NTNU
Innovation and Creativity

Ground Station and Hardware Peripherals for Fixed-wing UAV: CyberSwan

Mikael Kristian Eriksen

Master of Science in Engineering Cybernetics
Submission date: June 2007
Supervisor: Amund Skavhaug, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

A ground station is needed for supervision and control for the AUAV project.
The main objective for this thesis is the development of this ground station.

The assignment involves among other things:
- Research necessary background material.
- Design a system with radio communication, GPS and video transmission.
- Find suitable equipment for this system

To the extent possible within the given timeframe:
- Implement the necessary hardware and software
- Evaluate the achieved results

Assignment given: 29. January 2007
Supervisor: Amund Skavhaug, ITK

Preface

This report is the result of the master's thesis *Ground Station and Hardware Peripherals for Fixed-wing UAV: CyberSwan*. The thesis concludes the master's degree with the specialization *Dedicated Computer Systems* at the Norwegian University of Science and Technology (NTNU), under the Department of Engineering Cybernetics. The project has taken place in the spring semester in the 2nd year of the 2-year Master of Science study for Bachelor engineers.

I have chosen to write this report in English. I chose this in order to learn the the language better, since English often is used as a professional work language.

First of all I have to thank Jon Bernhard Høstmark and Edgar Bjørntvedt for including me in the AUAV project. Next I have to thank my supervisor, Associate Professor Amund Skavhaug, for his guidance and positive enthusiasm.

In addition there has been several other individuals that have helped me during the project, and deserves a big "thank you". Terje Haugen and Hans Jørgen Berntsen, at the departments metal workshop, for being extremely forthcoming and efficient making all odd parts used for mounting equipment. Peder Martin Evjen, Radiocrafts AS, for sponsoring RF equipment. Fellow students Audun Sølvberg, for helping me with data-logging from the IMU, and Eirik Tveiten, for his numerous LabVIEW-tips.

Trondheim, June 26, 2007

Mikael K. Eriksen

Abstract

In this master's thesis, a ground station (GS) for the fixed-wing UAV: CyberSwan (CS), has been developed. The CS was designed for surveillance purposes, and two other master's theses deals with the work of making it autonomous (Høstmark (2007) and Bjørntvedt (2007)). Having a GS will make it possible to communicate with the CS in-flight, and present data and video from the CS through communication devices.

The GS has been realised using LabVIEW development software from National Instruments (2007). A CS simulator was also developed in LabVIEW for test purposes. In addition was a Global Position System (GPS) receiver board, and a Radio Frequency (RF) communication board, developed. The GPS receiver was used to position the GS, and used as a source for position correction data. The RF communication board was developed for mounting in the CS and to be connected to its computer system to enable communication with the GS. The GS used a RF demo board for communication. A wireless camera was mounted on the CS for in-flight video surveillance, and a ultrasound ranging device was tested intended to be used in a autonomous landing situation.

A hardware in the loop (HIL) test was performed to test the GS's communication capacity. Here the developed CS simulator was used, as the CS computer system was not completed (Bjørntvedt 2007). The test proved it possible to transfer a CS status message at 4 Hz, making the chosen communication device a good choice for the intended purpose.

Contents

List of Figures

List of Tables

Abbreviations

ADC	Analog-to-Digital Converter
AFCS	Aircraft Flight Control System
ASCII	American Standard Code for Information Interchange
AUAV	Autonomus Unmanned Aerial Vehicle
AV	Audio-Visual
CNC	Computer Numerical Control
COTS	Commercial Off-The-Shelf
CS	CyberSwan
CSE	Control Segment
DGPS	Differential GPS
DoD	Department of Defense
EGNOS	European Geostationary Navigation Overlay Service
GCS	Ground Control Station
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GS	Ground Station
GUI	Graphical User Interface
HIL	Hardware In the Loop
IMU	Inertial Measurement Unit
LabVIEW	Laboratory Virtual Instrumentation Engineering Workbench
LDO	Low Drop-Out
NMEA	National Marine Electronics Association
NTNU	Norwegian University of Science and Technology
R&D	Research & Development
RF	Radio Frequency
SMD	Surface Mount Device
SS	Space Segment
SV	Space Vehicles
TTL	Transistor-Transistor Logic
UAV	Unmanned Aerial Vehicle

UERE ..... User Equivalent Range Errors
US ..... User Segment

The higher we soar the smaller we appear to those who cannot fly.

– Friedrich Nietzsche

Chapter 1

Introduction

1.1 Background

During the fall semester of 2006, Høstmark (2006) and Bjørntvedt (2006) started the work on the first autonomous fixed-wing unmanned aerial vehicle (AUAV), at the Norwegian University of Science and Technology (NTNU). Previously there have been several projects involving autonomous control undertaked at the Department of Engineering Cybernetics; a helicopter, CyberEagle (Ellingsen 2002), several robots for the Eurobot competition, several under-water vehicles and continous work on a bicycle, CyberBike (Sølvberg 2007).

The work on the AUAV resulted in a unmanned aerial vehicle (UAV) platform prototype, Figure 1.1 (Høstmark 2006), and a design of the instrumentation system for this prototype (Bjørntvedt 2006). The projects were continued for master's theses and expanded into one additional field, a ground station. The UAV was also given the nickname CyberSwan (CS).

Figure 1.1: Fixed-wing platform prototype.

Jon Bernhard Høstmark's thesis, *Modelling, Simulation and Control of Fixed-wing UAV: CyberSwan*, concerns, as the title implies, modelling, simulation and control of the fixed-wing aircraft in question. It also includes the implementation of a aircraft flight control system (AFCS).

Edgar Bjørntvedt's thesis, *Instrumentation of Autonomous Fixed-wing UAV*, concerns the development of the AUAV's instrumentation and main computer system. Høstmark's AFCS is to be implemented in this computer system, and here the communication device to communicate with the ground station is to be connected.

The fixed-wing UAV system was designed to be suitable for surveillance purposes. An important property was the use of electrical propulsion and commercial off-the-shelf (COTS) components, making it a low cost option to expensive tailored products. The airframe's design is the result of compromises between stability, linearity and manoeuverability. This gives the possibility for stable control with standard control theory (Høstmark 2006).

1.2 Motivation

The motivation for expanding the whole project into also involving a ground station is ease of operation and to add some level of security. A ground station will give the chance to communicate with the CS during the flight. It will also give the possibility to monitor it's position and other system data. The ultimate goal would be to make the CS autonomously fly a preprogrammed navigation route. Positioning the ground station will also give the CS a reference that could be used in an autonomous landing situation.

This thesis concerns the development of a ground station (GS) for communication with the CS. Navigational, sensor and visual data are to be transmitted and presented. It also imply the support of hardware peripherals for the CS: a communication device, a ultrasound ranger and a wireless camera system.

The main objective is navigational support. This implies presentation of position and other data transferred during flight. It is therefore concidered important to have communication with the CS at all time. The CS is quite small, and has limited space for equipment. It's power source are batteries which gives a limitation in current consumption for added electronics. Also, noise influence on the instrumentation system have to be limited to a minimum. Figure 1.2 gives an overview of the total CS AUAV system.

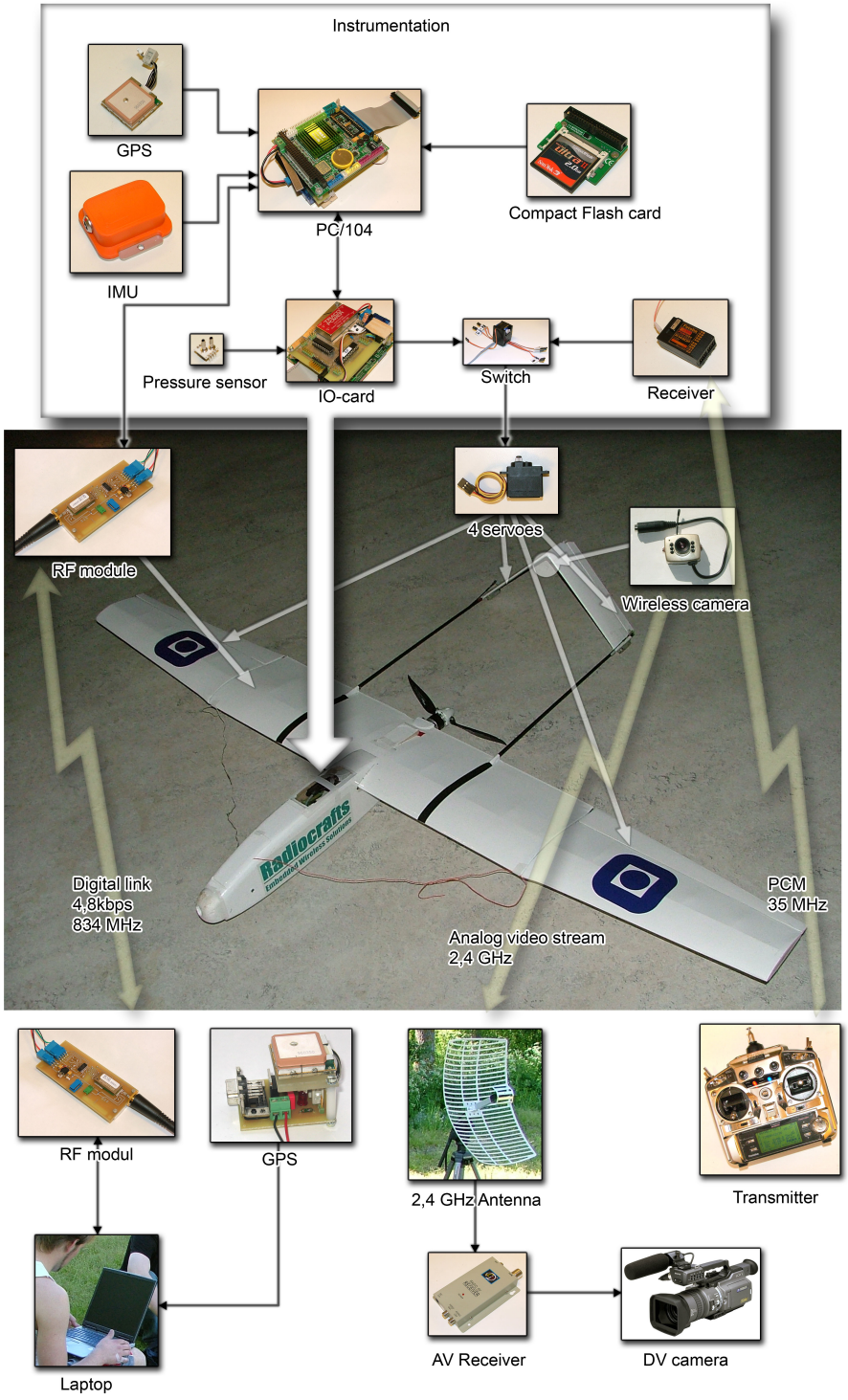


Figure 1.2: Overview of the total AUAV system.

CyberSwan general characteristics

- **Wing Span:** 1.7 m
- **Length:** 1.1 m
- **Speed:** 10 - 30 m/s
- **Fuel:** Lithium Polymer Batteries
- **Propulsion:** Two Brushless Electric Motors
- **Weight:** 2.8 kg (Max.)
- **Payload:** 1.0 kg (Max.)
- **Flight Time:** 2 Hours

1.3 Current System

The Turkish company Baykar Machine Inc. delivers complete designs from miniature to tactical range UAV systems, both helicopter and fixed-wing platforms are available.

Main system components such as avionics, ground control station, digital encrypted data link, operator interface and airborne platforms are designed, engineered and integrated in-house through research & development (R&D) projects, where some have been coordinated with Istanbul Technical University (Baykar Machine Inc 2007).

One of the platforms available are the Bayraktar Mini UAV System (Figure 1.3) which provides a fixed-wing UAV design close to our own CS. The mini UAV system is designed for aerial reconnaissance and surveillance activities with deliveries to the Turkish Military (Franchi 2006). The V-tail form, aerodynamically stable fixed-wing platform is designed and manufactured in a modular fashion with materials such as carbon fiber, kevlar, glass fiber and computer numerical control (CNC) machined aluminum parts. The mini UAV system could be hand launched and land on its body. The power system is composed of two electric motors mounted in the front of the main wing.

The avionics system is specifically engineered for Bayraktar Mini UAV. Its embedded real time software environment handles control, waypoint navigation, digital communication and functions for digital image transfer. The

autopilot supports autonomous flight, autonomous take-off and landing, half automatic control through joystick and manual control with a secure, digitally encrypted link.

The ground control station (GSC) system provides the management between the operator interface and the UAV. The operator interface, running on a Windows based PC system, supports the command, control and monitoring of the UAV in flight in real time (Figure 1.4). There is also an option for the flight to be visualized in a 3D environment such as Microsoft Flight Simulator. The system's graphical user interface (GUI) can been seen in Figure 1.4.

The current system is based on single UAV and GCS applications. The software architecture of the electronic systems within the Bayraktar Mini UAV system is based on the multi UAV systems applications, which is still in the development phase.

General characteristics

- **Wing Span:** 1.6 m
- **Length:** 1.2 m
- **Speed:** 27 - 60 knots
- **Fuel:** Lithium Polymer Battery
- **Propulsion:** Two Brushless Electric Motors
- **Weight:** 5 kg (Max.)
- **Payload:** 1.5 kg (Max.)
- **Flight Time:** 1 Hour (Min.)
- **Flight Envelope:** 3000 m (Max.)
- **Navigation:** GPS
- **Payload:** Digital Camera and IR Night Camera
- **Control Range:** 15 km LOS
- **Data Link:** Spread spectrum, Frequency Hopping Radio System 236 kbps in 902 - 928 MHz ISM band 20 km LOS) with Hardware Encryption

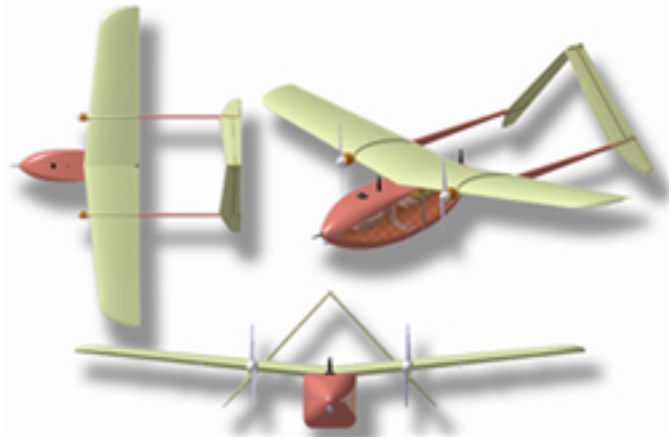


Figure 1.3: Bayraktar Mini UAV fixed-wing platform (Baykar Machine Inc.)

