout this report. A pre-study and literature study were conducted where the desire and will for using and developing such a system has been established. There are not that many UAS works or projects specifically directed towards a SAR scenario, hence studying this particular application for UAS is important to increase the possibility of UAS being utilized during SAR operations. This report has listed information as to how a SAR UAS can be developed, and discussed the issues regarding its design. A prototype UAS was further developed from the same system as used in the pre-study. Having a human recognition system on board a UAV is highly advantageous during SAR operations and thus developing this to be part of the UAS is important. A low power, low cost detection system was developed and proven functional through testing. A SAR UAS should be able to operate as autonomously as possible in order to keep the workload on the SAR personnel to a minimum. In addition, by having an autonomous UAS, the system could be possible to utilize even by unexperienced operators.

The result of this Master Thesis is a report that can be used for information purposes during the creation of a complete SAR UAS. Information regarding the design of a SAR UAS has been provided and a prototype UAS was developed. The prototype UAS is an almost working SAR UAS, which with some modifications, can be used during a SAR mission as a human detection tool. The reason for wanting UAS incorporated in SAR operations is that it could possibly increase the efficiency of the operation, hence increasing the probability of saving more lives than compared to not using the system. A SAR UAS can also be used in areas that are considered too dangerous for humans to search in, which is advantageous if such situations occur.

The implications of this Master Thesis on other parts of society are many, and if a SAR UAS is successfully developed, it can be used for many other applications besides SAR. The prototype UAS developed in this Thesis was developed to be as environmentally friendly as possible. This was accomplished by making it possible to use renewable energy sources for charging the batteries of the system; hence, the implications of utilizing this system on the environment will be minimal. As a comparison, manned helicopter uses a lot of fuel during their operations. Having a more environmentally friendly system that in some situations could be used in their place, would be advantageous. Other UAS application areas, like for instance police or military work, could benefit from the information presented in this report and help making a UAS usable for these applications as well. A UAS as discussed in this Thesis could also be used for sporting applications or in TV production.

10.1 Conclusion

The feasibility of developing a Search And Rescue (SAR) Unmanned Aerial System (UAS) has been stated in this report. A low cost, easy-to-use UAS can be developed and possibly contribute to increasing the efficiency of SAR operations. A solution for such a system using many pre-made components and devices was presented, and a prototype implemented during this Master Thesis. The necessary background theory and previous works were studied and used for determining the design of the prototype UAS. The developed UAS was tested and proven to work as expected given that the system was not fully developed. Some further work regarding SAR UAS was presented and also what needs to be done regarding the prototype UAS to make it functional was explained. A low power, low cost, computer vision detection system to use on board a UAV has been developed and tested, and the testing proved the feasibility of having an autonomous detection system on board a UAV during SAR operations.

The main objective of this Master Thesis was to investigate the feasibility of a concept of utilizing UAS during SAR operations and propose how such a system could be developed, and also implement a prototype of this concept. All these objectives are explained and implemented throughout this report, hence proving the feasibility of the concept.

Developing a SAR UAS in order to increase the efficiency of SAR operations, and be a useful tool for SAR personnel is highly achievable. A UAS can be designed and integrated into the SAR operations and be a factor that increases the possibilities of finding the missing persons during an operation. The UAS should be designed in a way that makes it possible to utilize the system for multiple purposes, like for instance human detection or communication assignments. The integration of a SAR UAS with the other components and participants of SAR operations has not been looked into during this Master Thesis. This will have to be studied in order for the system to be used as efficiently as possible. The developed prototype UAS can be used for determining how the system can, and should be integrated, and thus take part in a SAR operation.

The possibilities of UAS usage in society are many and not only restricted to the SAR scenario. Creating a SAR UAS will provide information on how to create UAS for other applications as well, and thus the development in this Thesis will not necessarily only contribute to the SAR community. Other UAS application developers may use the discoveries in this Thesis on their systems resulting in better UAS for other application areas as well.

The UAS developed in this Thesis is far from ready to be a commercialized product, but the system can be used as a possible solution and provide information to further research and development of SAR UAS. The prototype UAS developed in this Thesis is currently not recommended to be used during SAR operations, but it could be considered a fully usable and efficient SAR tool given some small alterations. The continuation of developing a SAR UAS is recommended and a well-working, functional system should be possible to create.

References

Bearing between GPS locations. http://www.movable-type.co.uk/scripts/latlong.html. Accessed: 2014-05-31.

Distance between GPS locations. http://williams.best.vwh.net/avform.htm. Accessed: 2014-05-31.

Forum Post. http://stackoverflow.com/questions/7477003/calculating-new-longtitude-latitude-from-old-n-meters. Accessed: 2014-05-31.

Global navigation satellite system. http://snl.no/GNSS. Accessed: 2013-09-13.

Maxi swift, flying wing. http://www.electricwingman.com/maxi-swift-white-epp-flying-wing.aspx. Accessed: 2013-10-16.

3DRobotics. Apm 2.6. http://store.3drobotics.com/products/apm-2-6-kit-1. Accessed: 2013-12-16.

3DRobotics. Radio Module. http://3drobotics.com/learn/. Accessed: 2014-05-19.

AEROVISION VEHICULOS AEREOS, S.L. FULMAR aerial teledetection system. www.aerovision-uav.com/. Accessed: 2013-11-06.

APM. APM Planner 2.0. http://planner2.ardupilot.com/. Accessed: 2014-04-24.

APM. ArduPilot for BeagleBone Black. http://dev.ardupilot.com/wiki/building-for-beaglebone-black-on-linux/. Accessed: 2014-04-28.

APM. Mission Planner. http://planner.ardupilot.com/. Accessed: 2014-04-24.

APM. Multiplatform autopilot. http://ardupilot.com/. Accessed: 2013-12-02.

Ball, T. Traing your own opencv haar classifier. http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html. Accessed: 2014-06-04.

BeagleBoard.org. BeagleBone Black. http://beagleboard.org/products/beaglebone%20black. Accessed: 2013-12-16.

Bungee Launcher. http://www.radiocontrolinfo.com/EDF/launch.php. Accessed: 2014-06-04.

Burns, A. and Wellings, A. (2009). *Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real-Time POSIX*. Addison-Wesley, 4th ed. edition. ISBN: 978-0-321-41745-9.

CAN-cia.org. CAN Protocol. `http://www.can-cia.org/index.php?id=systemdesign-can-protocol`. Accessed: 2014-04-24.

CLOSE-SEARCH. European project. `http://close-search-project.eu/index.php/aboutclosesearch`. Accessed: 2013-11-30.

Diebel, J. (2006). Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors.

Digi International Inc. RF Basics. `http://www.digi.com/technology/rf-articles/rf-basics`. Accessed: 2014-05-16.

Digi International Inc. XBee Modules. `http://www.digi.com`. Accessed: 2014-05-17.

DX7s. Spectrum radio. `http://www.spektrumrc.com/Products/Default.aspx?ProdId=SPM7800`. Accessed: 2013-09-12.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*.

Gade, R. and Moeslund, T. (2014). Thermal cameras and applications: A survey. *Machine Vision and Applications*, 25(1):245–262.

Gamnes, G. (2013). Autonomous Unmanned Aerial Vehicle in Search and Rescue - A Prestudy.

Google Earth. www.earth.google.com. Accessed: 2013-12-16.

Google.com. www.google.com. Accessed: 2013-12-19.

GoPro. Versatile cameras. `http://gopro.com/`. Accessed: 2014-05-22.

GPS World. Drone hack: Spoofing attack demonstration on a civili unmanned aerial vehicle. `http://gpsworld.com/drone-hack/`. Accessed: 2013-12-19.

Guldbrandsøy, K., Austad, A., Havstein, G., Pukki, A., Øvervoll, J.-M., Halgunset, M., Husum, P., Williams, G. S., Torkildsen, P. O., and Himle, A. (2009). Kompendium i søkemetoder. ISBN: 9788272500930.

Hammerseth, V. B. (2013). Autonomous unmanned aerial vehicle in search and recue. Master's thesis, NTNU.

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

HobbyKing. Brushless motor. `http://www.hobbyking.com/hobbyking/store/__19035__turnigy_park450_brushless_outrunner_1050kv.html`. Accessed: 2014-06-05.

HobbyKing. Metal gear servo. `http://www.hobbyking.com/hobbyking/store/ __6340__digital_metal_gear_servo_16g_3_9kg_13sec.html`. Accessed: 2014-06-05.

HobbyKing. Turnigy batteries. `http://www.hobbyking.com/hobbyking/store/__9184__turnigy_5000mah_3s_20c_lipo_pack.html`. Accessed: 2014-06-07.

Hovedredningssentralen (2014). Statistikk for hovedredningssentralen (samlet) 2013. `http://www.hovedredningssentralen.no/`. Accessed: 2014-05-09.

ICARUS. European commision's directorate-general for enterprise and industry, research project. `http://www.fp7-icarus.eu/`. Accessed: 2013-11-14.

IEEE. POSIX Standard, IEEE Std 1003.1, 2013. `http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/pthread.h.html`. Accessed: 2014-05-14.

Jain, R. and Templin, F. (2012). Requirements, challenges and analysis of alternatives for wireless datalinks for unmanned aircraft systems. *IEEE Journal on Selected Areas in Communications*, 30(5):852–860.

Jo, A., Jang, G.-J., Seo, Y., and Park, J.-S. (2013). Performance improvement of human detection using thermal imaging cameras based on mahalanobis distance and edge orientation histogram. *Lecture Notes in Electrical Engineering*, 253 LNEE:817–825.

Johnston Jr., M. (2006). Ground object geo-location using UAV video camera.

Kalvå, A. (2014). Collision detection system using computer vision on low power devices. Master's thesis, NTNU.

Lin, P. The Robot Car of Tomorrow May Just Be Programmed to Hit You. `http://www.wired.com/2014/05/the-robot-car-of-tomorrow-might-just-be-programmed-to-hit-you/`. Accessed: 2014-05-16.

Lockheed Martin Corporation. Kestrel Autopilot fixed wing. `http://www.lockheedmartin.com/us/products/procerus/kestrel.html`. Accessed: 2014-04-24.

Logitech. c270 HD Webcam. `http://www.logitech.com/no-no/product/hd-webcam-c270`. Accessed: 2014-03-06.

Luftfartstilsynet.no. Civil Aviation Authority - Norway. `http://www.luftfartstilsynet.no/`. Accessed: 2013-10-30.

MAVLink. http://qgroundcontrol.org/mavlink/start. Accessed: 2014-02-10.

Meador, B. (2008). A Survey of Computer Network Topology and Analysis Examples.

METRO. Portuguese navy's drone launch ends in embarrassing fail. `http://metro.co.uk/2014/04/19/portugal-drone-fail-video-unmanned-aircraft-drops-into-lisbon-harbour-4703360/`. Accessed: 2014-05-08.

Mohammed Ali Koteich, K. R. (2011). Overall technical uav solution. Master's thesis, NTNU.

Morvan, Y. (2009). *Acquisition, Compression and Rendering of Depth and Texture for Multi-View Video*. PhD thesis.

NASA. Aircraft rotations, body axes. `http://www.grc.nasa.gov/WWW/K-12/airplane/rotations.html`. Accessed: 2014-05-19.

Ogale, N. A. A survey of techniques for human detection.

OpenCV. Computer Vision Library. `http://opencv.org/`. Accessed: 2013-12-02.

OutbackChallenge. UAV Challenge - Outback Rescue. `www.uavoutbackchallenge.com.au/`. Accessed: 2014-04-23.

Pandaboard.org. `http://pandaboard.org/`. Accessed: 2013-12-19.

Parkinson, B. W. and Spilker, J. J. (1996). *Global positioning system : theory and applications*, volume vol. 163-164. American Institute of Aeronautics and Astronautics.

Post- og teletilsynet. Information. `http://www.npt.no/teknisk`. Accessed: 2014-05-17.

Post- og teletilsynet (2012). Forskrift om generelle tillatelser til bruk av frekvenser. `http://lovdata.no/dokument/SF/forskrift/2012-01-19-77`. Accessed: 2013-12-16.

Radiocrafts. Embedded wireless solutions. `http://radiocrafts.no/`. Accessed: 2013-12-20.

RECCO. The RECCO System. `http://www.recco.com/the-recco-system`. Accessed: 2014-05-19.

Royal Ministry of Justice and Police (2002). Department of civil emergency and rescue planning. the norwegian search and recue service. `http://www.redningsnett.no/Redningstjenesten/Informasjonshefter`. Accessed: 2013-10-07.

Rudol, P. and Doherty, P. (2008). Human body detection and geolocalization for uav search and rescue missions using color and thermal imagery. Big Sky, MT.

Scopus. www.scopus.com. Accessed: 2013-12-19.

S.W.A.R.M. Volunteer Search & Rescue Network. `http://sardrones.org/`. Accessed: 2014-05-16.

Szeliski, R. (2011). *Computer Vision: Algorithms and Applications*. Springer London. ISBN: 9781848829350.

Termios. Terminal I/O API. `http://linux.die.net/man/3/termios`. Accessed: 2014-05-21.

Thermal-Eye 3600AS. Thermal imaging camera. `https://www.pr-infrared.com/shop/thermal-eye-3600as-thermal-imaging-camera-core/`. Accessed: 2014-05-08.

Torkildsen, P. O. (2009). *Savnet og ettersøkt: en studie om savnede personer på land i Norge og de søk som blir iverksatt for å finne dem*, volume 2009:2. Politihøgskolen.

Trimble. Trimble UX5 Aerial Imaging Rover System. `http://www.norgeodesi.no/trimble/uav-uas-rpas/trimble-ux5-aerial-imaging-rover-system/c-25/c-83/p-187`. Accessed: 2014-05-16.

TTK4155, NTNU (2012). Industrielle og innebygde datasystemers konstruksjon. Lecture notes.

UAS Norway. Non-profit independent organization. `http://www.uasnorway.no/`. Accessed: 2013-11-22.

UVS-INFO. The international remotely piloted system information source. `http://uvs-info.com/`. Accessed: 2013-11-14.

VIKO. Information finding and academic writing guide. `http://www.ntnu.no/viko/`. Accessed: 2013-11-30.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. pages 511–518.

Wang, W., Joshi, R., Kulkarni, A., Leong, W., and Leong, B. (2013). Feasibility study of mobile phone wifi detection in aerial search and rescue operations. Singapore.

Yanko Design and Rolle, T. Airborne Avalanche Rescue System. `http://www.yankodesign.com/2011/11/10/life-saving-air-drones/`. Accessed: 2014-05-19.

ZigBee.org. ZigBee Alliance. `http://zigbee.org/`. Accessed: 2013-10-31.

Appendix A

BeagleBone Black: Setup and Code

The *BeagleBone Black* was used during this thesis for performing computer vision tasks, communication with the UAV control system, and testing XBee devices. This appendix will list the code that was used for these tasks and explain the setup of the *BeagleBone*.

The *BeagleBone Black* was setup as explained by the *Getting Started with BeagleBone & BeagleBone Black* guide [BeagleBoard.org].

The unit was also altered so that the memory was expanded to utilize the uSD card for saving images while the unit still booted from the internal memory.

A.1 Versions

A.1.1 Operating System

The operating system used on the *BeagleBone Black* during the testing of this Thesis was determined by running the following commands: "*lsb_release -a*", "*cat /proc/version*", and "*uname -a*". The output from the command calls can be seen in Listing A.1.

Listing A.1: Setup of BeagleBone Black

```
root@beaglebone:~/humanDetection# lsb_release -a
Distributor ID: Angstrom
Description:    Angstrom GNU/Linux v2012.12 (Core edition)
Release:        v2012.12
Codename:       Core edition

root@beaglebone:~/humanDetection# cat /proc/version
Linux version 3.8.13 (koen@rrMBP) (gcc version 4.7.3 20130205 (prerelease) (Linaro GCC
    4.7-2013.02-01) ) #1 SMP Wed Sep 4 09:09:32 CEST 2013

root@beaglebone:~/humanDetection# uname -a
Linux beaglebone 3.8.13 #1 SMP Wed Sep 4 09:09:32 CEST 2013 armv7l GNU/Linux
```

As seen, the operating system Angstrom GNU/Linux v2012.12 (Core edition) is used. This was a somewhat unfamiliar operating system for the developer in this Master Thesis, but since most of the programming did not relate to any specific operating system, it was not exchanged.

A.1.2 OpenCV

The computer vision library *OpenCV* was also used during this thesis for performing the computer vision algorithms. The library was pre-installed with the Angstrom operating system used on the *BeagleBone Black*.

The version of *OpenCV* was found by printing the variables
CV_MAJOR_VERSION and
CV_MINOR_VERSION
from a computer vision application. The version used was found to be version 2.4.

A.2 CPU Clock Frequency

The *BeagleBone Black* has the opportunity of altering its CPU clock frequency and altering the frequency policy that is used for determining the frequency. The policy can be changed between the modes *conservative, ondemand, userspace, powersave,* and *performance* by using the command "*cpufreq-set -g MODE*", for example "*cpufreq-set -g performance*". The mode *userspace* was used during the testing with a frequency of 300, and 1000 MHz. The unit is capable of running at 300, 600, 800, or 1000 MHz, and the frequency can be altered by using the command "*cpufreq-set -f FREQUENCY*", for example "*cpufreq-set -f 1000MHz*".