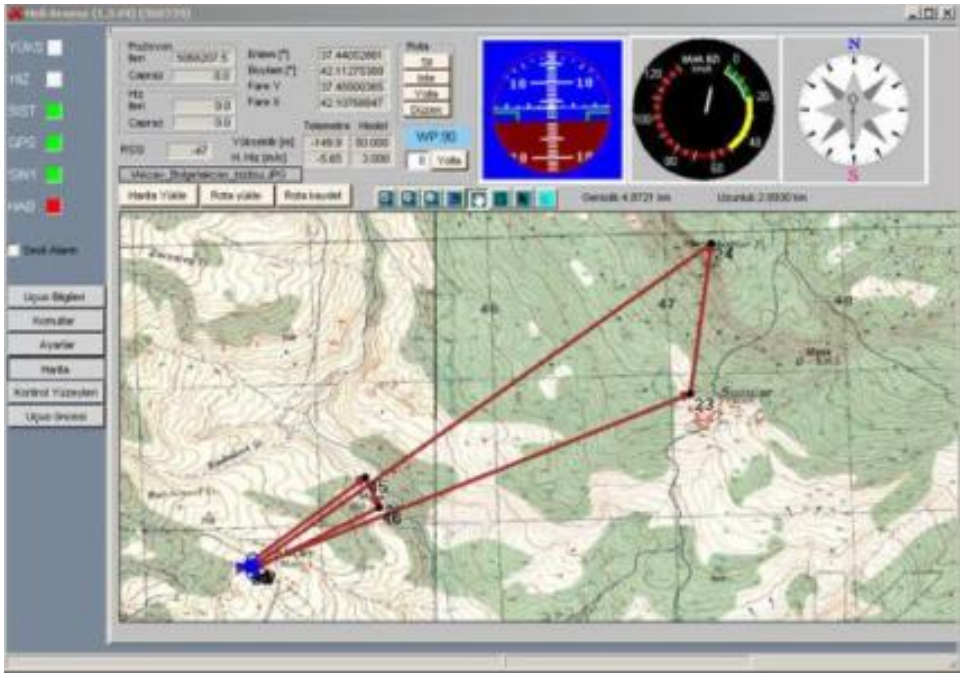


Figure 1.4: Bayraktar Mini UAV System Operator System (Baykar Machine Inc.)

1.4 Disposition of the thesis

This thesis has eight chapters including this introduction.

Chapter 2 gives background information on the GPS system and the development software alternatives.

In chapter 3 the GS system is outlined, the hardware and development software are selected, and a GS design is presented.

Chapter 4 is a short user guide to the developed GS software.

In chapter 5 the developed hardware, protocol definitions and developed software are presented.

Chapter 6 presents the achieved results.

In chapter 7 the obtained results are discussed. The hardware, software and system design choices are evaluated, and recommendations for further work are presented.

Chapter 8 concludes the thesis.

Appendix A is a DVD which contains the developed software, developed hardware documentation, references, videoes and this report's LaTeX source.

A number of the references used in this thesis where found on the Internet. These links were all available at the date of this thesis' completion. Most of them can also be found as pdf documents in Appendix A.

Chapter 2

Background

In this chapter background information on the GPS system and the development software alternatives are presented.

2.1 GPS

The Global Positioning System (GPS) is currently the only fully functional Global Navigation Satellite System (GNSS). Up until recently the system consisted of 24 satellites, with three spare satellites in case of failure. The satellites are distributed equally among six circular orbital planes (gps.gov 2007).

The United States Department of Defense (DoD) developed and implemented the satellite network as a military navigation system (pnt.gov 2007).

It is now open to the public, and enables a GPS receiver to determine its location, speed and direction. GPS has become a widely used aid to navigation worldwide, and a useful tool for map-making, land surveying, commerce, and scientific uses.

System segmentation

The current GPS consists of three major segments: the space segment (SS), a control segment (CSE), and a user segment (US) (www.aero.org 2007).

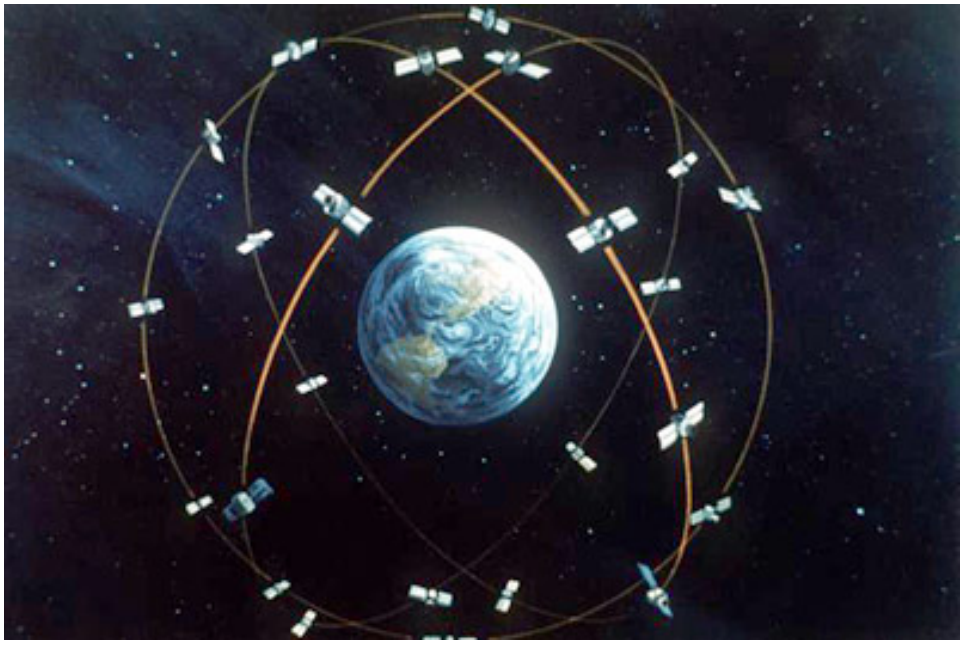


Figure 2.1: Artist's concept of the GPS satellite constellation (DoD)

Space segment

The SS is composed of the orbiting GPS satellites, also called space vehicles (SV). The SV's six orbital planes are centered on the Earth. They are tilted approximately 55° relative to Earth's equator, and are separated by 60° right ascension of the ascending node.

Orbiting at an altitude of approximately 20,200 kilometers, each SV makes two complete orbits each sidereal day, passing over the same location on Earth once each day. The orbits are arranged so that at least six satellites are always within line of sight from almost everywhere on Earth's surface.

Control segment

The CSE consists of six ground stations located around the world, which makes sure the satellites are working properly. The master control station at Schriever Air Force Base, near Colorado Springs, Colorado, runs the system. The ground stations contacts each SV regularly with a navigational update. These updates synchronize the atomic clocks on board the SVs to within one microsecond and adjust the ephemeris of each satellite's internal orbital model. The updates are created by a Kalman Filter which uses inputs

from the ground monitoring stations, space weather information, and various other inputs.

User segment

The user's GPS receiver is the US of the GPS system. In general, GPS receivers are composed of an antenna, tuned to the frequencies transmitted by the satellites, receiver-processors, and a highly-stable clock. Many also include a display for providing location and speed information to the user. A receiver is often described by its number of channels indicating how many SVs it can monitor simultaneously. Originally limited to four or five, this has increased to between twelve and twenty channels.

GPS receivers may include an input for differential correction of the position. Many GPS receivers can also relay position data to an external device like a PC using the NMEA 0183 protocol. NMEA 2000 is a newer and less widely adopted protocol. Both are proprietary and controlled by the US-based National Marine Electronics Association (NMEA). Other proprietary protocols exist as well, such as the SiRF and MTK protocols. The interfacing with other devices can usually be done using a serial connection, USB or Bluetooth.

Calculating position

A GPS receiver calculates its position by measuring the distance between itself and three or more GPS SVs. Measuring the time delay between transmission and reception of each GPS radio signal, gives the distance to each SV since the signal travels at a known speed. In order for the distance information to be of any use, the receiver also has to know where the SVs actually are. Therefore the GPS receiver stores an almanac that tells where every SV should be at any given time. The signals also carry information about the SVs location. By determining the position of, and distance to, at least three SVs, the receiver can compute its position using trilateration. Receivers typically do not have perfectly accurate clocks and therefore track one or more additional SVs to correct the receiver's clock error.

2.2 Differential GPS

The GPS system assumes the radio signals will make their way through the atmosphere at a consistent speed. The Earth's atmosphere slows the electromagnetic energy down somewhat, particularly as it goes through the ionosphere and troposphere. The delay varies depending on where on Earth the position is, which means it's difficult to accurately factor this into the distance calculations. Problems can also occur when radio signals bounce off large objects, such as skyscrapers, giving a receiver the impression that a satellite is farther away than it actually is. Differential GPS (DGPS) helps correct these errors. The basic idea is to use a stationary receiver station with a known location as a reference. Since the DGPS hardware at the station already knows its own position, it can easily calculate its receiver's inaccuracy. The station then broadcasts a radio signal to all DGPS-equipped receivers in the area, providing signal correction information for that area. This correction makes the DGPS receiver much more accurate than ordinary GPS receivers (Environmental Studies 2007).

Accuracy and error sources

Ranging errors are grouped into the six following classes (Wormley 2007):

- Ephemeris data - Errors in the transmitted location of the SV
- Satellite clock - Errors in the transmitted clock
- Ionosphere - Errors in the corrections of pseudorange caused by ionospheric effects
- Troposphere - Errors in the corrections of pseudorange caused by tropospheric effect
- Multipath - Errors caused by reflected signals entering the receiver antenna
- Receiver - Errors in the receiver's measurement of range caused by thermal noise, software accuracy, and inter-channel biases

Table 2.1 on the facing page shows how much these errors influences the accuracy. Typical normal accuracy for well-designed civil equipment under nominal operating conditions should be about 10 m horizontal and 13 m vertical for GPS and below 5 m using DGPS equipment.

Error source	GPS [m]	DGSP [m]
Ephemeris	2.1	0.0
Satellite clock	2.1	0.0
Ionospheric	4.0	0.4
Tropospheric	0.7	0.2
Multipath	1.4	1.4
Receiver	0.5	0.5
User eq. range errors (UERE), rms	5.3	1.6
Filtered UERE,rms	5.1	1.5
Vertical error	12.8	3.9
Horizontal error	10.2	3.1

Table 2.1: Standard GPS and DGPS receiver error model.

2.3 EGNOS

The European Geostationary Navigation Overlay Service (EGNOS) is a joint project of the European Space Agency (ESA), the European Commission (EC) and Eurocontrol, the European Organisation for the Safety of Air Navigation. Simplified this is a satellite supported DGPS system (Figure 2.2). The correction signals that improve the accuracy of the GPS receivers are transmitted by geostationary satellites instead of a stationary reference station on the ground (ESA 2007). According to specifications, position accuracy is improved to between 1 and 2 meters horizontal, and between 2 and 4 meters vertically (ESA 2005), which has been confirmed with practical measurements (Ventura-Traveset et al. 2005).

The system send the correctional signals at the same frequencies as the GPS. A modern 12 channel GPS receiver maximally receives signals from 10 satellites, giving the option to also receive EGNOS correctional signals. As the GPS the use of these correction data is free of charge (Environmental Studies 2007).

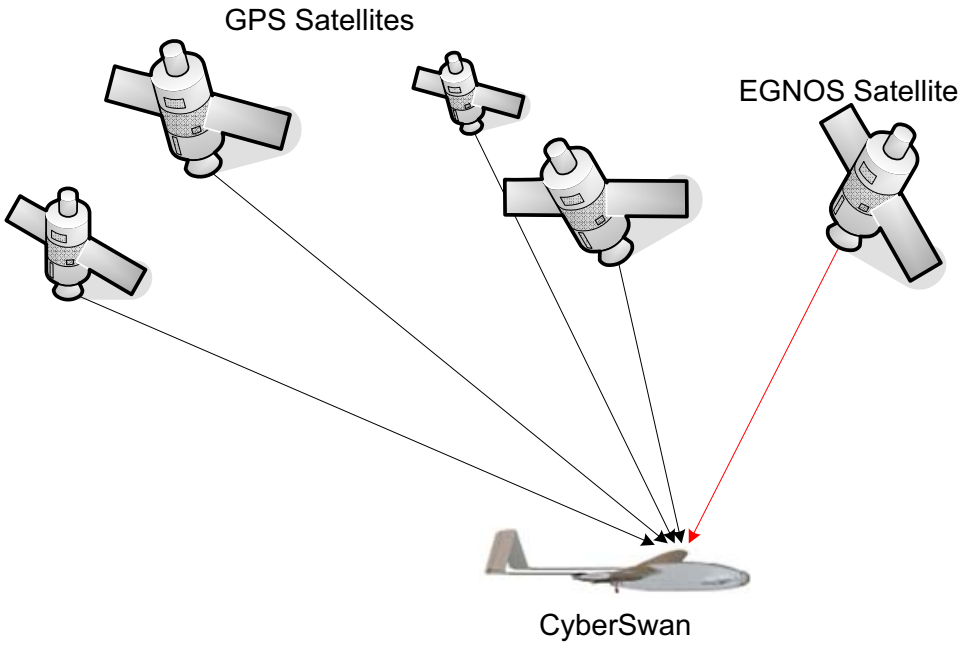


Figure 2.2: Positioning with GPS and EGNOS

2.4 Development Software

LabVIEW

Laboratory Virtual Instrumentation Engineering Workbench (LabVIEW) is a platform and development environment from National Instruments. LabVIEW is a visual programming language and uses a graphical language named "G". LabVIEW is commonly used for data acquisition, instrument control, and industrial automation on a variety of platforms including Microsoft Windows, UNIX, Linux, and Mac OS. The latest version of LabVIEW is version 8.20, released in honor of LabVIEW's 20th anniversary (National Instruments 2007).

The graphical programming language, G, is a dataflow language. Execution is determined by the structure of a graphical block diagram, the source code, on which the programmer connects different function-nodes by drawing wires. These wires propagate variables and any node can execute as soon as all its input data become available. Since this might be the case for multiple nodes simultaneously, G is inherently capable of parallel execution. Multi-

processing and multi-threading hardware is automatically exploited by the built-in scheduler, which multiplexes multiple OS threads over the nodes ready for execution.

LabVIEW programs/subroutines are called virtual instruments (VIs). Each VI has three components: a block diagram, a front panel and a connector pane. The latter may represent the VI as a subVI, with the shape of an icon, in block diagrams of calling VIs. Controls and indicators on the front panel allows input of data into or extract data from a running VI. The front panel can also serve as a interface for programming. Ergo, a VI can either be run as a stand-alone program, with the front panel serving as a user interface, or, inserted as a node onto the block diagram. The front panel then defines the inputs and outputs for the given node through the connector pane. This gives the possibility to easily test each VI before embeddeding it as a subroutine into a larger program.

The threshold for programming is low, and allows non-programmers to build programs by simply dragging virtual representations of the lab equipment with which they are already familiar. The LabVIEW programming environment, with the included examples and the documentation, makes it simpler to create small applications. This is a benefit on one side but there is also a certain danger of underestimating the expertise needed for good quality G programming. For complex algorithms or large-scale code it is important that the programmer possess an extensive knowledge of the special LabVIEW syntax and the topology of its memory management. The most advanced LabVIEW development systems offer the possibility of building stand-alone applications. Furthermore, it is possible to create distributed applications which communicate by a client/server scheme, and thus is easier to implement due to the inherently parallel nature of G-code.

One benefit of LabVIEW compred to other development environments is the extensive support for accessing instrumentation hardware. Drivers and abstraction layers for many different types of instruments and buses are included or are available for direct implementation. These present themselves as graphical nodes, subVIs. The abstraction layers offer standard software interfaces to communicate with hardware devices. The provided driver interfaces save program development time.

LabVIEW includes a compiler that produces native code for the CPU platform. The graphical code is translated into executable machine code by interpreting the syntax and by compilation. The LabVIEW syntax is strictly enforced during the editing process, and it continously gives feedback if the

code is executable or not. If it's not, error and warning feedback is provided. Debugging is also possible using execution highlightning, probing, using break points, and single-stepping through the block diagram. Using execution highlighting, it is possible to view an animation of the execution of the block diagram. The animation shows the movement of data on the block diagram from one node to another using bubbles that move along the wires. This can also be used in conjunction with single-stepping to see how data move from node to node through a VI. The code is compiled into executable machine code when requested to run or upon saving. When saving the executable and the source code are merged into a single file. The executable runs with the help of the LabVIEW run-time engine, which contains some precompiled code to perform common tasks that are defined by the G language. The run-time environment makes the code portable across platforms. Generally, LabVIEW code can be slower than equivalent compiled C-code, although the differences often lie more with program optimization than inherent execution speed.

LabVIEW provides libraries with a large number of functions for data acquisition, signal generation and processing, communication, mathematics, analysis, control and vision to name a few. This, along with numerous graphical interface elements are provided in LabVIEW's package options.

The LabVIEW Professional Development System allows creating standalone executables and the resultant executable can be distributed an unlimited number of times. The run-time engine and its libraries can be provided freely along with the executable.

A benefit of the LabVIEW environment is the platform independent nature of the G-code, which is, with the exception of a few platform-specific functions, portable between the different LabVIEW systems for different operating systems.

C++ with Qt toolkit

C++

C++ is a general-purpose, high-level programming language with low-level facilities. It is a statically typed free-form multi-paradigm language, supporting procedural programming, data abstraction, object-oriented programming, generic programming and run-time type identification. Bjarne Stroustrup developed C++ in the early 1980s at Bell Laboratories as an enhance-

ment to the C programming language. Enhancements started with the addition of classes, followed by, among other features, virtual functions, operator overloading, multiple inheritance, templates, and exception handling (Deitel & Deitel 2001).

Compared to the C language, C++ introduced extra features, including declarations as statements, function-like casts, new/delete, bool, reference types, inline functions, default arguments, function overloading, namespaces, classes (including all class-related features such as inheritance, member functions, virtual functions, abstract classes, and constructors), operator overloading, templates, the :: operator, exception handling, and runtime type identification.

Qt toolkit

Qt is a cross-platform application development framework, produced by the Norwegian company Trolltech, widely used for the development of GUI programs. Since the release of Qt 4, it is also used for developing non-GUI programs such as console tools and servers. Examples of applications where Qt is used are the KDE desktop environment, the web browser Opera, Google Earth and Skype (Trolltech 2007*a*).

Qt includes a C++ class library and tools for cross-platform development and internationalization (Trolltech 2007*b*).

The Qt Class Library is a library, currently with more than 400 classes, which encapsulates all infrastructure needed for end-to-end application development. The Qt API includes a mature object model, a rich set of collection classes, and functionality for GUI programming, layout, database programming, networking, XML, internationalization, OpenGL integration and much more.

Qt Designer is a powerful GUI layout and forms builder, enabling rapid development of high-performance user interfaces with native look and feel across all supported platforms.

Qt Linguist is a set of tools designed to smooth the internationalization workflow. Using Qt Linguist, development teams can outsource the translation of applications to non-technical translators, increasing accuracy and greatly speeding the localization process.

Qt Assistant is a fully customizable, redistributable help file/documentation browser that can be shipped with Qt-based applications. With Qt Assistant,

development teams significantly speed the documentation process.

Qt uses standard C++, but extends the language by providing an additional pre-processor that generates the C++ code which is necessary to implement Qt's extensions. Qt is also available on Java, using the cross-platform, rich client application development framework Qt Jambi.

Complete abstraction of the GUI

Qt uses its own paint engine and controls. This makes the work of porting to other platforms easier because very few classes in Qt depended on the target platform. Qt use the native styles API of the different platforms to draw the Qt controls.

Meta-Object compiler

The Meta-Object compiler also called the moc, generates meta-object information for QObject subclasses. This information is used by Qt to provide programming features not available in C++: the signal and slot mechanism, the run-time type information and the dynamic property system. The signal and slot mechanism is Qt's solution to inter-object communication.

Dual licensing model

Through Trolltech's Dual licensing model, Qt delivers all of the advantages of open source in a commercially-supported, proven framework. Open source benefits include an active open source developer community contributes to the ongoing development of Qt while complete code transparency allows Qt developers to "see under the hood", customizing and extending Qt to meet their unique needs.

Chapter 3

System Design

In this chapter the overall design of the system is outlined. The principal system design for implementation is specified, the hardware options and development software are evaluated and chosen. In the end the GS design is presented.

3.1 System Outline

Figure 3.1 shows the general system structure. Both the GS and the CS needs a positioning device, and a wireless communication link is needed between the GS and the CS in order to transmit sensor and navigation data. A visual link and a ground ranging device are to be gathered and mounted on the CS.

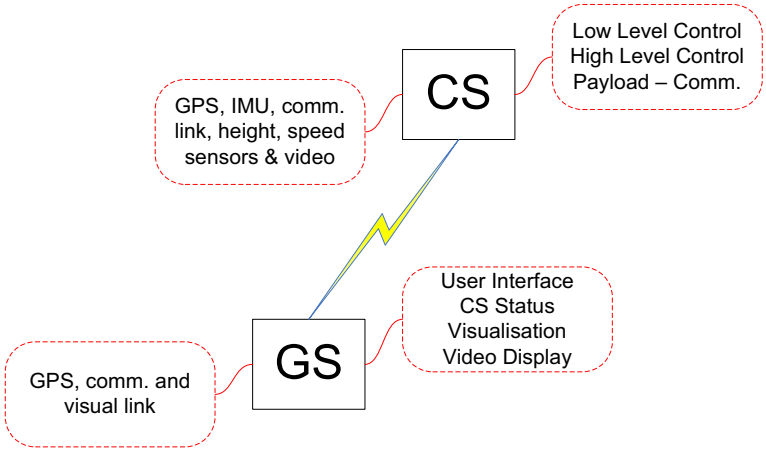


Figure 3.1: Overall system overview.

3.2 Hardware Selection

Positioning

The position of the GS is determined by the use of a GPS reciever. The hardware system in the CS will use a GlobalSat EM-411 GPS module (Bjørntvedt 2006). The module provides a position with accuracy below 10 m, and within 5 meters with WAAS enabled. The EM-411 supports NMEA 0183 and SiRF output protocol. To ensure minimal difference in error, numerical error particularly, this module was also chosen for the GS. This would give the GS and the CS approximately the same error in the same area.

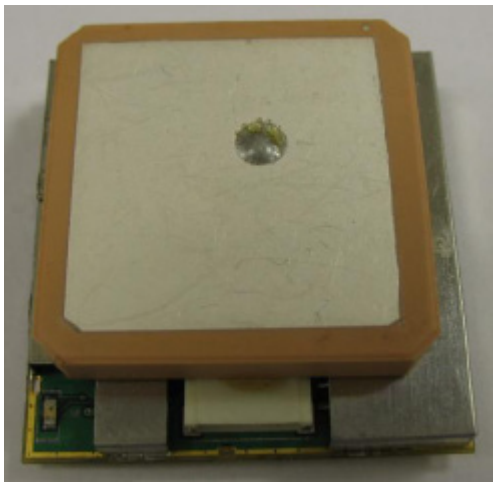


Figure 3.2: GlobalSat EM-411 GPS Receiver.

GPS chipset	SiRFstarIII
Position accuracy	10 m, 2D RMS
	5m (WAAS enabled), 2D RMS
Velocity accuracy	0.1 ms
Time accuracy	1 μs synchronized to GPS time
Datum	Default: WGS-84
Supply voltage	4.5 - 6.5 V
Power consumption	60 mA
Electical output	TTL level: 0 - 2.85 V
Baud rate	Default: 4800 bps
Output messages	NMEA 0183 Protocol: GGA, GSA,
	GSV, RMC, VTG and GLL

Table 3.1: GlobalSat EM-411 GPS engine board specifications.

NMEA 0183

NMEA 0183 is a standard that uses a simple ASCII, serial communications protocol that defines how data is transmitted in a sentence from one talker to one or more listeners. It also defines the contents of each sentence (message) type so that all listeners can parse messages accurately. The first two letters define the strings source, and the three following letters defines what message it is. $GPGGA tells that the source, GP, is the Global Position System, and, GGA, is Global Positioning System Fix Data.

To get the wanted information from the GPS receiver, position, altitude, speed and course, two sentences has to be read; $GPGGA and $GPRMC. Table 3.2 and Table 3.4 holds detailed information about the contents of the messages.

$GPGGA-Global Positioning System Fixed Data

$GPGGA,124637.000,6325.0840,N,01024.1304,E,1,06,1.8,56.1,M,41.5,M,,0000*6B

Name	Example	Description/Unit
Message ID	$GPGGA	GGA protocol header
UTC Time	124637.000	[hhmmss.sss]
Latitude	6325.0840	[ddmm.mmmm]
N/S Indicator	N	N=north or S=south
Longitude	01024.1304	[dddmm.mmmm]
E/W	E	E=east or W=west
Pos. Fix Indicator	1	See Table 3.3
Satellites Used	06	Range 0 to 12
HDOP	1.8	Horizontal Dilution of Precision
MSL Altitude	56.1	[m]
Units	M	Unit for Altitude
Geoid Separation	41.5	[m]
Units	M	Unit for Geoid Separation
Age of Diff. Corr.		Null fields when DGPS is not used [s]
Diff. Ref. Station ID	0000	[0000-1023]
Checksum	*6B	
<CR><LF>		End of message term. (Hex: 0D 0A)

Table 3.2: GGA Data Format.

Value	Description
0	Fix not available or invalid
1	GPS SPS Mode, fix valid
2	Differential GPS, SPS Mode, fix valid
3	GPS PPS Mode, fix valid

Table 3.3: Position Fix Indicator.

$GPRMC - Recommended Minimum Specific GNSS Data

$GPRMC,124637.000,A,6325.0840,N,01024.1304,E,0.16,46.98,150607,,*38

Name	Example	Description/Unit
Message ID	$GPRMC	RMC protocol header
UTC Time	124637.000	[hhmmss.sss]
Status	A	A=data valid or V=data not valid
Latitude	6325.0840	[ddmm.mmmm]
N/S Indicator	N	N=north or S=south
Longitude	01024.1304	[dddmm.mmmm]
E/W	E	E=east or W=west
Speed Over Ground	0.16	[knots]
Course Over Ground	46.98	Relative to true north [deg.]
Date	150607	[ddmmyy]
Magnetic Variation		E=east or W=west [deg.]
Checksum	*38	
<CR><LF>		End of message term. (Hex: 0D 0A)

Table 3.4: RMC Data Format.

Checksum is the absolute value calculated by using exclusive-OR on the 8 data bits of each character in the sentence, between, but excluding $ and *. The hexadecimal value of the most significant and least significant 4 bits of the result are then converted to two ASCII characters (0-9,A-F) for transmission. The most significant character is transmitted first.

DGPS reference station

By using the same GPS receiver module in the GS as in the CS, both units will have approximate the same errors when they are in the same area. This gives the possibility of using the GS as an terrestrial reference station. The GS's GPS coordinates are interpolated for some time to get a position which is treated as its absolute position. Any change in its GPS reading compared to this position is concidered to be an error. This error is sent to the CS as a correction signal for its position, improving it accordingly. This method is the Differential GPS method (Section 2.2 on page 12). The accuracy obtained by this method ranges between 1 - 3 meters, whereby the accuracy depends on the distance to the correctional data transmitter and the signal quality (Environmental Studies 2007). If in addition the GPS receiver supports reception of EGNOS correction data, the accuracy should be expected to be even better (Section 2.3 on page 13).