To get information regarding the CPU, the command "*cpufreq-info*" can be run. The output from this command is listed in Listing A.2, and shows that the unit is currently running at 300 MHz with a *userspace* policy.

Listing A.2: BeagleBone Black CPU information

```
root@beaglebone:~/humanDetection# cpufreq-info
cpufrequtils 008: cpufreq-info (C) Dominik Brodowski 2004-2009
Report errors and bugs to cpufreq@vger.kernel.org, please.
analyzing CPU 0:
  driver: generic_cpu0
  CPUs which run at the same hardware frequency: 0
  CPUs which need to have their frequency coordinated by software: 0
  maximum transition latency: 300 us.
  hardware limits: 300 MHz - 1000 MHz
  available frequency steps: 300 MHz, 600 MHz, 800 MHz, 1000 MHz
  available cpufreq governors: conservative, ondemand, userspace, powersave, performance
  current policy: frequency should be within 300 MHz and 1000 MHz.
                  The governor "userspace" may decide which speed to use
                  within this range.
  current CPU frequency is 300 MHz (asserted by call to hardware).
  cpufreq stats: 300 MHz:nan%, 600 MHz:nan%, 800 MHz:nan%, 1000 MHz:nan%
```

A.3 Single image face detection

The code in Listing A.3 is used for doing face detection on a single image. The face detection function provided by the *OpenCV* library is either run directly on the image, or run after utilizing the blob detection approach as explained in Chapter 6 and extracting a region before the detection is performed. The application assumes that *OpenCV* is installed and needs the Haar cascade classifier file "*haarcascade_frontalface_alt.xml*", provided by the library, to be located in the same folder as the executable. A timer function, listed in Listing A.6, is also needed for determining the run-time of the program.

Listing A.3: Face Detection in image

```

/*
 * Find face in still image with or without Blob detection
 *
 * main.cpp
 *
 * main.cpp
 * dependencies: funcTimer.h and .cpp for run-time calculations
 *  ; and the cascade files for the OpenCV function needs to be in the same folder as the
     run file
 *
 *
 *  Created on: 15. mai. 2014
 *      Author: Gaute Gamnes
 */



/*
 * Some code is adapted from http://stackoverflow.com/questions/8166024/how-does-
    findcontours-cycle-through-the-image-opencv-2-3
 * Other code is adapted from Vegard Hammerseth, 2013.
 */

#include <stdio.h>
#include <iostream>
#include <fstream>
#include <string>
#include <time.h>
#include <sstream>

extern "C" {
  #include <pthread.h>
}

#include <opencv2/core/core.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <opencv2/nonfree/nonfree.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/objdetect/objdetect.hpp>
```

```cpp
#include <opencv2/contrib/contrib.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/stitching/stitcher.hpp>
using namespace std;
using namespace cv;

#include "funcTimer.h"

//#define GUI or BBB mode
//#define SAVE_TO_FILE
#define GRAPHICS
#define DEBUG

#ifdef SAVE_TO_FILE
#define STF(x) x
#else
#define STF(x)
#endif

#ifdef GRAPHICS
#define GUI(x) x
#else
#define GUI(x)
#endif

#ifdef DEBUG
#define DB(x) x
#else
#define DB(x)
#endif


// Extracts the red channel of the image and removes some of the red from other colors
void extractRed(const cv::Mat& src, cv::Mat& dst) {
   //Get red channel from src
   cv::Mat colorChannel(src.rows, src.cols, CV_8UC1);
   int fromTo[] = {2,0};
   cv::mixChannels(&src, 1, &colorChannel, 1, fromTo, 1);

   //Get grayscale of src
   cv::Mat gray(src.rows, src.cols, CV_8UC1);
   cv::cvtColor(src, gray, CV_BGR2GRAY);

   //Remove other colors than "real" red (white is also removed for instance)
   cv::Mat pureColor(src.rows, src.cols, CV_8UC1);
   pureColor = colorChannel - gray;

   dst = pureColor.clone();
}


// Finds the largest Contour in a vector of contours and saves the index in maxIndex.
// Returns the area of the largest contour.
double findLargestContour(const vector< vector<cv::Point> > &contours, int *maxIndex) {
   double area;
   double maxArea = 0;

   for(size_t i=0; i<contours.size(); i++) {
       area = contourArea(contours[i]);
       //cout << "Area of contour: " << area << endl;
       if(area > maxArea) {
           maxArea = area;
           *maxIndex = i;
       }
   }

   return maxArea;
}


// Find a face in an image. Can also be used for finding other features by choosing
// which cascade file to load. The result is shown with a green circle.
int findFaceInImage(Mat &img, Mat &result) {
  vector<Rect> found, found_filtered;

  CascadeClassifier face_cascade;
  string cascadeFileName1 = "hogcascade_pedestrians.xml";
  string cascadeFileName2 = "haarcascade_frontalface_alt.xml";
  string cascadeFileName3 = "haarcascade_fullbody.xml";
  string cascadeFileName4 = "haarcascade_lowerbody.xml";
  string cascadeFileName5 = "haarcascade_mcs_upperbody.xml";
  string cascadeFileName6 = "haarcascade_upperbody.xml";

  if( !face_cascade.load(cascadeFileName2) ) {
   cout << "Failed loading cascade file" << endl;
   exit(0);
  }

  //Parameters for cascade detection
  double scaleFactor = 1.1; //How much the image size is reduced each image scale
  int minNeighbors = 2; //How many neighbor each candidate rectangle should have to
   retain it
  int flags = 0|CV_HAAR_SCALE_IMAGE; //Not used in new cascade
  Size minSize = Size(8,8); //Minimum possible object size
  Size maxSize = Size(32,32); //Maximum possible object size

  Size s = img.size();
  cout << "Detecting faces... on img of size: " << s.height << " " << s.width << endl;
  face_cascade.detectMultiScale(img, found, scaleFactor, minNeighbors, flags, minSize);//
   , maxSize);
  cout << "Number of objects found: " << found.size() << endl;

  for(size_t j=0; j<found.size(); j++) {
   Point center( found[j].x + found[j].width*0.5, found[j].y + found[j].height*0.5 );
   int radius = cvRound( (found[j].width + found[j].height)*0.25 );
   circle(result, center, radius, Scalar(0,255,0), 4, 8, 0);
  }

  return found.size();
}

// Threshold values for The binary segmentation. Might need to be altered to fit the red
    value that is desired
#define THRESHOLD_RED 50
#define MAX_BINARY_VALUE 255

// The algorithm which find a face in the still image. It can choose to perform the face
    detection on the entire image,
// or, do the face detection on the segmented image. Chosen by Argument 1. Argument 2 is
    the path to the image
int main(int argc, char** argv) {

  if(argc < 3) {
    cout << "Arguments needs to be input to this program. " << endl;
    cout << "Arg 1: mode selector between blob (1) or no blob (0) detection" << endl;
    cout << "Arg 2: path to input image" << endl;
    exit(0);
  }
  int modeSelector = atoi(argv[1]);

  Mat frame;
  frame = imread(argv[2]);

  funcTimer duration;
  // Process image: SELECT BETWEEN BLOB OR NOT BLOB DETECTION
  cout << "Processing started" << endl;
  if(!modeSelector) {
    Mat faceResult;
    faceResult = frame.clone();

    DB( duration.timerStart(); ) // Timer start
    if(findFaceInImage(frame, faceResult)) {
      DB( duration.timerEnd(); )   // Timer end
      DB( cout << duration.getTimeInMS() << endl; )

      DB( cout << "Face found" << endl; )
      imwrite("images/faceNoBlobResult.jpg", faceResult);
    }
  } else if (modeSelector) {
    DB( duration.timerStart(); ) // Timer start

    //Extract red blobs
    DB( cout << "### Extract red blobs" << endl;)
    Mat redMat;
    extractRed(frame, redMat);

    //Threshold into binary
    DB( cout << "### Threshold into binary" << endl; )
    cv::threshold(redMat, redMat, THRESHOLD_RED, MAX_BINARY_VALUE, THRESH_BINARY);

    //Save image
    imwrite("images/BLOB_thresRed.jpg", redMat);

    //Extract contours
    DB(cout << "### Extract contours" << endl;)
    vector< vector<cv::Point> > contours;
    vector<Vec4i> hierarchy;
    findContours(redMat, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point
    (0,0) );
    DB( cout << "Contours found: " << contours.size() << endl; )

    //Find largest contour
    DB(cout << "### Find largest contour" << endl;)
    int largestContourArea = -1;
    int largestContourIndex = -1;
    if(contours.size() != 0) {
      largestContourArea = findLargestContour(contours, &largestContourIndex);
      DB( cout << "Largest contour index: " << largestContourIndex << endl; )
    }

    // Process contour area:
    DB(cout << "### Process contour area" << endl;)
    if(largestContourIndex != -1) {
      DB( cout << "Object found. Size: " << largestContourArea << endl; )
      // Extract contour area from frame
      cv::Mat imgROI;
      cv::Rect rect = cv::boundingRect(contours[largestContourIndex]);

      //Since the object used was only wearing a red sweater,
      //the ROI needs to be expanded somewhat larger in order to do the detection of a
    face
      cv::Rect newRect;
      newRect = rect + cv::Size(rect.area()*0.05,rect.area()*0.05);
      Size imgSize = frame.size();
      // Make sure the ROI is contained to the original image
      if(newRect.height > imgSize.height) newRect.height = imgSize.height;
      if(newRect.width > imgSize.width) newRect.width = imgSize.width;

      // Set location of rect
      newRect.x -= newRect.width/2;
      newRect.y -= newRect.height/2;

      // Make sure the ROI stays within the boundaries of the original image
      if(newRect.x < 0) newRect.x = 0; // Test left side
      if(newRect.y < 0) newRect.y = 0; // Test top
      if( (newRect.x + newRect.width) > imgSize.width ) newRect.width -= ( (newRect.x +
    newRect.width) - imgSize.width);
      if( (newRect.y + newRect.height) > imgSize.height ) newRect.height -= ( (newRect.y
    + newRect.height) - imgSize.height);

      imgROI = frame(newRect);
      Mat result = imgROI.clone();

      imwrite("images/BLOB_ROI.jpg", imgROI);

      // Do the actual processing
      if(largestContourArea > 100) {
        DB( cout << "Object found and above threshold" << endl; )
        if(findFaceInImage(imgROI, result)) {
          DB( cout << "Face found" << endl; )
          imwrite("images/BLOB_faceResult.jpg", result);
        }
      }
      DB( duration.timerEnd(); )    // Timer end
      DB( cout << duration.getTimeInMS() << endl; )
    }
  }

  return 0;
}
```