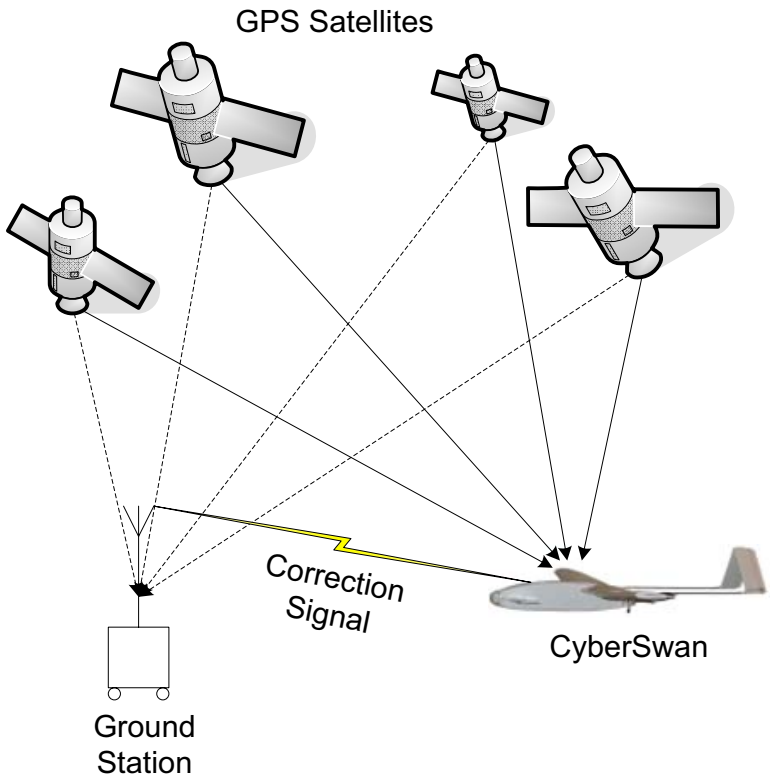


Figure 3.3: Ground Station working as a reference station for the CyberSwan

Communication

In Bjørntvedt (2006) the Radiocraft RC1240 radio frequency (RF) module, Figure 3.4, was chosen as the communication link to monitor the CS's state during flight. The module's range, price and simplicity of use where taken into account during the selection. The RC1240 (433.050-434.790 MHz) has a specified range of 2-4 km with 10 mW output power. This is though strongly dependent on which antenna is used (Evjen 2006), and a more realistic range with a quarter-wave antenna is 1-2 km. In pursuit for even longer range in communication an alternative was also concidered. The Radiocrafts RC1280HP (869.400 - 869.650 MHz) is a RF module with a high power amplifier offering up to 500 mW output power. When used with a quarter-wave antenna a line-of-sight range of 5-6 km can be achieved Evjen (2006). This though comes at the cost of a conciderable higher current consumption. 600 mA for the RC1280HP versus 26 mA for the RC1240 module. Both modules provides a data rate of 4800 bps. The research for a diffent solution, supporting a higher data rate, low current consumption and long enough range did not yield any result. The communication device used in the Bayraktar Mini UAV System (Section 1.3 on page 4) provides a data rate of 236 kbps. Developed in-house by Baykar Machine Inc this was not available. The communication link should have a range long enough to be able to monitor the plane at all stages of the flight. In this stage of the project it is not planned to fly the plane in autonomus mode longer that visual range. Then the plane has to be operated in an open area, and at 1 km the plane will be just a dot in the sky. 1-2 km will be a sufficient communication range for this project, and considering the current consumption, the Radiocrafts RC1240 RF-module was chosen as the communication link. Table 3.5 contains the RC1240's specifications. After the selection an option was found. The Aerobotics Research Group at Monash University (2007) uses MaxStream modems supporting higher throughput. Due to the stage the project was in, at the time of discovery, this was not pursued further.

Frequency band	433.05 - 434.79 MHz
Data rate	4800 bps
Max ouput power	8 dBm
Supply voltage	2.8 - 5.5 V
Current consumption, RX	20.2 mA
Current consumption, TX	26 mA

Table 3.5: Radiocrafts RC1240 specifications.

Figure 3.4: Radiocrafts RC1240 RF module.

Visual link

One of the possible areas of application for this system is video surveillance and still photography. Three options are concidered:

- Digital photograpy
- Digital video
- Analog video

The available digital communication link has a low data transfer rate. The chosen link, have a capability of 4800 bps. Regarding the limited data rate, range and output power there are limited possibilites for additional transfer of digital images and video on the chosen RF-link. No communication links are currently available to make this possible. A separate communication device for the transfer of visual information is therefore needed. With an analog video link with output power in the range 0.1 - 0.5 W, the possible range, dependent on modulation, will be in the area 100 - 1000 m scope. The video quality will decrease with the distance between the camera's transmitter and reciever. Although, compared to a digital option, which will either have a signal, or not. The analoge option will suffer signal degrated images, but will still give an image. Therefore the analog video option is taken.

Hi Cam Pro X2 system

The Australian company Hi Cam develop video transmission for airial use. Their Pro X2 system is a video transmitter which can be connected to any video source. The example videos from their web page shows video of great quality without any disturbances. These example videos are enclosed in Appendix A. The Pro X2 it could be hidden in one of the CS's wings, and the camera mounted at any other spot on the CS, linking them together with a cable. It would be an advantage to place the transimitter away from the body, limiting it's noise influence on the main computer system. The sysems range is stated to be 667 m, superior to other systems performance. These advantages made the Pro X2 system to be ideal choice. Unfortunately the system was out of stock. Hi Cam is currently working on an updated version with a molded casing. The new version will have an external microphone option, but otherwise have the same features as the old version. Tables 3.6 to 3.8 contains the Pro X2 system specifications (Hi Cam 2007). The new version was not available in time to be used in this project, so a different camera system had to be used.

LUDA Minikameran

Researching for other cameras to compete with the Pro X2 system's quality and range proved to be a difficult task. The research yielded no other system developed for airial use. As size was an important property the choices were limited to simple mass-produced camera systems ment for home security surveillance on ground level. The chosen camera, Luda Minikameran, has a built in transmitter with a stated range of 100 m. Figure 3.6 on page 30 shows the camera and AV-receiver, and Table 3.9 on page 30 contains the camera's specifications. In order to improve the range an directional antenna was purchased. The antenna provides a signal amplification of 19 dBi which will increase the systems range substantially.

Figure 3.5: Pro X2 transmitter and camera.

Imaging System	1/4 CCD" (colour)
Available TV Formats	PAL or NTSC
CCD Resolution	380 lines
Minimum Illumination	1.0 Lux (F2.0)
Supply Voltage	4.0 - 5.6 V
Current Consumption	160mA (5 V)

Table 3.6: Impact resistant 380 line camera specifications.

RF Output Power	200 mW
Channel Frequencies	2.410, 2.430, 2.450 and 2.470 GHz
Modulation Type	FM
Video Input	1 Volt p/p 75 Ω composite
Audio	Built-in microphone
Voltage Supply	4.2 - 5.6 V
Current Consumption	250 mA (5 V)
Detachable Antenna	Omni-directional (hinged version optional)
On-board Switches	Dip switches for setting channel
Range	∼2000' (∼667 m) (dep. on local conditions)

Table 3.7: Pro X2 transmitter specifications.

Antenna Connector	SMA, 50 Ohm
Video Output	1 Volt p/p 75 Ω composite
Audio Output	Line Level
Channels	Modulation: same as transmitter
Sensitivity	-85 dBm
Voltage Supply	12 V
Current Consumption	300 mA (approx)
Detachable Antenna	Patch, 8 dBi

Table 3.8: Pro X2 AV receiver specifications.

Figure 3.6: Luda Elektronik's wireless Minikameran camera with AV receiver.

Range	∼100 m (dep. on local conditions)
Imaging System	1/3" CMOS (colour)
TV Format	PAL
CMOS Resolution	382 lines (628x582 pixels)
Minimum Illumination	1.0 Lux (F2.0)
Supply Voltage	8 - 9 V
Current Consumption	80 mA
Output Power	10 mW
Channel Frequencies	2.414, 2.432, 2.450 and 2.468 GHz
Video Input	1 Volt p/p 75 Ω composite
Audio	Built-in microphone
On-board Switches	Dip switches for setting channel

Table 3.9: Luda camera specifications.

Ultrasound ranging device

To enable the CS's to perform an autonomous landing a good altitude measurement is needed. The GPS provides an altitude measurement once a second. This might be good enough if the landing area is completely flat, which is experieced to be an unreasonable concideration. Therefore a redundant altitude measurement is concidered to be necessary during the landing phase. For this use the MaxSonar-EZ1, Figure 3.7, from MaxBotix was chosen. It supports a measurement from 0 - 6.45 m makes it a suitable sensor. Table 3.10 contains the sonar's specifications. Also, it's small size supports mounting at the bottom the CS's fuselage, measuring the distance to the ground.

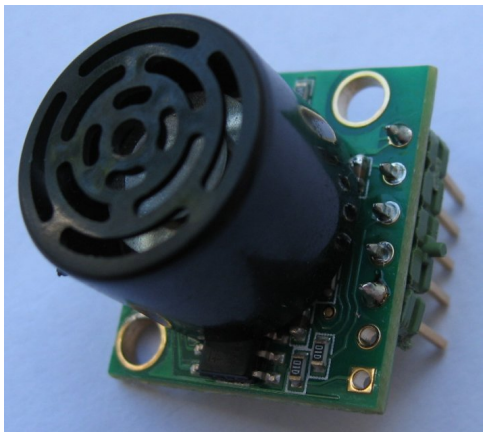


Figure 3.7: LV-MaxSonar-EZ1 used for height measurement.

Power Supply	2.5 - 5.5 V
Current Consumption	Typical: 2 mA
Range	0 - 6.45 m
Resolution	2.54 cm
Ouput Format	Serial (9600 baud), Analog and PWM
Operation Frequency	42 kHz

Table 3.10: MaxBotix LV-MaxSonar-EZ1 specifications.

3.3 Ground Station Design

Hardware setup

With the hardware choices made, the physical structure of the system is as illustrated in Figure 3.8. A portable computer is the core of the system, running the GS software. The antenna system, with the RF antenna on top, and a grid antenna to receive the video feed, is mounted on a portable tripod. The electronics draws its power from a 12 V car battery or similar, regulated individually to their specified voltage. The video is fed from the audio-visual (AV) receiver to an individual video display or into the laptop through a usb media adapter or similar.

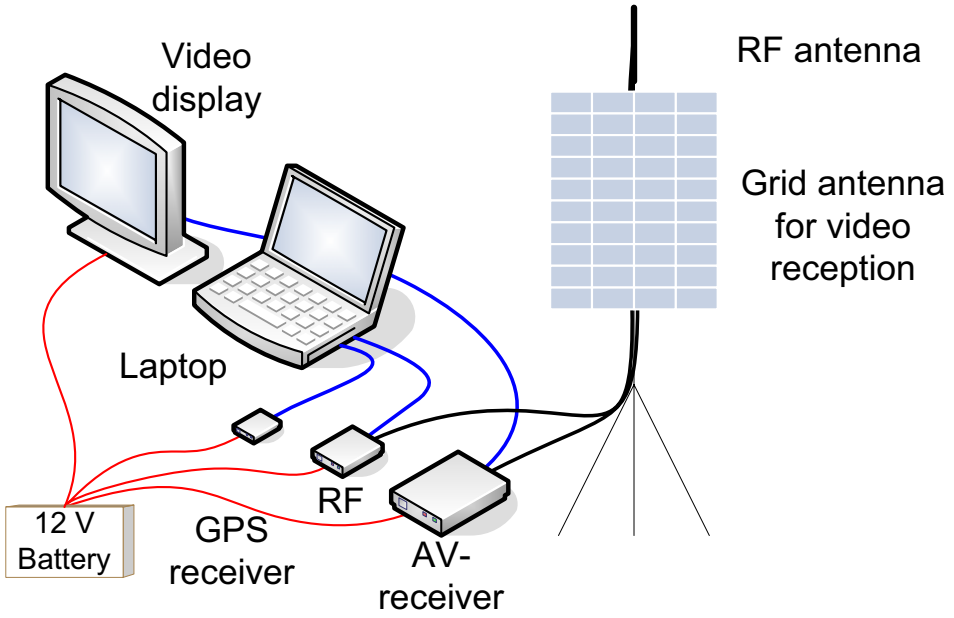


Figure 3.8: Illustration of Ground Station hardware setup.

Development software

C++ with the Qt toolkit gives the possibility to develop the ground station software to satisfy own specifications. This would include: low level serial driver development, GPS message parsing, communication protocol and GUI development, to name a few. Still, freedom of choice is not always for the better, meaning that this would be a big task for one person. In LabVIEW all these properties are already available. Also, LabVIEW is commonly used for control purposes. It is concidered that using LabVIEW (National Instruments 2007) would increase the possibility to create a working product, and it was therefore chosen to be used as the development software.

Ground station software design

The software should have the following properties:

- Message reception and transmission
- Presentation of received CyberSwan data
  - Position (GPS)
  - Physical orientation (Inertial Measurement Unit (IMU)
  - Actuator (Servo/Receiver)
  - Ultrasound and pressure sensor
- Position correction
- Navigation

Message reception and transmission

Communication is serial through RS-232 interface. Messages are chosen to be coded the same way as the messages received from the GPS receiver, the NMEA 0183 protocol. This gives only a message identifier and parser, no coding or decoding is needed, exept checksum generation and control. This also gives the advantage to receive and transmit messages from a terminal program, confirming correctness directly in the development phase.

Data presentation

All received data can be presented directly using LabVIEW. For the presentation of the GS and CS position it is of interest to visualise this on a map. Several professional chart plotter products are available but they are often expensive. A cheap alternative is to use Google Earth Pro which supports plotting. This would have been a good option, but this was not relevant as the current maps are not detailed enough. It is also of interest to keep everything limited to one program simplifying the flow of data. Therefore it was chosen to develop a plotter using LabVIEW.

Position correction

The system is to support the CS with correctional position data. This correction data is to be calculated and transmitted to the CS once a second through the communication device.

Navigation

To enable the CS to fly a preprogrammed route it is of interest to plan this route using the GS. This is to be implemented on the plotter with limited waypoints to prove it's possible.

Chapter 4

Ground Station User Guide

This chapter presents a short guide of normal use of the GS GUI, explaining its properties and possiblities, and its interaction with the CS. A video was also made, and is available in Appendix A.

4.1 Starting the GS

When starting the GS the transceiver normally would be the active panel. This is, in addition to the GPS panel, the panels that needs to be interacted with before the GS can be engaged. Figure 4.1a and Figure 4.1b on page 37 shows the respective panels. As the Figure's information display message indicates, *GS is not ready- calculate GS position in GPS panel, then engage!*, one have to chose the GPS panel to calculate the GS position. Also, the status indicator in not lit, indicating that the GS is not ready to be started.

The GPS panel has three displays and one button. The indicator on the right is showing the current position of the connected GPS receiver. The indicator on the left is for showing the calculated static GS position, and the indicator in the middle will show the difference between the two other indicators, the position error. Positive values will define a position North and East of the GS position. A green light will indicate that the connected GPS has a valid position, and the position should be refreshed once a second. Pressing the *Calculate GS Fix* button will start the collection of position data. When the button is released it will calculate the GS position. The position error display should now show close to zero error, defining the correction signal

to be sent to the CS.

On the transceiver panel, the *Ready*-light should now be lit and the information display should indicate, *GS ready - push Engage button to start GS!*. Doing this engages the plotter, starts the IMU and servos panels, and sends the GS position to the CS. The plotter will start with its default map chosen, plotting the CS position on the map, with a small red square, provided it is within the map's bounds.