A.4 Webcam face detection

The code in Listing A.4 is used for doing face detection on a webcam stream. The face detection function provided by the *OpenCV* library, as in the single image face detection application, is either run directly on the images, or run by utilizing the blob detection approach and extracting a region before the detection is performed. The application assumes *OpenCV* is installed, and needs the "*haarcascade_frontalface_alt.xml*" file provided by the *OpenCV* library to be located in the same folder as the executable. A timer function, listed in Listing A.6, is also needed for determining the run-time of the program. The *BeagleBone Black* needs to be setup for utilizing the uSD card for storing the images because the application saves the results to the external uSD card once the detection on an image is completed.

Listing A.4: Face Detection in webcam stream

```
/*
 * Find face in webcam frame with or without Blob detection
 *
 * main.cpp
 * dependencies: funcTimer.h and .cpp for run-time calculations
 *   ; and the cascade files for the OpenCV function needs to be in the same folder as the
     run file
 *
 *  Created on: 15mai2014
 *      Author: Gaute Gamnes
 */


/*
 * Some code is adapted from http://stackoverflow.com/questions/8166024/how-does-
   findcontours-cycle-through-the-image-opencv-2-3
 * Other code is adapted from Vegard Hammerseth, 2013.
 */
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <string>
```

```
#include <time.h>
#include <sstream>
#include <unistd.h>

extern "C" {
  #include <pthread.h>
}

#include <opencv2/core/core.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <opencv2/nonfree/nonfree.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/objdetect/objdetect.hpp>
#include <opencv2/contrib/contrib.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/stitching/stitcher.hpp>
using namespace std;
using namespace cv;

#include "funcTimer.h"

//#define GUI or BBB mode
#define GRAPHICS
#define DEBUG


#ifdef DEBUG
#define DB(x) x
#else
#define DB(x)
#endif

// Extracts the red channel of the image and removes some of the red from other colors
void extractRed(const cv::Mat& src, cv::Mat& dst) {
   //Get red channel from src
   cv::Mat colorChannel(src.rows, src.cols, CV_8UC1);
   int fromTo[] = {2,0};
   cv::mixChannels(&src, 1, &colorChannel, 1, fromTo, 1);

   //Get grayscale of src
   cv::Mat gray(src.rows, src.cols, CV_8UC1);
   cv::cvtColor(src, gray, CV_BGR2GRAY);

   //Remove other colors than "real" red (white is also removed for instance)
   cv::Mat pureColor(src.rows, src.cols, CV_8UC1);
   pureColor = colorChannel - gray;

   dst = pureColor.clone();
}

// Finds the largest Contour in a vector of contours and saves the index in maxIndex.
// Returns the area of the largest contour.
double findLargestContour(const vector< vector<cv::Point> > &contours, int *maxIndex) {
   double area;
   double maxArea = 0;

   for(size_t i=0; i<contours.size(); i++) {
       area = contourArea(contours[i]);
       //cout << "Area of contour: " << area << endl; //For debugging
       if(area > maxArea) {
           maxArea = area;
           *maxIndex = i;
       }
   }

   return maxArea;
}


// Find a face in an image. Can also be used for finding other features by choosing
// which cascade file to load. The result is shown with a green circle.
int findFaceInImage(Mat &img, Mat &result) {
  vector<Rect> found, found_filtered;

  CascadeClassifier face_cascade;
  string cascadeFileName1 = "hogcascade_pedestrians.xml";
  string cascadeFileName2 = "haarcascade_frontalface_alt.xml";
  string cascadeFileName3 = "haarcascade_fullbody.xml";
  string cascadeFileName4 = "haarcascade_lowerbody.xml";
  string cascadeFileName5 = "haarcascade_mcs_upperbody.xml";
  string cascadeFileName6 = "haarcascade_upperbody.xml";

  if( !face_cascade.load(cascadeFileName2) ) {
   cout << "Failed loading cascade file" << endl;
   exit(0);
  }

  //Parameters for cascade detection
  double scaleFactor = 1.1; //How much the image size is reduced each image scale
  int minNeighbors = 2; //How many neighbor each candidate rectangle should have to
   retain it
  int flags = 0|CV_HAAR_SCALE_IMAGE; //Not used in new cascade
  Size minSize = Size(8,8); //Minimum possible object size
  Size maxSize = Size(32,32); //Maximum possible object size

  Size s = img.size();
  cout << "Detecting faces... on img of size: " << s.height << " " << s.width << endl;
  face_cascade.detectMultiScale(img, found, scaleFactor, minNeighbors, flags, minSize);//
   , maxSize);
  cout << "Number of objects found: " << found.size() << endl;

  for(size_t j=0; j<found.size(); j++) {
   Point center( found[j].x + found[j].width*0.5, found[j].y + found[j].height*0.5 );
   cout << "Object found at; x: " << found[j].x + found[j].width*0.5 << " and y: " <<
   found[j].y + found[j].height*0.5 << endl;
   int radius = cvRound( (found[j].width + found[j].height)*0.25 );
   circle(result, center, radius, Scalar(0,255,0), 4, 8, 0);
  }

  return found.size();
}

// Data used for communication between the threads
struct thread_data_t {
  Mat frame;
  int test;
  int algorithmModeSelector;
}GLOB_thread_data_t;

pthread_mutex_t frameLock;
pthread_barrier_t getStartedBarrier;

// Retrieval thread. Gets the frames from the webcamera
void * getWebcamFrames(void * argument) {
  //thread_data_t *data = (reinterpret_cast<thread_data_t*>(argument));
  VideoCapture capture(0); // 0 for standard camera, 1 for other. If laptop has camera,
    might need to use 1.
  if(!capture.isOpened()) {
    cout << "Cannot open the video capture" << endl;
    exit(0);
  }

  capture.set(CV_CAP_PROP_FRAME_HEIGHT, 720); //set camera rate
  capture.set(CV_CAP_PROP_FRAME_WIDTH, 1280); //set camera rate
  capture >> GLOB_thread_data_t.frame;
  pthread_barrier_wait(&getStartedBarrier);

  while(1) {
    pthread_mutex_lock(&frameLock);
    capture >> GLOB_thread_data_t.frame;
    pthread_mutex_unlock(&frameLock);
    usleep(100);
    //pthread_yield(); //give up CPU
    //WITHOUT SLEEP OTHER THREAD HAS TO WAIT FOR A LONG TIME BEFORE GAINING ACCESS TO
    IMAGE!
  }
  pthread_exit(NULL);
  return 0;
}

// Threshold values for The binary segmentation. Might need to be altered to fit the red
    value that is desired
#define THRESHOLD_RED 50
#define MAX_BINARY_VALUE 255

// The algorithm which find a face in the webcamera frame. It can choose to perform the
    face detection on the entire image,
// or, do the face detection on the segmented image. Chosen by Argument.
void * objectAlgorithm(void * argument) {
  //thread_data_t *data = (reinterpret_cast<thread_data_t*>(argument));
  pthread_barrier_wait(&getStartedBarrier);
```

```
DB( funcTimer duration; )

DB( int saveImageCounter = 0; )
DB( string fileName;
  ofstream myFile; )


 int modeSelector_FindRedBlobs = GLOB_thread_data_t.algorithmModeSelector;
 //USE this for selection if blob detection is to be performed or not. 1 => use blob, 0
   => no blob.

 if(!modeSelector_FindRedBlobs) myFile.open("/media/mmc0/noBlob/timeResults.txt");
 else myFile.open("/media/mmc0/blob/timeResults.txt");


 while(1) {
   //Get frame from webcam
   DB( duration.timerStart(); ) // Timer start
   pthread_mutex_lock(&frameLock);
   Mat frame = GLOB_thread_data_t.frame.clone();
   pthread_mutex_unlock(&frameLock);
   DB( duration.timerEnd(); )   // Timer end
   DB( cout << "!!!Get frame wait: " << duration.getTimeInMS() << endl; )

   // Keep track of the images:
   DB( stringstream ss;
     ss << saveImageCounter;
     saveImageCounter++; )

   // Process image: SELECT BETWEEN BLOB OR NOT BLOB DETECTION
   if(!modeSelector_FindRedBlobs) {

     Mat faceResult;
     faceResult = frame.clone();

     DB( duration.timerStart(); ) // Timer start
     if(findFaceInImage(frame, faceResult)) {
       DB( duration.timerEnd(); )   // Timer end
       DB( cout << duration.getTimeInMS() << endl; )

       DB( cout << "Face found" << endl; )

       DB(   fileName = "/media/mmc0/noBlob/result_" + ss.str() + ".jpg";
           imwrite(fileName, faceResult);
           myFile << "Time used on file: " << fileName << " \t was: " << duration.
   getTimeInMS() << endl; )
     }
   } else if (modeSelector_FindRedBlobs) {
     DB( duration.timerStart(); ) // Timer start

     //Save Frame
     DB( fileName = "/media/mmc0/blob/Frame_" + ss.str() + ".jpg";
       imwrite(fileName, frame); )

```

```
    //Extract red blobs
    DB( cout << "### Extract red blobs" << endl;)
    Mat redMat;
    extractRed(frame, redMat);

    //Threshold into binary
    DB( cout << "### Threshold into binary" << endl; )
    cv::threshold(redMat, redMat, THRESHOLD_RED, MAX_BINARY_VALUE, THRESH_BINARY);

    //Save binary image
    DB(   fileName = "/media/mmc0/blob/Binary_" + ss.str() + ".jpg";
        imwrite(fileName, redMat); )

    //Extract contours
    DB(cout << "### Extract contours" << endl;)
    vector< vector<cv::Point> > contours;
    vector<Vec4i> hierarchy;
    findContours(redMat, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE,
  Point(0,0) );
    DB( cout << "Contours found: " << contours.size() << endl; )

    //Find largest contour
    DB(cout << "### Find largest contour" << endl;)
    int largestContourArea = -1;
    int largestContourIndex = -1;
    if(contours.size() != 0) {
      largestContourArea = findLargestContour(contours, &largestContourIndex);
      DB( cout << "Largest contour index: " << largestContourIndex << endl; )
    }

    // Process contour area:
    DB(cout << "### Process contour area" << endl;)
    if(largestContourIndex != -1) {
      DB( cout << "Object found. Size: " << largestContourArea << endl; )
      // Extract contour area from frame
      cv::Mat imgROI;
      cv::Rect rect = cv::boundingRect(contours[largestContourIndex]);

      DB ( cout << "Extracting ROI from image, h: " << rect.height << " w: " << rect.
  width << endl; )
      //Since the object used was only wearing a red sweater,
      //the ROI needs to be expanded somewhat larger in order to do the detection of a
  face
      cv::Rect newRect;
      newRect = rect + cv::Size(rect.area()*0.05,rect.area()*0.05);
      Size imgSize = frame.size();
      // Make sure the ROI is contained to the original image
      if(newRect.height > imgSize.height) newRect.height = imgSize.height;
      if(newRect.width > imgSize.width) newRect.width = imgSize.width;

      // Set location of rect
      newRect.x -= newRect.width/2;
      newRect.y -= newRect.height/2;

```

```
        // Make sure the ROI stays within the boundaries of the original image
        if(newRect.x < 0) newRect.x = 0; // Test left side
        if(newRect.y < 0) newRect.y = 0; // Test top
        if( (newRect.x + newRect.width) > imgSize.width ) newRect.width -= ( (newRect.x +
     newRect.width) - imgSize.width);
        if( (newRect.y + newRect.height) > imgSize.height ) newRect.height -= ( (newRect.
    y + newRect.height) - imgSize.height);

        DB ( cout << "Extracting ROI from image, h: " << newRect.height << " w: " <<
    newRect.width << endl; )
        imgROI = frame(newRect);
        Mat result = imgROI.clone();

        //Save ROI
        DB(   fileName = "/media/mmc0/blob/ROI_" + ss.str() + ".jpg";
            imwrite(fileName, imgROI); )

        // Do the actual processing
        if(largestContourArea > 100) {
          DB( cout << "Object found and above threshold" << endl; )
          if(findFaceInImage(imgROI, result)) {
            DB( duration.timerEnd(); )   // Timer end
            DB( cout << duration.getTimeInMS() << endl; )

            DB( cout << "Face found, image saved" << endl; )
            DB(   fileName = "/media/mmc0/blob/result_" + ss.str() + ".jpg";
                imwrite(fileName, result);
                myFile << "Time used on file: " << fileName << " \t was: " << duration.
    getTimeInMS() << endl; )
          }
        }
      }
    }
  }
  pthread_exit(NULL);
  return 0;
}

// Starts the two threads and never ends.
// Argument 1 is the mode selector between blob or no blob detection.
int main(int argc, char** argv) {
  if(argc < 2) {
    cout << "Arguments needs to be input to this program. " << endl;
    cout << "Arg 1: mode selector between blob (1) or no blob (0) detection" << endl;
    exit(0);
  }
  int modeSelector = atoi(argv[1]);
  if( (modeSelector == 0) || (modeSelector == 1)) {
    GLOB_thread_data_t.algorithmModeSelector = atoi(argv[1]);
  }
  else {
    cout << "Error in input argument 1. Needs to be 1 for blob, or 0 for no blob" << endl
    ;
    exit(0);
  }

  pthread_t getImageFromWebcam, objectAlgorithm_t;

  pthread_mutex_init(&frameLock, NULL);
  pthread_barrier_init(&getStartedBarrier, NULL, 2);

  int createThread1 = pthread_create(&getImageFromWebcam, NULL, getWebcamFrames, NULL);//
    reinterpret_cast<void*>(&thread_data_t));
  if(createThread1 != 0) exit(EXIT_FAILURE);
  int createThread2 = pthread_create(&objectAlgorithm_t, NULL, objectAlgorithm, NULL);//
    reinterpret_cast<void*>(&thread_data_t));
  if(createThread2 != 0) exit(EXIT_FAILURE);

  pthread_join(getImageFromWebcam, NULL);
  pthread_join(objectAlgorithm_t, NULL);

  return 0;
}
```

A.5 Timer Function

The code in Listing A.5 and Listing A.6 is used for determining the run-time of the algorithms that are tested on the *BeagleBone Black* in this thesis. The code assumes that the *OpenCV* library is installed.

Listing A.5: Header file for run-time calculations

```


#ifndef FUNCTIMER_H_
#define FUNCTIMER_H_

class funcTimer{
private:
  double duration;
public:
  funcTimer();
  virtual ~funcTimer();
  void timerStart();
  void timerEnd();
  double getTimeInMS();
};



#endif
```

Listing A.6: Functions for run-time calculations

```



```

```

#include "funcTimer.h"
#include <iostream>
#include <opencv2/core/core.hpp>

funcTimer::funcTimer() {
   duration = 0;
}

funcTimer::~funcTimer() {
   // TODO
}

void funcTimer::timerStart() {
   duration = static_cast<double>(cv::getTickCount());
}

void funcTimer::timerEnd() {
   duration = static_cast<double>(cv::getTickCount()) - duration;
}

double funcTimer::getTimeInMS() {
   return ( duration / cv::getTickFrequency() );
}
```

A.6 APM 2.6: Sniffing Application

The *BeagleBone Black* was used during this Thesis for sniffing the packets sent from the *APM 2.6* autopilot system to its radio module. The code in Listing A.7 is an application that reads the packets sent by the *APM 2.6* and writes some of the contents to the terminal.