4.2 Operation

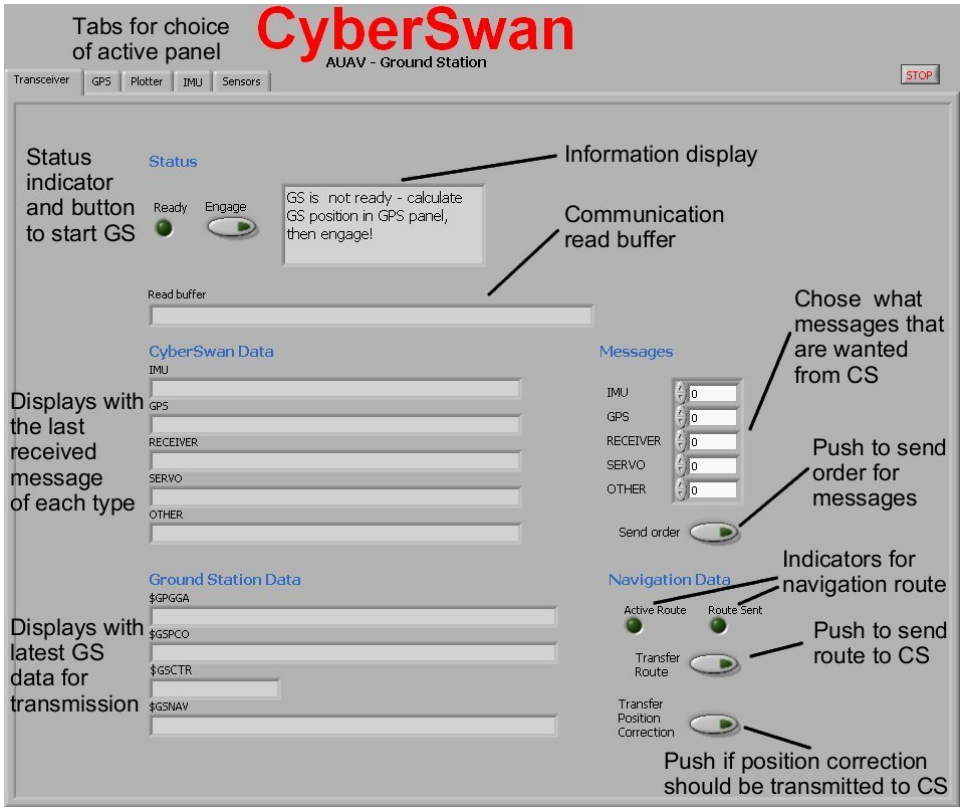
In the transceiver panel, the lower left section should now show the $GPGGA string, corresponding to the GS's position. Also, the $GSPCO string is displayed, changing once a second, corresponding to the position error display in the GPS panel.

Position correction

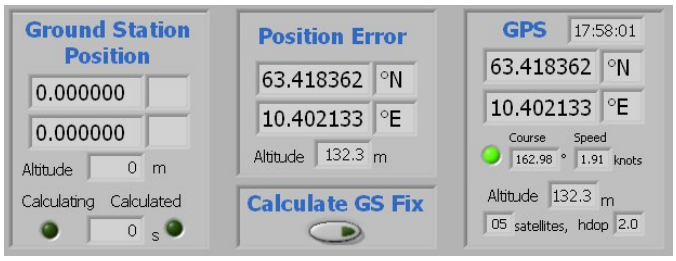
To transfer the position correction signal to the CS, press the *Transfer Position Correction* button, positioned in the lower right corner of the transceiver panel, once. As long as the button is active, the correction message will be sent to the CS once a second.

Ordering messages

To the right, in the middle of the transceiver panel, are five numeric controls arranged in a column. Here one can decide what messages to receive from the CS within one second, defined by the categories on the left of the column. How many messages possible to receive is decided by the bandwidth of the serial communication link between the GS and the CS. By using the RF card developed in this project, this is limited to four messages. Ordering more than four messages will not be taken into account, as messages will be ordered with priority counting from the top of the column. After chosing the number of messages wanted, using the small dials, or entered directly, pressing the *Send order* button will show the $GSCTR string in the lower left corner and transmits it to the CS. Received messages will be displayed in the indicators and its correspondent panel: plotter, IMU and sensors.



(a) Transceiver panel with description details.



(b) GPS receiver panel.

Figure 4.1: Ground Station transceiver and GPS panel.

4.3 Plotting

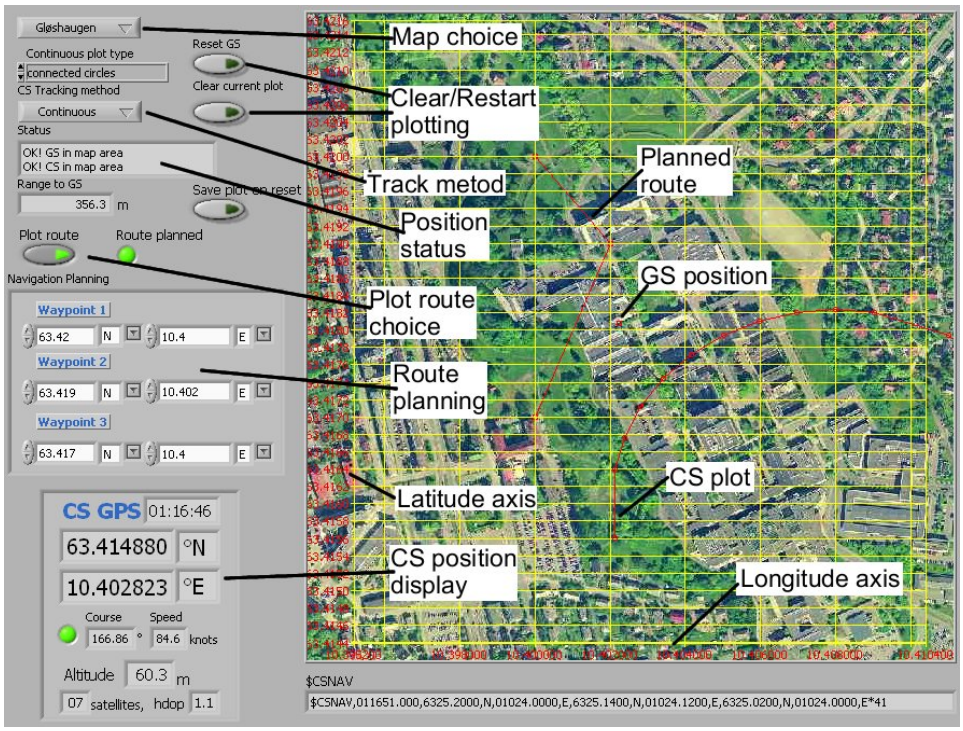


Figure 4.2: GS plotter display with description details.

Map

When receiving the $CSGPS message, the plotter is fully functional. Figure 4.2 shows the plotter panel, with the CS's position display in the lower left corner. The top left control is the choice of what map to draw. To change the map, th the *Reset GS* button has to be pressed, in order to draw the new map. This also regards the GS position, if it has been recalculated.

Tracking

There are two choices regarding CS tracking. Either continously, or just the current position, chosen with the *CS Tracking method* control. The status window will continously tell if the GS and the CS is within the chosen map's

area. Below the status window the current range to the CS is presented. There is also a possibilty to save the current CS plot by pushing the *Save plot on reset* button and then the *Reset GS* button. This brings up a dialog to save the CS plot to a text file. This can then be examined in a text editor.

Navigation

The navigation route planning concists of three waypoints. Entering the waypoints, and pressing the *Plot route* button will draw up the planned route. The *Route planned* light will now be lit, as well as the *Active route* light on the transceiver panel. The route can now be transferred to the CS, but this has to happen while the GS is disengaged. Pushing the *Engage* button once, disengages the GS. Activating the *Transfer route* button will send the message. Now an acknowledge message is expected from the CS within a short time. If its received, the *Route sent* light will be lit, if not, the *Transfer route* button pops out, leaving the *Route sent* light unlit.

4.4 CS orientation

When receiving the $CSIMU message, the received data updates the indicators in the IMU panel, as shown in Figure 4.3. The source of the data is the CS's IMU giving information about its orientation in space. The roll, pitch and yaw angles are used to rotate the 3D model, showing the CS's orientation. The model can also be rotated using the mouse, and the display is resetted to default by pressing the *Reset display* button.

4.5 Sensors

The last panel contains information contained in the $CSSRV, $CSREC and $CSOTH messages. The received messages are parsed and the information is displayed in its corresponding indicator, as shown in Figure 4.4.

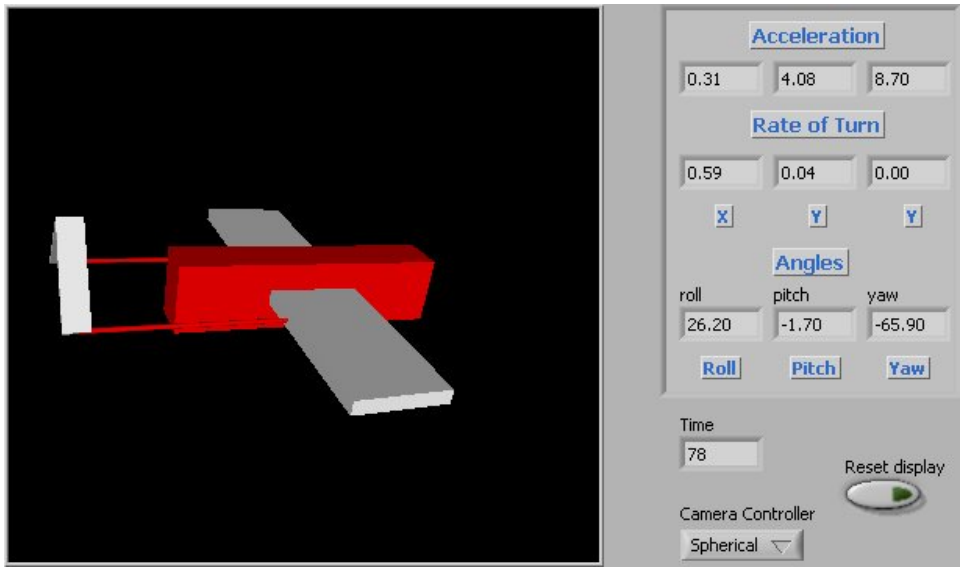


Figure 4.3: CS 3D orientation.

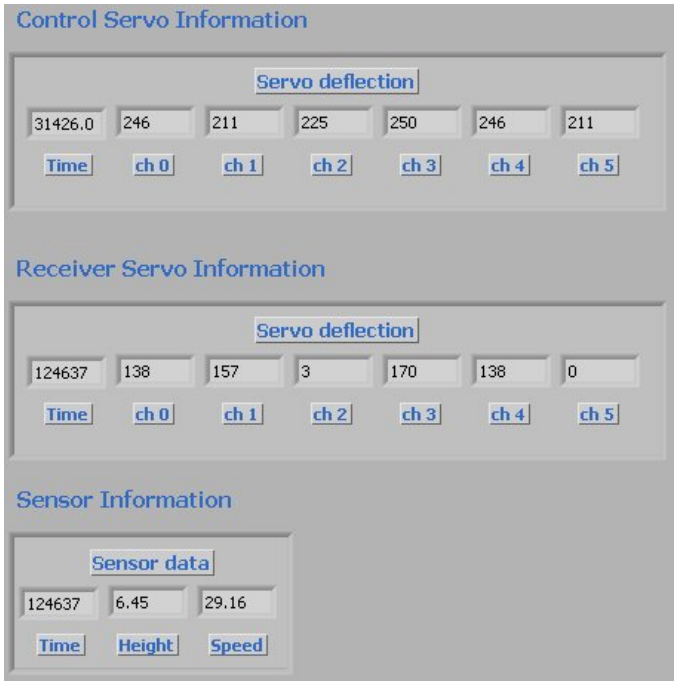


Figure 4.4: Sensors panel.

Chapter 5

Development

This chapter describes the development of the hardware and software produced according to the system design in Chapter 3. It also describes the protocols used for communication.

5.1 Hardware development

The EAGLE Layout Editor (CadSoft Online 2006) was used for circuit board layout, and the developed circuits were produced in the Department of Engineering Cybernetics' workshop.

GPS

In order to use the EM-411 GPS module a power supply and a suitable communication interface is needed. Since it has to be connected to a computer, a RS-232 interface is preferable. The data sheet indicated an electical interface with 5 V power supply and a transistor-transistor logic (TTL) level output of 0 - 2.85 V. The module was tested with a normal RS-232 transceiver, though thought for TTL devices with a 0 - 5 V output level. Still, this worked fine, since 2.85 V is above the transceivers "high"-level threshold. This gave a circuit with a block schematic representation as shown in Figure 5.1. Since there were no special demands regarding the circuit's size, standard in stock components were used. The result can be seen in Figure 5.2. Spacers were mounted in order to fit the GPS module in a stable position.

Using a terminal program the module was configured to only output the $GPGGA and $GPRMC sentences as they contain the information needed. The serial interface was changed also to 9600 baud. The protocol for changing the settings were found in the user manual enclosed in Appendix A.

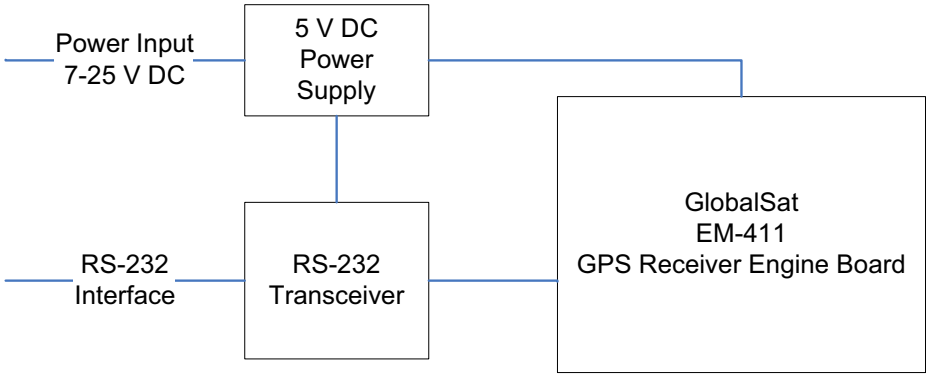


Figure 5.1: Block diagram of GPS Interface.

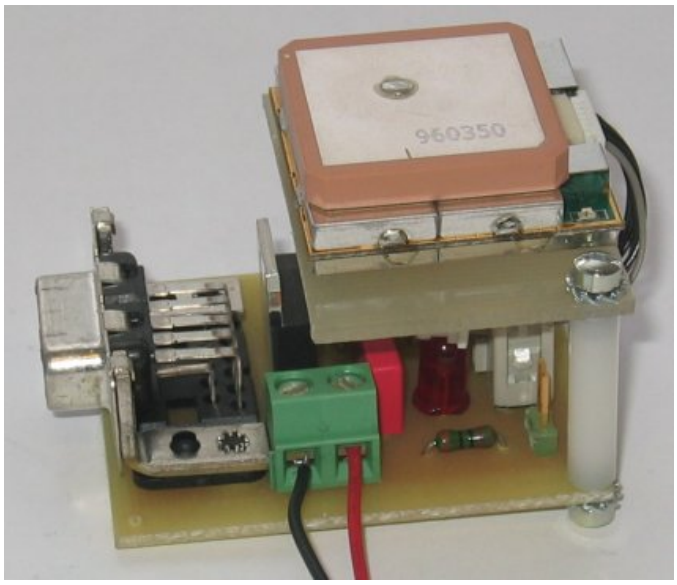


Figure 5.2: EM-411 GPS Module with RS-232 Interface.