Listing A.7: APM 2.6 Sniffing Application

```
/*
*
*      Written by Gaute Gamnes, NTNU
*        spring of 2014. Date: 140521
*
*        /* Code borrowed and modified from several sources...
*      Serial information:
*      http://en.wikibooks.org/wiki/Serial_Programming/termios
*      http://blog.eduardofleury.com/archives/2007/11/16
*      MAVLink: as well as serial:
*
http://diydrones.com/forum/topics/need-help-for-serial-comunication?id=705844%3ATopic%3
    A1038239&page=1#comments
*
*        Compile with:
*        gcc main.c -o programExe -I ../../mavlink/Hello-World/mavlink/include/mavlink/v1
    .0/common
*
*        Before running the program UART needs to be enabled on the BeagleBone.
*       UART 4 is used in this application; Pin 11(Rx) and Pin 13(Tx) on Header P9.
*       To enable UART 4: run command
*       echo BB−UART4 > /sys/devices/bone_capemgr.*BACKSLASHslots
*       The BACKSLASH is a / but can not be inserted in the comment section of this code.
*
*   Using MAVLink: Download the newest .zip file from https://github.com/mavlink/mavlink/
    downloads
*   Unzip on the BeagleBone Black and include <mavlink.h> in the code.
*   When compiling the program, include the −I flag during compile and append the path to
     the
*   ...../include/mavlink/v1.0/common folder.
*   If the MAVLink headers are included in the PATH environment of the BeagleBone it
    might not be needed with
*   the compile flag.
*
*/

#include <termios.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <sys/ioctl.h>

#include <mavlink.h>

#define TTY_SERIAL_PORT "/dev/ttyO4" // Path to serial port, UART 4, pin 11(Rx) and 13(Tx
    )

int init_serial_communication(char* port) {
  int tty_fd = 0; // The returned file handle for the device
  tty_fd = open(port, O_RDWR | O_NOCTTY | O_NDELAY);
  // O_RDWR − opens port for read and write
  // O_NOCTTY − The port never becomes the controlling terminal of the process
  // O_NDELAY − Non−blocking IO
  if(tty_fd == −1) {
    printf("Failed to open port\n");
    return tty_fd;
  }

  struct termios serial_options, original_serial_options;
  // Get current terminal settings
  if(tcgetattr(tty_fd, &original_serial_options) == −1) {
    printf("Error getting tty attributes %s − %s(%d).\n", tty_fd, strerror(errno), errno)
    ;
    return −1;
  }
  serial_options = original_serial_options; // Copy settings to new and play with them

  // Set termianl to something like raw mode
  //Input is availa char by char, echoing is disabled, no special processing of IO char
  cfmakeraw(&serial_options);
  // Set input and output speed?
  cfsetspeed(&serial_options, 57600);
  // 8N1
  serial_options.c_cflag &= ~PARENB;
  serial_options.c_cflag &= ~CSTOPB;
  serial_options.c_cflag &= ~CSIZE;
  serial_options.c_cflag |= CS8;
  // no flow control
  serial_options.c_cflag &= ~CRTSCTS;
  serial_options.c_cflag |= CREAD | CLOCAL;  // turn on READ & ignore ctrl lines
  serial_options.c_iflag &= ~(IXON | IXOFF | IXANY); // turn off s/w flow ctrl
  serial_options.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG); // make raw
  serial_options.c_oflag &= ~OPOST; // make raw
  // see: http://unixwiz.net/techtips/termios-vmin-vtime.html
  serial_options.c_cc[VMIN]  = 0;
  serial_options.c_cc[VTIME] = 20;

  // Apply settings
  if(tcsetattr(tty_fd, TCSANOW, &serial_options) == -1) {
    printf("Error setting tty attributes %s - %s(%d).\n", tty_fd, strerror(errno), errno)
    ;
    return -1;
  }
  // clear all flags on descriptor, enable direct IO
  if(fcntl(tty_fd, F_SETFL, 0)) {
      printf("Error setting tty non-blocking %s - %s(%d).\n", tty_fd, strerror(errno),
    errno);
      exit(EXIT_FAILURE);
  }

  return tty_fd;
}

int main(int argc, char* argv) {
  const char *dev = "/dev/ttyO4"; // Path to serial port, UART 4, pin 11 and 13

  int tty_fd = init_serial_communication(TTY_SERIAL_PORT);
  if(tty_fd == -1) {
    printf("Error opening serial port!\n");
    exit(EXIT_FAILURE);
  }

  /* Receive information from serial port */
  char rec_byte;
  //NEEDS MAVLINK for this to work
  mavlink_status_t msg_status;
  mavlink_message_t msg;
  static const int FIXED_HEADER_LEN = MAVLINK_NUM_HEADER_BYTES +
    MAVLINK_NUM_CHECKSUM_BYTES;
  int serial_fd = tty_fd; //would have to be used if sent into a thread, thus taking the
    value of args.
  while(1) {
    int rdChar = read(serial_fd, &rec_byte, 1);
    if(rdChar == -1) {
        perror("Error reading serial port");
    }
    if(rdChar == 0) {
        usleep(10*1000); //wait 10msec try again
        continue;
    }
    // Read the packet that is receive using the MAVLink provided function parse char.
    if ( mavlink_parse_char(MAVLINK_COMM_0, rec_byte, &msg, &msg_status) ) {
      //Packet received
      printf("Serial -> BBB: SYS: %d, COMP: %d, LEN: %d, MSG ID: %d\n",
          msg.sysid, msg.compid, msg.len, msg.msgid);
      // Print the packet
      int j;
      for (j=0; j<msg.len + FIXED_HEADER_LEN; j++) {
        printf("0x%02x ", *((uint8_t*)&msg.magic + j) & 0x00ff);
        if(((j+1) % 8) == 0) printf("\n");
      }
      // To tidy up the print outs
      if(! ((j+1) % 8) == 0) printf("\n");

      //Switch case to handle messages,
      // Take mavlink_message_t* msg as input.
      switch(msg.msgid) {
        case MAVLINK_MSG_ID_HEARTBEAT:
          printf("Heartbeat received\n");
          break;
        case MAVLINK_MSG_ID_COMMAND_LONG:
          printf("Command long received\n");
          break;
        case MAVLINK_MSG_ID_GPS_RAW_INT:
          printf("GPS raw int received\n");
          break;
        case MAVLINK_MSG_ID_GPS_STATUS:
          printf("GPS status received\n");
          break;
        case MAVLINK_MSG_ID_RAW_IMU: //#27
          printf("Raw IMU received\n");
          break;
        case MAVLINK_MSG_ID_SCALED_PRESSURE: //#29
          printf("Scaled pressure received\n");
          break;
        case MAVLINK_MSG_ID_ATTITUDE: //#30
          printf("Attitude received\n");
          mavlink_attitude_t packet;
          mavlink_msg_attitude_decode(&msg, &packet);
          printf("Pitch: %f, yaw: %f, roll: %f \n",
            packet.pitch, packet.yaw, packet.roll);
          break;
        case MAVLINK_MSG_ID_GLOBAL_POSITION_INT: //#33, filtered GPS
          printf("Global position int received\n");
          mavlink_global_position_int_t packet2;
          mavlink_msg_global_position_int_decode(&msg, &packet2);
          printf("Latitude: %d, Longitude: %d, Altitude: %d \n",
              packet2.lat, packet2.lon, packet2.alt);
          //Lat and lon expressed as *1E7. Alt in meters, * 1000 (mm).
          break;
        case MAVLINK_MSG_ID_VFR_HUD: //#74, Metrics for fixed wing
          printf("VFR_HUD received\n");
          break;
        default:
          //Do nothing
          printf("MSG ID not handled\n");
      }
    }
  }

  /* Done receiving from serial port */

  // Restore terminal settings NO NEED in this particular case, cause program never ends
  //tcsetattr(0, TCSANOW, &original_serial_config);

  close(tty_fd);

  return 0;
}
```

A.7 XBee: Serial Communication Application

The code listed in Listing A.8 is the code used for achieving communication between the *BeagleBone Black* and *XBee* modules. The application consists of two threads, one for handling received messages and one for sending messages.

Listing A.8: XBee Serial Communication Application

```
/*
*       Written by Gaute Gamnes, NTNU
*       spring of 2014. 140521
*
*   Code borrowed and modified from several sources...,
*       Serial information:
*     http://en.wikibooks.org/wiki/Serial_Programming/termios
*     http://blog.eduardofleury.com/archives/2007/11/16
*
*     compile with: g++ main.cpp -o programExe -pthread
*
*   UART1 needs to be enabled for this to work
*   Run command: echo BB-UART1 > /sys/devices/bone_capemgr.*BACKSLASHslots
*   BACKSLASH must be replaced with an actual backslash /
*   Run the command on the BBB itself to enable UART1 on pin
*   P9:24 (Tx), P9:26 (Rx).
*
*/

#include <termios.h>
#include <iostream>
#include <stdio.h>
```

```
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <sys/ioctl.h>

using namespace std;

extern "C" {
  #include <pthread.h>
}

#define TTY_SERIAL_PORT "/dev/ttyO1" // Path to serial port, UART 1, UART1 pins are P9:
    TX(24), RX(26)

int init_serial_communication(char* port) {
  int tty_fd = 0; // The returned file handle for the device
  tty_fd = open(port, O_RDWR | O_NOCTTY); // | O_NDELAY); Could use delay?
                      // O_RDWR - opens port for read and write
                      // O_NOCTTY - The port never becomes the controlling terminal of
    the process
                      // O_NDELAY - Non-blocking IO
  if(tty_fd == -1) {
    printf("Failed to open port\n");
    return tty_fd;
  }

  struct termios serial_options, original_serial_options;
  if(tcgetattr(tty_fd, &original_serial_options) == -1) {   // Get current terminal
    settings
    printf("Error getting tty attributes %s - %s(%d).\n", tty_fd, strerror(errno), errno)
    ;
    return -1;
  }
  serial_options = original_serial_options; // Copy settings to new and play with them

  cfsetspeed(&serial_options, 9600); // Set input and output speed
  // 8N1
  serial_options.c_cflag &= ~PARENB;
  serial_options.c_cflag &= ~CSTOPB;
  serial_options.c_cflag &= ~CSIZE;
  serial_options.c_cflag |= CS8;
  // no flow control
  //serial_options.c_cflag &= ~CRTSCTS;
  serial_options.c_cflag |= CREAD | CLOCAL;  // turn on READ & ignore ctrl lines
  serial_options.c_iflag = 0;
  serial_options.c_lflag = 0;
  serial_options.c_oflag = 0;

  //Might not be needed
  //serial_options.c_cc[VMIN]  = 0; //Wait for an array of a certain size to enter buffer
     before moving on?
  //serial_options.c_cc[VTIME] = 1; //Timeout after 0.1 seconds

  // Apply settings
  if(tcsetattr(tty_fd, TCSANOW, &serial_options) == -1) {
    printf("Error setting tty attributes %s - %s(%d).\n", tty_fd, strerror(errno), errno)
    ;
    return -1;
  }
  // clear all flags on descriptor, enable direct IO
  if(fcntl(tty_fd, F_SETFL, FNDELAY)) {
      printf("Error setting tty non-blocking %s - %s(%d).\n", tty_fd, strerror(errno),
    errno);
      exit(EXIT_FAILURE);
  }

  return tty_fd;
}

void* receiveFromSerial(void* argument) {
  int tty_fd = *(reinterpret_cast<int*>(argument));

  /* Receive/send information from serial port */
  char rec_byte;
  while(1) {
    int rdChar = read(tty_fd, &rec_byte, 1);
    if(rdChar > 0) {
      cout << "Rec_byte: " << rec_byte << endl;
    }

  }
  /* Done receiving/sending from serial port */

  return 0;
}

void* sendToSerial(void* argument) {
  int tty_fd = *(reinterpret_cast<int*>(argument));

  while(1) {
    char input[256];
    cout << (">");
    fgets(input, 256, stdin);
    int size = strlen(input);
    cout << "Input: " << input << " size: " << size << endl;

    write(tty_fd, input, size);
  }

  return 0;
}

int main(int argc, char** argv) {
  int tty_fd = init_serial_communication(TTY_SERIAL_PORT);
  if(tty_fd == -1) {
    printf("Error opening serial port!\n");
    exit(EXIT_FAILURE);
  }

  pthread_t receive_t, send_t;

  int createThread1 = pthread_create(&receive_t, NULL, receiveFromSerial,
    reinterpret_cast<void*>(&tty_fd));
  if(createThread1 != 0) cout << "Error creating receive thread" << endl;
  int createThread2 = pthread_create(&send_t, NULL, sendToSerial, reinterpret_cast<void
    *>(&tty_fd));
  if(createThread2 != 0) cout << "Error creating send thread" << endl;
  pthread_join(receive_t, NULL);
  pthread_join(send_t, NULL);

  close(tty_fd);

  return 0;
}
```

Appendix B

GPS Localization

In order to test the object localization application explained in this Thesis, some GPS formulas had to be used in order to calculate the bearing and distance between two GPS locations, and to calculate a new GPS location given an old one and some movement along the latitude and longitude axis from this old location. Some of these formulas were not evaluated properly and thus their credibility was not determined. The formulas might have to be changed to get a more accurate and reliable result.

In most of the formulas used, the angles in radians are used to calculate the different results.

B.1 Calculate the bearing and distance between two GPS locations

In order to determine the bearing, or compass course between two GPS locations, a formula was found and tested to be functional, and thus used in the work of this Thesis [GPS, a].

Equation B.1 shows a formula for calculating the bearing between two GPS locations.

$$\begin{aligned} \texttt{bearing} = \texttt{MOD}(& \\ & \arctan(\, \cos(\texttt{lat1}) * \sin(\texttt{lat2}) - \sin(\texttt{lat1}) * \cos(\texttt{lat2}) * \cos(\texttt{long2} - \texttt{long1}); \\ & \sin(\texttt{long2} - \texttt{long1}) * \cos(\texttt{lat2})\,) \\ & ;\, 2 * \pi) \end{aligned} \tag{B.1}$$

All variables are in radians. The Lat1 variable is the latitude of the first GPS location. Lat2 is the latitude of the second location. Long1 and Long2 are the longitude of the positions. The result is the bearing between the first GPS location to the second location in radians.

To calculate the distance between two GPS locations, another formula was found and tested to provide satisfactory results [GPS, b]. Equation B.2 shows a formula for calculating this distance.

$$\begin{aligned}\texttt{distance(km)} &= \arccos(\cos(\texttt{lat1}) * \cos(\texttt{lat2}) * \cos(\texttt{long1} - \texttt{long2}) \\ &\quad + \sin(\texttt{lat1}) * \sin(\texttt{lat2})) * \texttt{earthRadius}\end{aligned} \tag{B.2}$$

The variables lat1, lat2, long1, long2 are the same as in the bearings formula, and the result of Equation B.2 is the distance between the two GPS locations in km.

B.2 Calculate a new GPS location given movement in meters along latitude and longitude axis

Since the testing and implementation of the localization application was done as the last task of this Master Thesis, there was not enough time to study how the new latitude and longitude could be calculated given a movement in meters from another location. A formula found on a forum website was tested, and the tests proved the function to perform quite well given the distances needed during this project, and was therefore decided to be used in the localization application. Equation B.3 lists a formula for calculating a new latitude and longitude position, given a starting GPS position, and movement from that position in meters along the latitude and longitude axis [GPS, c]. All positions are in this formula in decimal degrees.

$$\begin{aligned}\texttt{newLat(deg)} &= \texttt{cameraLat} + [(\frac{\texttt{moveInX}}{1000})/(\texttt{earthRadius}))] * (180/\pi) \\ \texttt{newLong(deg)} &= \texttt{cameraLong}[(\frac{\texttt{moveInY}}{1000})/(\texttt{earthRadius} * \cos(\texttt{newLat} * \pi/180))] * (180/\pi)\end{aligned} \tag{B.3}$$

As explained, the formula gives the new latitude and longitude after a move from one GPS position, given a movement in meters, moveInX, along the latitude axis, and a movement in meters, moveInY, along the longitude axis.

As explained, this function was not evaluated properly, which possibly resulted in some small errors in the results when it was used. Due to time limitations, this was however not corrected during this Thesis and will have to be fixed in future implementations in order to give a correct GPS localization result. However, since the formula was tested to work, and the distances used in this Thesis were so small, the formula will probably not have caused that much error in the result.

B.3 OpenCV webcam application for calculating GPS position of an object

The code listed in this section are utilizing the formulas listed previously in this appendix regarding GPS location calculations. The application is also stationary, meaning that the rotation and position of the camera needs to be coded into the source code and the application compiled before it can be tested. The camera matrix used for determining the camera frame coordinate given the pixel coordinate was found using the calibration program of the OpenCV library. This calibration matrix will thus only be valid for this particular webcam and the resolution settings chosen to be used during the calibration.