RF board

Figure 5.3 shows the block schematic representation of the RF board produced for the CS. A low drop-out (LDO) voltage regulator is used to regulate the inputted power from a range 4.85 - 15 V DC, down to 3.3 V DC. The regulator supplies both the RS-232 transceiver and the RF module with power. The MAX3232 RS-232 transceiver is used to make it possible to communicate with the RF module through the RS-232 protocol. The Radiocrafts RC1240 module is used for RF communication between the GC and CS.

The circuit board is designed to keep the size and thickness small to enable an easy fit inside the CS. Figure 5.4 shows the produced circuit board. All devices used on the board are of surface mount device (SMD) type. The exception are the connectors, where the right-angle shrouded board-mount headers are chosen. The antenna connector is of female SMA-type. It is mounted perpendicular to the ground plane, leaving the SMA connectors transmission line contact point parallel to the circuit board's transmission line. This was done first of all due to production reasons. The antenna transmission line has to be routed on the top plane of the circuit board since the ground plane is on the bottom plane. To be able to solder the transmission line to the antenna connector, a connector for chassis mounting was chosen. This allows the connector to be soldered directly to the transmission line. As a result of mounting it in this way is that the antenna can easily be fitted inside the CS's wing length, without any chance of bulging out of its wrapping. The connector's screen is fixed to the ground plane with a small bar of aluminium, using screws through the circuit board. The transmission line's impedance is matched to 50 Ω. The impedance is defined by the width of the transmission line, and the thickness and dielectric constant of the circuit board's dielectricum. Figure 5.4 shows a magnified view of the transmission line in a circular box. The width of the line was calculated to 107.9 mil (2.74 mm) using the Microstripline Analysis & Design calculator provided by Green Bay Professional Packet Radio (2007), giving close to 50 Ω impedance.

The module is connected to the PC at a baud rate of 19200. It then sends the buffered data if its data buffer is full (128 byte), a unique end character is sent, or a modem time limit is reached. It was chosen to transmit data using a unique end charcter, line feed. The RF module was configured accordingly using a terminal program and commands found in Radiocrafts' RS232 user manual enclosed in Appendix A.

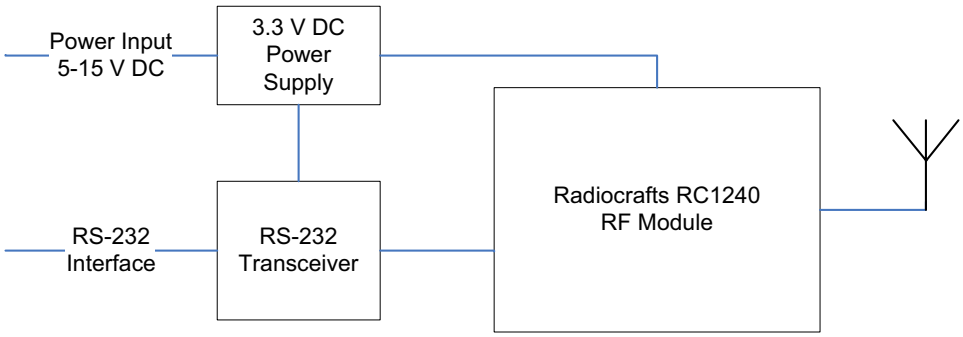


Figure 5.3: Block diagram of CS RF board.

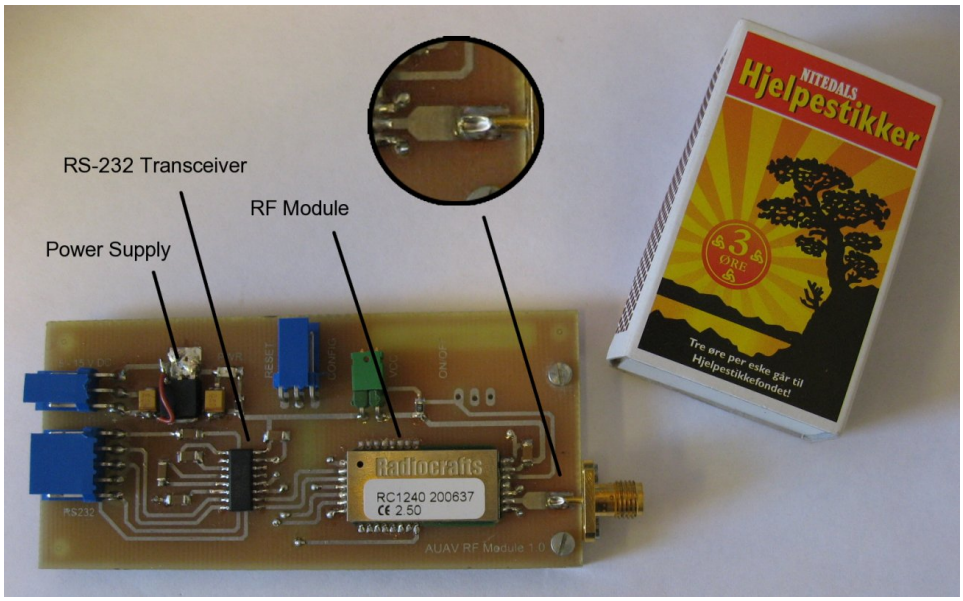


Figure 5.4: CS RF board with Radiocrafts RC1240 Module.

Camera

The Luda Minikameran was fixed to back wing on the CS, as shown in Figure 5.5. A video from a test flight is available in Appendix A.

Figure 5.5: Minikameran mounted on the CS's back wing.

5.2 Communication Protocols

It was necessary to define own sentences in order to be able to transfer messages with the wanted contents. The same structure as the NMEA 0183 protocol was chosen, and the following messages where specified in cooperation with Bjørntvedt (2007): $CSGPS, $CSIMU, $CSSRV, $CSREC and $CSOTH. The first two letter defines the sentences source, CS = CyberSwan. The next three letters defining the type of sentence.

$CSGPS is a collection of information from the CS's GPS receiver. Since one NMEA 0183 does not contain all data of importance (Section 3.2 on page 22), an adapted sentence was made. Now position, speed, course and altitude where collected in the same message.
$CSGPS,124637.000,6325.0840,N,01024.1304,E,1,06,1.0,72.3,M,29.16,46.98*39 is an example, and table 5.1 gives a detailed description of its contents.

Name	Example	Description/Unit
Message ID	$CSGPS	GPS protocol header
Time	161229.487	[hhmmss.sss]
Latitude	6325.0840	[ddmm.mmmm]
N/S Indicator	N	N=north or S=south
Longitude	01024.1304	[dddmm.mmmm]
E/W	W	E=east or W=west
Pos. Fix Indicator	1	See Table 3.3
Satellites Used	06	Range 0 to 12
HDOP	1.0	Horizontal Dilution of Precision
Altitude	72.3	[M]
Speed Over Ground	29.16	[knots]
Course Over Ground	46.98	Relative to true north [deg.]
Checksum	*39	
<CR><LF>		End of message term. (Hex: 0D 0A)

Table 5.1: $CSGPS sentence data format description.

$CSIMU is collected data form the CS's IMU measurement. This message includes accelerations, rate of turn and roll, pitch and yaw angles.
$CSIMU,124637.000,0.01,0.00,9.81,0.01,0.02,0.03,-0.14,0.62,107.04*6B is an example of such a message, with a detailed description in Table 5.2.

$CSSRV and $CSREC messages have the same structure. While the $CSSRV contains information information about the servos, which are the CS's actuators, position. Each of the six channels can have a value between 0

Name	Example	Units	Description
Message ID	$CSIMU		IMU protocol header
Time	124637.000		[hhmmss.sss]
Acc. X	0.01	$\frac{m}{s^2}$	Acceleration
Acc. Y	0.00	$\frac{m}{s^2}$	Acceleration
Acc. Z	9.81	$\frac{m}{s^2}$	Acceleration
RoT X	0.01	$\frac{deg}{s}$	Rate of Turn
RoT Y	0.02	$\frac{deg}{s}$	Rate of Turn
RoT Z	0.03	$\frac{deg}{s}$	Rate of Turn
ϕ (roll)	-0.14	deg	rotation around X_G, $-180° < \phi < 180°$
θ (pitch)	0.62	deg	rotation around Y_G, $-90° < \theta < 90°$
ψ (yaw)	107.04	deg	rotation around Z_G, $-180° < \psi < 180°$
Checksum	*6B		
<CR><LF>			End of message term. (Hex: 0D 0A)

Table 5.2: $CSIMU sentence data format description.

to 255, where 0 defines a far left position, and 255 defines a far right position. $CSREC defines the values for the same channels, but of the handheld receiver used to fly the CS manually. $CSREC,124637.000,138,157,3,170,138,0*75 and $CSSRV,124637.000,182,240,67,186,182,0*14 are examples of these messages. The latter message's details are described in Table 5.3.

Name	Example	Description/Unit
Message ID	$CSSRV	SRV protocol header
Time	124637.000	[hhmmss.sss]
ch0	182	Servo deflection ch 0 [0..255]
ch1	240	Servo deflection ch 1 [0..255]
ch2	67	Servo deflection ch 2 [0..255]
ch3	186	Servo deflection ch 3 [0..255]
ch4	182	Servo deflection ch 4 [0..255]
ch5	0	Servo deflection ch 5 [0..255]
Checksum	*48	
<CR><LF>		End of message term. (Hex: 0D 0A)

Table 5.3: $CSSRV sentence data format description.

$CSOTH contains information from the other sensors available from the CS. $CSOTH,124637.000,6.45,29.16*4F is an example of this message. First, as in the other messages, the messages time is given, then height form the ultrasound range sensor and air speed from the CS's pitot tube.

It was also necessary to design a few messages with a GS origin, giving the source letters GS: $GSPCO, $GSCTR and $GSNAV.

$GSPCO contains the position correction data for the CS GPS receiver. The correction data is the result of the GS GPS received position - the GS position. The theory behind this is explained in Section 3.2 on page 24 A positive latitudal value defines a position North of the GS position, a positive longitudal value defines a position East of the GS position, and a positive altitude defines a higher altitude. The CS GPS position is corrected by substracting the correctional data from its position. Table 5.4 contains the details about the example sentence $GSPCO,100223.000,0.00002083,-0.00007833,-6.9*73.

Name	Example	Description/Unit
Message ID	$GSPCO	PCO protocol header
Time	100223.000	[hhmmss.sss]
Latitude corr.	0.00002083	[deg.]
Longitude corr.	-0.00007833	[deg.]
Altitude corr.	-6.9	[m]
Checksum	*73	
<CR><LF>		End of message term. (Hex: 0D 0A)

Table 5.4: $GSPCO sentence data format description.

$GSCTR is the control sentence sent to the CS for ordering messages. There are five different types of messages. The number given, defines how many messages of that type is wanted each second. Table 5.5 gives the details of the example sentence $GSCTR,2,1,0,0,0*4E.

Name	Example	Description/Unit
Message ID	$GSCTR	CTR protocol header
IMU	2	Frequency [Hz]
GPS	1	Frequency [Hz]
RECEIVER	0	Frequency [Hz]
SERVO	0	Frequency [Hz]
OTHER	0	Frequency [Hz]
Checksum	*73	
<CR><LF>		End of message term. (Hex: 0D 0A)

Table 5.5: $GSCTR sentence data format description.

$GSNAV is the sentence containing the waypoint data for the navigation route planning. $CSNAV,123152.000,6325.2600,N,01023.8800,E,6325.1400,N, 01024.3600,E,6324.9000,N,01023.94000,E*42 is an example sentence, with details in Table 5.6.

Name	Example	Description/Unit
Message ID	$GSNAV	NAV protocol header
Time	123152.000	[hhmmss.sss]
Latitude Wp 1	6325.2600	[ddmm.mmmm]
N/S	N	N=north or S=south
Longitude Wp 1	01023.8800	[dddmm.mmmm]
E/W	E	E=east or W=west
Latitude Wp 2	6325.1400	[ddmm.mmmm]
N/S	N	N=north or S=south
Longitude Wp 2	01024.3600	[dddmm.mmmm]
E/W	E	E=east or W=west
Latitude Wp 3	6324.9000	[ddmm.mmmm]
N/S	N	N=north or S=south
Longitude Wp 3	01023.94000	[dddmm.mmmm]
E/W	E	E=east or W=west
Checksum	*42	
<CR><LF>		End of message term. (Hex: 0D 0A)

Table 5.6: $GSNAV sentence data format description.

Acknownledge is needed as a confirmation that the $GSNAV sentence was received. This will always have the structure $CSNAV,ACK*2C.

5.3 Software development

Software developed in LabVIEW includes front panels and block diagrams. The block diagrams can easily become large and complex. This can to some extent be solved by creating subVIs, reducing its size. This also gives a higher abstraction level, making it possible to present the developed program in a sensible fashion. Then each subVI can be presented down to a chosen detail level. It is chosen to first present the developed VIs main structures in order to explain their function. Then each individual subVI is explained once, in order of appearance. When showing the subVIs, its graphical icon are added to the block diagram, in its top left corner, for identification in the VI it is used. The subVI's that are not explained are enclosed as a pdf in in Appendix A. This will be noted in the text as they appear. Since the GS's front panels were presented in the previous chapter, this section will focus on the VIs block diagram structure.

Ground Station

Functional description

The program structure consist of one main program, the GS VI, which only job is to load the diffent VI's developed into panels on the VI's front panel. Chosing a tab in the top left corner of the front panel (Figure 4.1a on page 37) changes which VI is displayed. Each VI can be thought of as a module working on its own, beeing coupled together by regards of data from other VIs in order to function. The passing of information between VIs are realised using global variables. Doing it this way made it possible to develop each function individually before connecting them together into one program.

The GS concists of five panels:

- Transceiver
- GPS
- IMU
- Plotter
- Sensors

Figure 5.6 shows the copling between the VIs. The red lines are global boolean variables, while the blue lines are lines for passing data between the VIs.