Listing B.1: Find GPS location of object given pixel position

```
/*
 * findGPSlocationGivenPixel
 *
 *  Created on: 31. mai. 2014
 *      Author: Gaute Gamnes
 */


// This applications finds the GPS location of an object in the frame, given
// that the GPS location of the camera and its orientation and heading are known
/*
 * This application assumes that a webcamera is connected, i.e. find a webcam at capture
    (1)
 * and also that OpenCV is installed and the libraries are correctly linked.
 *
 * The camera orientation and GPS location/height needs to be hardcoded into the program,
 * could be done in run-time if desired.
 */

#include <stdio.h>
#include <iostream>
#include <string>
#include <time.h>
#include <sstream>

extern "C" {
  #include <pthread.h>
}

#include <opencv2/core/core.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <opencv2/nonfree/nonfree.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/objdetect/objdetect.hpp>
#include <opencv2/contrib/contrib.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/stitching/stitcher.hpp>
using namespace std;
using namespace cv;

#define PI 3.14159265358979323846264

struct thread_data_t {
  Mat frame;
  int test;
  int pixelX;
  int pixelY;
}GLOB_thread_data_t;

void onMouse(int event, int x, int y, int, void*) {
  if( event != CV_EVENT_LBUTTONDOWN ) return;
  Point pt = Point(x,y);
  for(int i=0; i<10; i++) {
    cout << "x: " << pt.x << "\t y: " << pt.y << endl;
    GLOB_thread_data_t.pixelX = pt.x;
    GLOB_thread_data_t.pixelY = pt.y;
  }
}

void * getWebcamFrames(void * argument) {
  //thread_data_t *data = (reinterpret_cast<thread_data_t*>(argument));
  VideoCapture capture(1);
  capture.set(CV_CAP_PROP_FRAME_HEIGHT, 480);
  capture.set(CV_CAP_PROP_FRAME_WIDTH, 640);

  Mat frame;
  namedWindow("frame", CV_WINDOW_AUTOSIZE);

  // Rotation matrices etc, pi is declared as PI

  // ---------------INPUT, camera orientation ------------------------------
  double phi = 1.3474889519; // om x
  double theta = 0; // om y
  double psi = PI/2 + 1.0847126821; // om z
  // -----------------------------------------------------------------------

  Mat rotX = (Mat_<double>(3,3) <<  1,0,0,
                     0, cos(phi), sin(phi),
                     0, -sin(phi), cos(phi) );
  Mat rotY = (Mat_<double>(3,3) <<  cos(theta), 0, -sin(theta),
                     0, 1, 0,
                     sin(theta), 0, cos(theta) );
  Mat rotZ = (Mat_<double>(3,3) <<  cos(psi), sin(psi), 0,
                     -sin(psi), cos(psi), 0,
                     0, 0, 1 );
  Mat rotW2C = rotX * rotY * rotZ;
  //cout << "RotXYZ" << " " << rotXYZ << endl;
  Mat K = (Mat_<double>(3,3) <<   8.1331102645202134e+02, 0, 3.1950000000000000e+02,
                   0, 8.1331102645202134e+02, 2.3950000000000000e+02,
                   0, 0, 1 );


  // ---------------INPUT, camera height and location lat long------------------------------
  Mat Cw = (Mat_<double>(3,1) <<  0,0,-16 );
  double cameraLatitudeCoordinate = 63.41856;
  double cameraLongitudeCoordinate = 10.40109;
  // ------------------------------------------------------------------------------------------

  Mat IandCw = (Mat_<double>(3,4) <<  1,0,0, -Cw.at<double>(0,0),
                    0,1,0, -Cw.at<double>(1,0),
                    0,0,1, -Cw.at<double>(2,0) );
  Mat P = K * rotW2C * IandCw;
  //cout << "MAT:" << " " << P << endl;
  Mat Pwith3columnRemoved = (Mat_<double>(3,3) <<
      P.at<double>(0,0), P.at<double>(0,1), P.at<double>(0,3),
      P.at<double>(1,0), P.at<double>(1,1), P.at<double>(1,3),
      P.at<double>(2,0), P.at<double>(2,1), P.at<double>(2,3)  );

  // Calculate Xw homogeneous using the pixel Xi
  Mat Xi = (Mat_<double>(3,1) << 320, 240, 1 );
  Mat Xw = Pwith3columnRemoved.inv() * Xi;
  Xw = Xw / Xw.at<double>(2,0);
  cout << "MAT:" << " " << Xw << endl;

  // Declare to make sure they contain value
  GLOB_thread_data_t.pixelX = 0;
  GLOB_thread_data_t.pixelY = 0;

  int key = 0;
  while(key != 27) {
    capture >> frame;
    cv::circle(frame, Point(320,240), 5, CV_RGB(0,0,250), 2, 8);
    setMouseCallback("frame", onMouse, 0);
    //mat, point, radius, color, thickness, connectivitiy
    imshow("frame", frame);
    imwrite("locationResult.jpg", frame);

    Xi.at<double>(0,0) = GLOB_thread_data_t.pixelX;
    Xi.at<double>(1,0) = GLOB_thread_data_t.pixelY;
    Xw = Pwith3columnRemoved.inv() * Xi;
    Xw = Xw / Xw.at<double>(2,0);
    // Radius earth is assumed to be 6371, new LAT and LONG are calculated based on
    formulas found
    // http://stackoverflow.com/questions/7477003/calculating-new-longtitude-latitude-
    from-old-n-meters
    double newLatCoordinate = cameraLatitudeCoordinate + ( (Xw.at<double>(0,0)/1000) /
    6371 ) * (180/PI);
    double newLongCoordinate = cameraLongitudeCoordinate +
        ( (Xw.at<double>(1,0)/1000) / (6371*cos(PI*newLatCoordinate/180)) ) * (180/PI);
    cout << "World Coordinate: (" << Xw.at<double>(0,0) << ", " << Xw.at<double>(1,0) <<
    "), New GPS lat/long: ("
        << newLatCoordinate << ", " << newLongCoordinate << ")" << endl;

    key = waitKey(1);
  }
  pthread_exit(NULL);
  return 0;
}

int main(int argc, char** argv) {
  pthread_t getImageFromWebcam;

  std::cout.precision(8);

  int createThread1 = pthread_create(&getImageFromWebcam, NULL, getWebcamFrames, NULL);//
    reinterpret_cast<void*>(&thread_data_t));
  if(createThread1 != 0) exit(EXIT_FAILURE);

  pthread_join(getImageFromWebcam, NULL);

  return 0;
}
```

Appendix C

System Components

Table C.1 is a table listing all the components included in the prototype UAS created in this Thesis. The status of the components is either OK, or MISSING. If the status of a components if OK, the components is present in the system and possible to use for further development. If the status is MISSING, the components is not present and will thus have to be acquired for the system to work. The other equipment not necessarily mounted on board the prototype UAS, but which needs to be acquired to be able to utilize the system is listed in Table C.2.

Table C.1: System Components

Component	**Details**	**Quantity**	**Weight**	**Price /unit**	**Status**
Airframe:	MaxiSwift Flying wing	1	700g	$98	OK
Battery:	Turnigy 5000mAh 3S 20C Lipo Pack	2	412g	$25	OK
Motor:	Turnigy Park450 Brushless Outrunner 1050kv	1	66g	$15	OK
Propeller:	Composite Propeller, 10 x 5 E	1	75g	$2	OK
ESC:	20A 2-4S 5V/3A BEC	1	30g	$7	OK
Servo:	HD-1810MG Digital Servos	2	16g	$15	OK
Autopilot:	APM 2.6	1	33g	$160	MISSING
GPS and Compass:	u-blox GPS with compass XT60	1	17g	$79	MISSING
Power Module:	APM Power Module with Connectors Kit	1	~20g	$25	MISSING
Telemetry Radio:	APM Telemetry Radio 433MHz	1	~4g	$100	MISSING
Embedded Computer:	BeagleBone Black	1	40g	$45	MISSING
Webcam:	Logitech c270 HD Webcam	1	227g	$50	MISSING
Total all units:	**Weight:** 1801g **Price:** $661				

Table C.2: Extra Components

Component	Details	Price /unit	Status
Battery Charger:	Hobbyking ECO SIX 80W 6A 2 6S Battery Balance Charger AC/DC w/PSU	$25	OK
RC Controller and Receiver:	DX7s and AR8000	$433	OK
GoPro Camera:	GoPro Hero 3		MISSING
Laptop:	Acer, Windows 7		MISSING