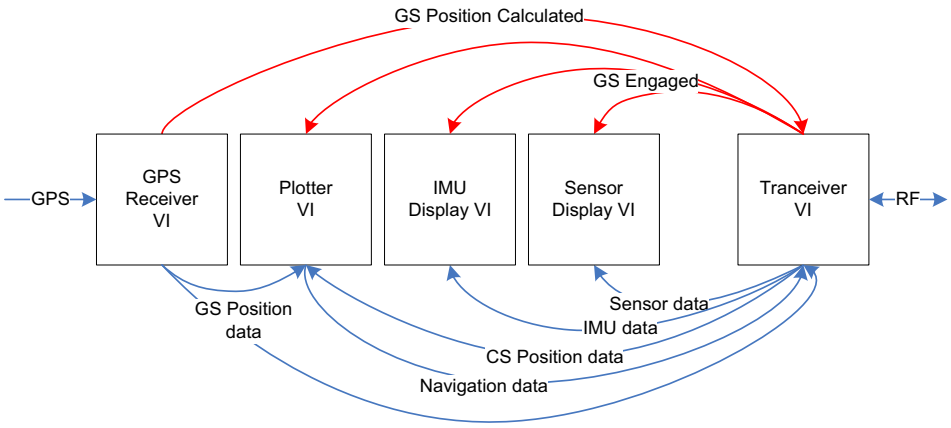


Figure 5.6: Ground Station coupling block diagram.

The GPS and the Transceiver VIs are the core part of the GS. Before it can be used the GS's position have to be calculated. The theory behind this is discussed in Section 3.2 on page 24 and explained in Chapter 4 on page 35. When this is done the global *GS Position Calculated* signal is set. When the GS is engaged, by pushing the engage-button on the Transceiver VI's front panel, this sets the global *GS Engaged signal* variable. This variable, as indicated in Figure 5.6, enables the plotter, IMU display and sensor display VIs. The GS is now operational.

Transceiver VI

The complete transceiver VI's block diagram was to big to fit in one page. See the pdf-file in Appendix A for the complete diagram.

The first thing that happens when the transciever VI is started is the serial port configuration, which is executed once. Then, the VI waits for the global variable *GS position calculated* and for the user to press the *Engage* button. When the GS position is calculated the VI can now be in two states: engaged or not engaged. It can only go into the engaged state if the GPS VI has calculated the GS position and the *Engage* button on its front panel are pressed.

In engaged state, which is the operational GS state, two while loop work in parallel to communicate with the other VIs and with the CS through a serial port. Figure 5.7 shows the reception while loop. Incoming messages are continously received and presented in indicators on its front panel, in addition they are written to their respective global variable. Here the global variables from the GPS and Plotter VIs are read and written to indicators.

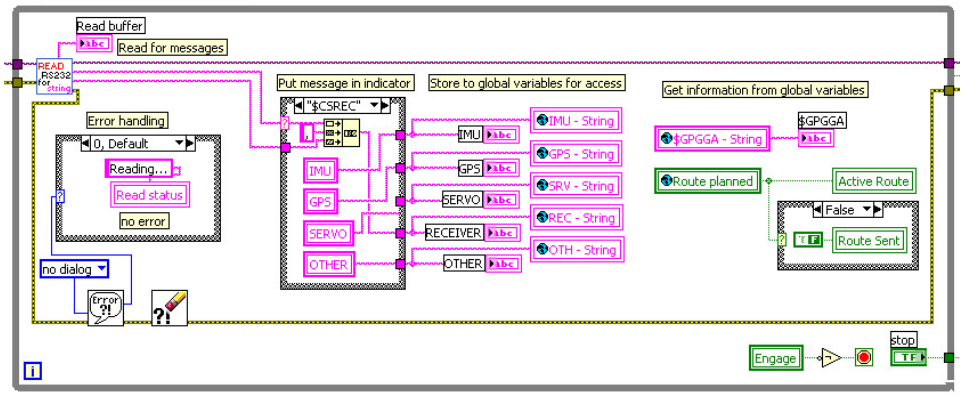


Figure 5.7: Receiver while loop for reception of messages from the CS.

The RS232 config subVI in Figure 5.8 shows how the serial port is configurated. The VISA resource name decides which port is opened. In the transeiver VI this port is hard coded to COM1. As a finished product this could have been solved differently, for example as a choice on the GS front panel, or with a pop-up when it's starting. Further, configuration is set to terminate reading on line feed character, a time out of 10 ms, and with a data rate of 19200 baud. The subVI also sets a standard 8 data bits, no parity and one stop bit. There is currently no flow control implemented, but this can be configured here if so is wanted. Next the buffer size is set, and an option for sofware flow control character setting is given.

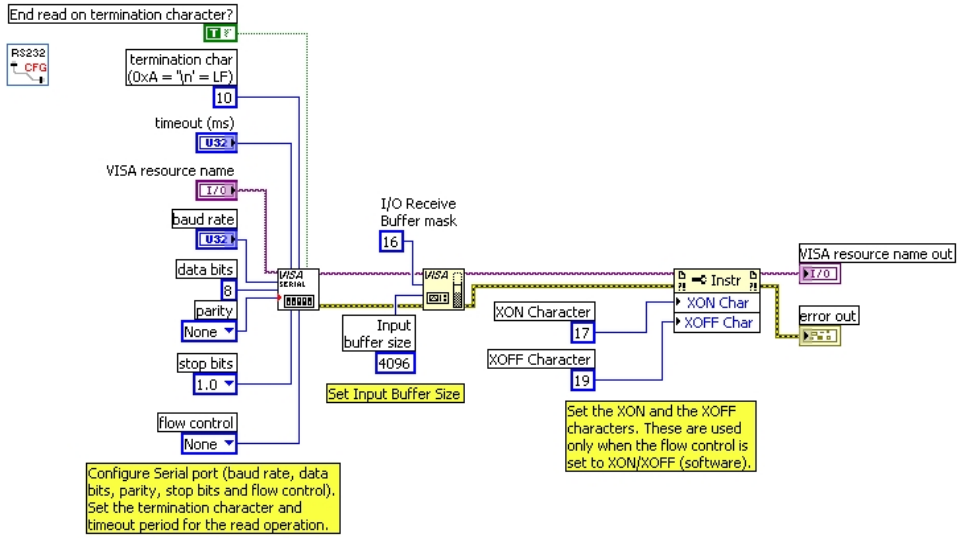


Figure 5.8: RS232 config subVI.

The READ RS232 from string subVI in Figure 5.9 shows how data is read and controlled before parsing. If a time out occurs, which will happen quite frequently since data comes asynchrounously, an error will occur. This error is therefore cleared in the bottom right case, using a simple error handler. The checksum control subVI checks if the received sentence has the correct checksum, if not, an error message will appear on the front panel. Next, provided the sentence has been received correctly, it will be split in two messages, one providing the type of message, an the second the message's information.

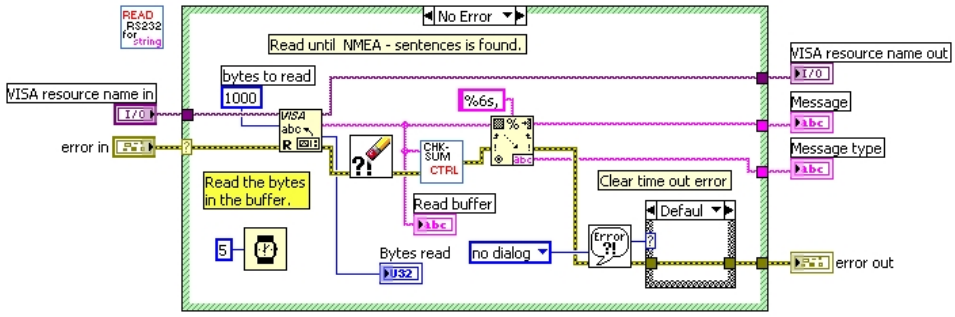


Figure 5.9: Read RS232 for message subVI.

The message type, and its information are so combined again, using a case structure, placed in its indicator on the front panel, and at the same time written to its global variable.

The Checksum control subVI in Figure 5.10 uses the checksum generator subVI to calculate the checksum on the input sentence. This checksum is then used as an input, to see if the input sentence contains the same checksum. If it does not, an error is raised using the false case in the case statement.

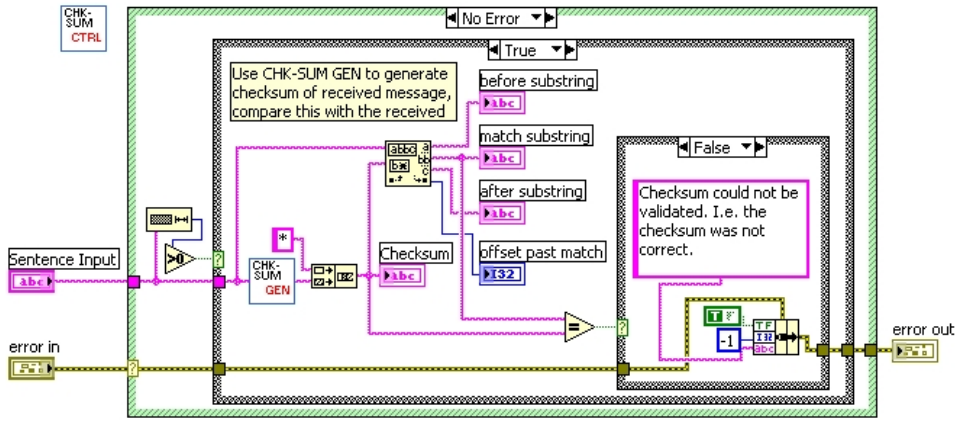


Figure 5.10: Checksum control subVI.

The Checksum generator subVI in Figure 5.11 uses a MathScript node to generate the checksum for an input sentence. The characters in the sentence's string are arranged in an array, and the code traverses through the characters between $ and * using bitwise exclusive or. This produces a two digit hexadecimal value, the checksum.

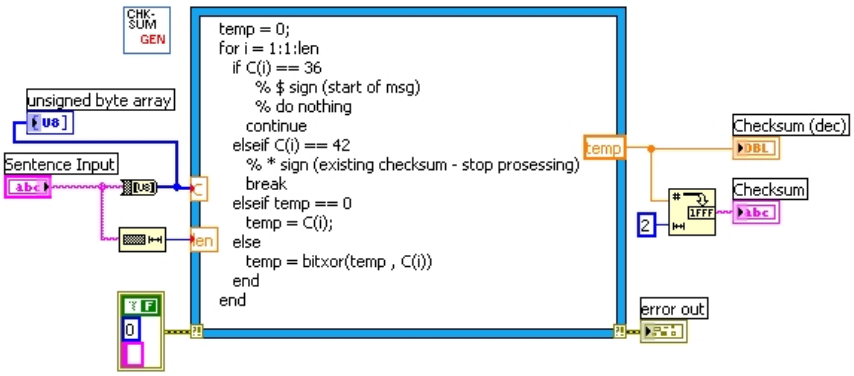


Figure 5.11: Checksum generator subVI.

The second while loop, shown in Figure 5.12, handles the transmission of messages to the CS. The while loop contains a flat sequence structure with three frames that execute sequentially. In the first frame the GS position is transmitted, on the first call once engaged, and then only if it has been recalculated. In the second frame the position correction message is transmitted each second if the respective button on the front panel is pressed. In the third and last frame the order message is generated with respect of choises made from the *Messages* table on the front panel, and transmitted if the *Send order* button is pressed. This message is the basis for which messages one wants to receive from the CS.

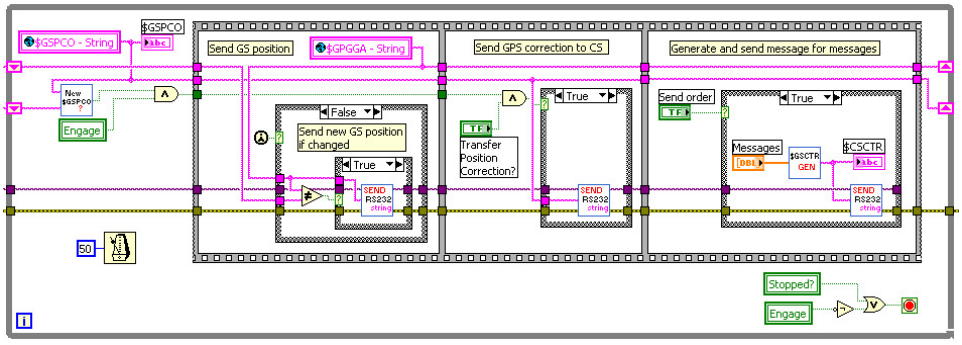


Figure 5.12: While loop for transmission of messages to the CS.

The Check for new $GSPCO string subVI in Figure 5.13 compares the sentence type and time information in two strings, giving a true output if they are different. This is used to ensure that the position correction signal is only sent once a second, ignoring that the position might change once in between, due to the GPS receiving two different NMEA sentences.

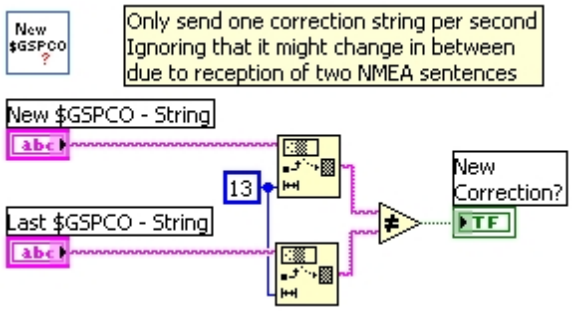


Figure 5.13: Check for new $GSPCO string subVI.

The Send RS232 message subVI in Figure 5.14 transmits the input sentence to the write buffer, on the serial port chosen by the VISA resource name, provided its longer than six characters.

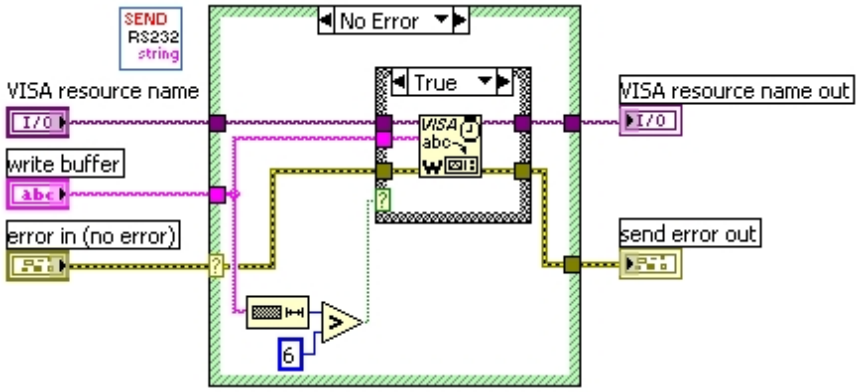


Figure 5.14: Send RS232 message subVI.

The $GSCTR generator subVI in Figure 5.15 uses the numbers in the *Messages* array on the front panel to concatenate the $GSCTR string. Then the checksum generator is used to calculate the strings checksum, concatenating it with carriage return and line feed constants making it a proper sentence for transmission.

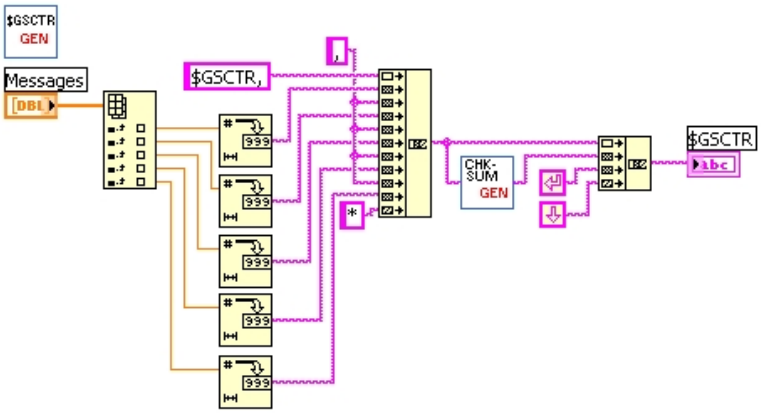


Figure 5.15: $GSCTR generator subVI.

When in not engaged state, the transmission of an active navigation route is possible. Figure 5.16 shows the case where the navigational message will be sent once, then the CS has some time to send an response. The while loop times out after 10 ms, and the number of times the while loop interates defines the response window. If an response is received, the case sentence frame "$CSNAV" will compare the received message with "ACK". If this is what was received, a "true" respose will be given, stopping the while loop, and setting a confirmation light on the front panel.

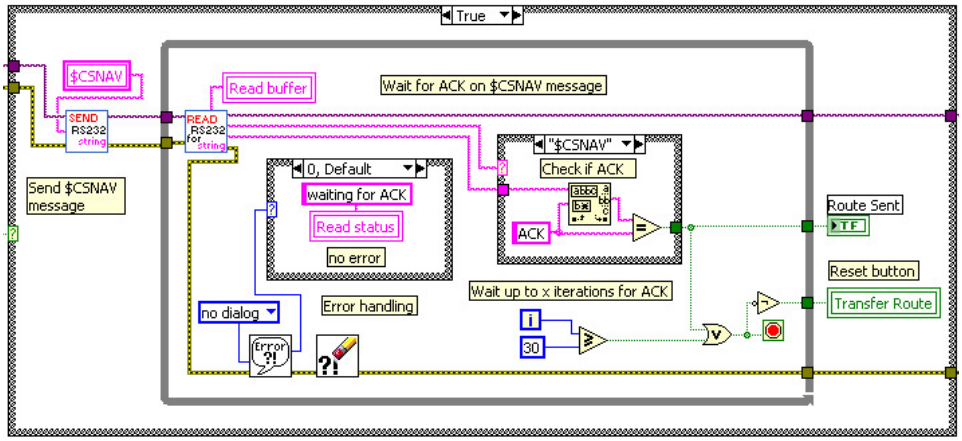


Figure 5.16: Transmission of navigation message and ACK reception.

GPS receiver VI

Figure 5.18 shows the complete GPS receiver VI's block diagram. Figure 5.17 shows the main while loop. Before this is entered, the serial port is configurated the same way as in the transceiver VI, only here with a data rate of 9600 baud on COM7. In this main while loop the GS's GPS receiver is continously read, and its position is presented on the front panel. The while loop also includes a case, where the error between the GS's position and the GPS recieved position is calculated. Before the GS position is calcuated the error will be the received position. The calculation of the GS position is performed in the while loop shown in Figure 5.22. The position is calculated while the *Calculate* button on the front panel is pressed. When released, the the calculated position is presented on the front panel, the global *GS Position Calculated* variable is set, a $GPGGA NMEA-string is generated and written to its global variable, *$GPGGA-String*, for transmission to the CS.

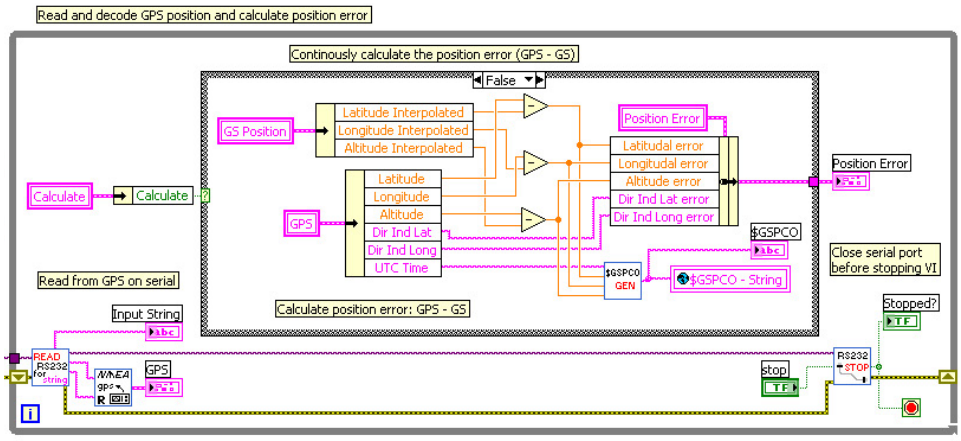


Figure 5.17: Main GPS reciever while loop and position error calculation.

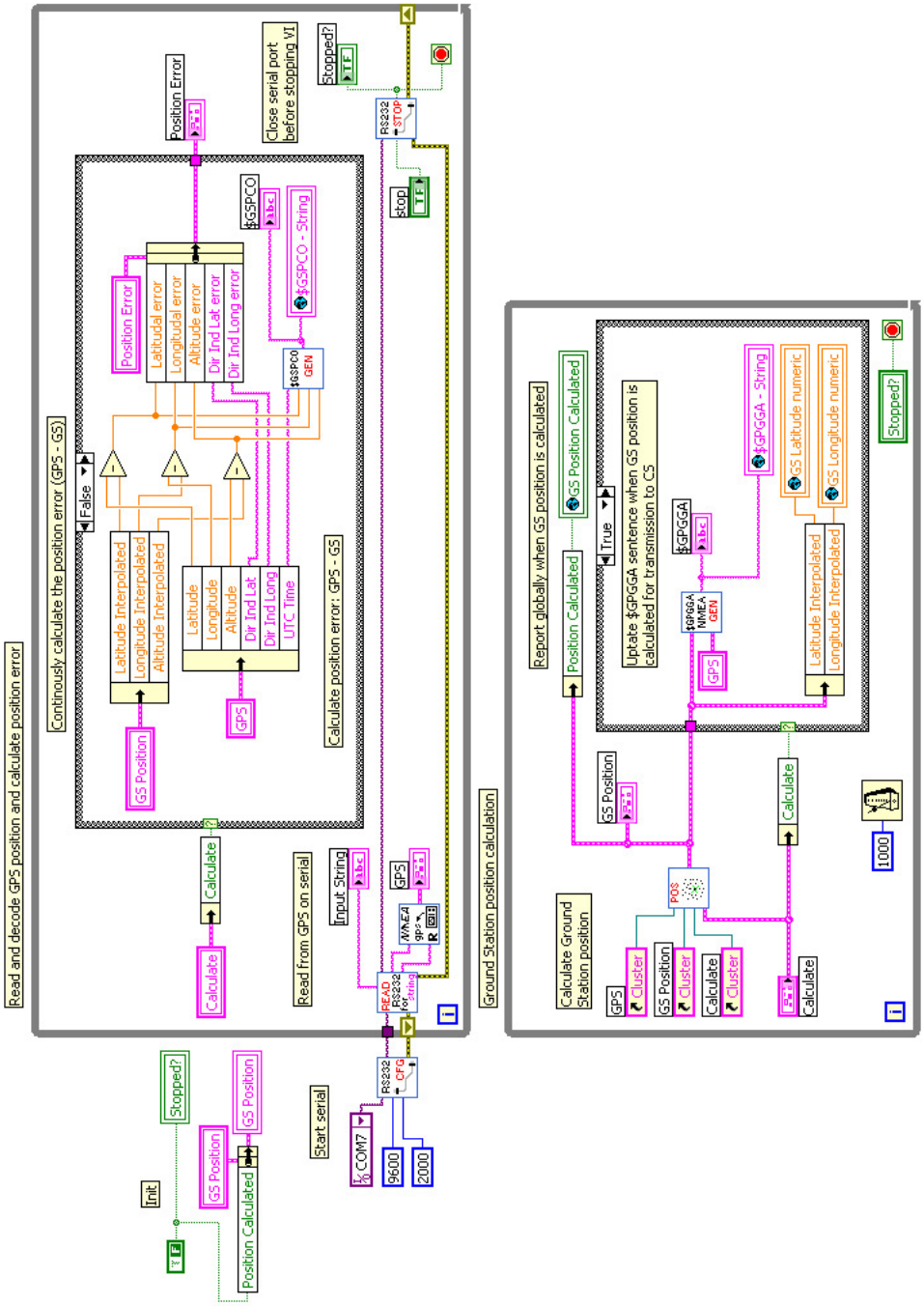


Figure 5.18: GPS receiver VI block diagram.

The Read & format NMEA GPS subVI in Figure 5.19 gets the received sentence type and data as an input. This is then fed to the Read GPS NMEA subVI for parsing. Its output is then bundeled and fed to the GPS display. Before the position is bundeled, it goes through the *Decode & format position string* subVI to be recalculated to degrees, and gets added a degree sign. The two latter VIs are documented in Appendix A.

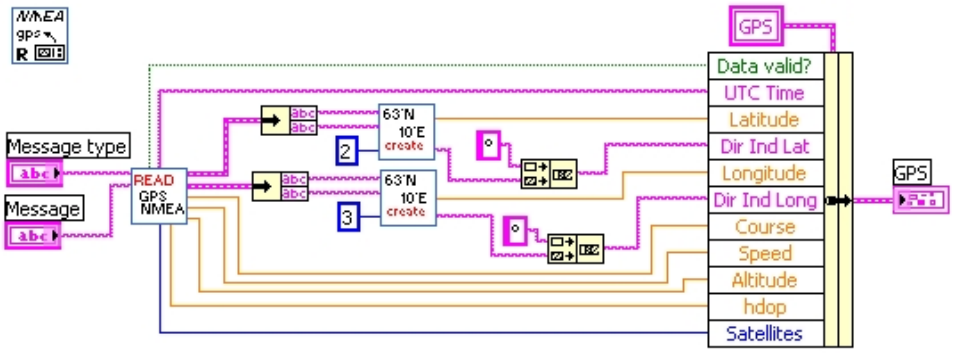


Figure 5.19: Read & format NMEA GPS subVI.

The $GSPCO generator subVI in Figure 5.20 is located in the "false" case in the case structure. It takes the time and position error variables, and concatenates it into a string. A checksum, carriage return and line feed constants are concenated to make it the valid $GSPCO sentence ready for transmission.

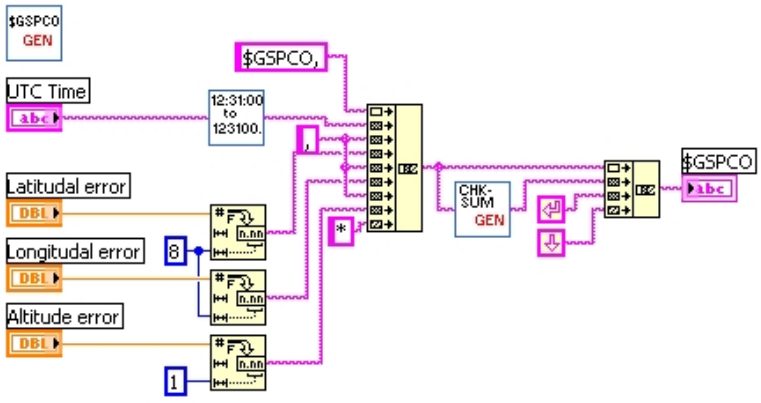


Figure 5.20: $GSPCO generator subVI.

The “false” case in the main while loop, Figure 5.21, is active when the GS position is calculated. This sets the position error to zero during the calculation. It also controls the lights on the GS position display.

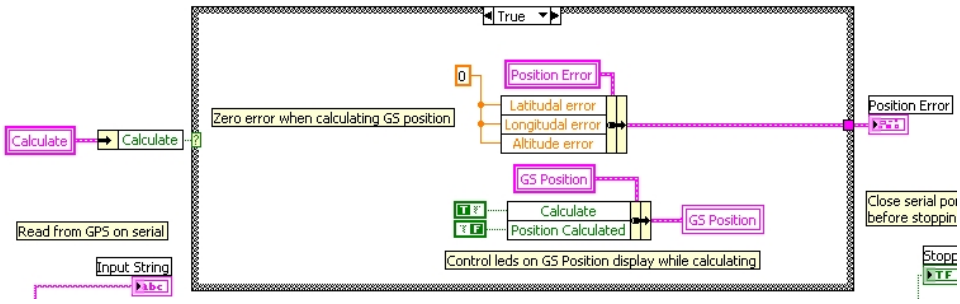


Figure 5.21: GPS true case when calculating GS position.

In the second while loop, Figure 5.22, the GS position calculation takes place, in the Interpolate position subVI. When this is finished the $GPGGA sentence will be generated, updating its display and global variables.

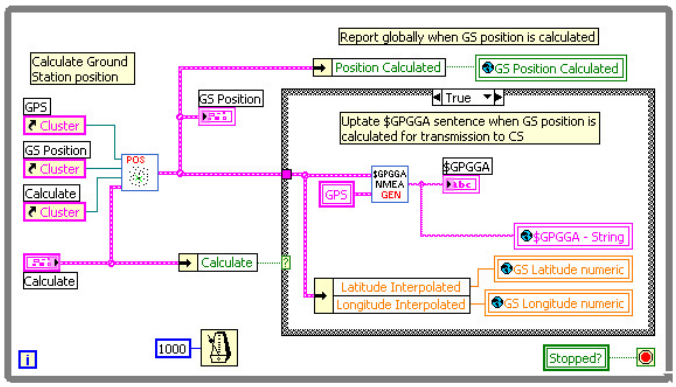


Figure 5.22: While loop for calculating and reporting the GS position.

The Interpolate position subVI, represented by the "true" case in Figure 5.24, takes the following references: *GPS* display, *GS position* display and the *Calculate* button, in addition to the *Calculate* button's boolean value. References are used to be able to read the respective variables change in values from inside the subVI. While the *Calculate* button on the front panel is "true", the latitude, longitude and altitude will be added to each their array, continously incrementing in size, until the *Calculate* button is released. Then each array is interpolated to its mean value. These values represents the GS position and are written to its display. The display will also hold the time the calculation took. In the "false" case the GS position display is merely kept unchanged. This figure is available as a pdf in Appendix A.

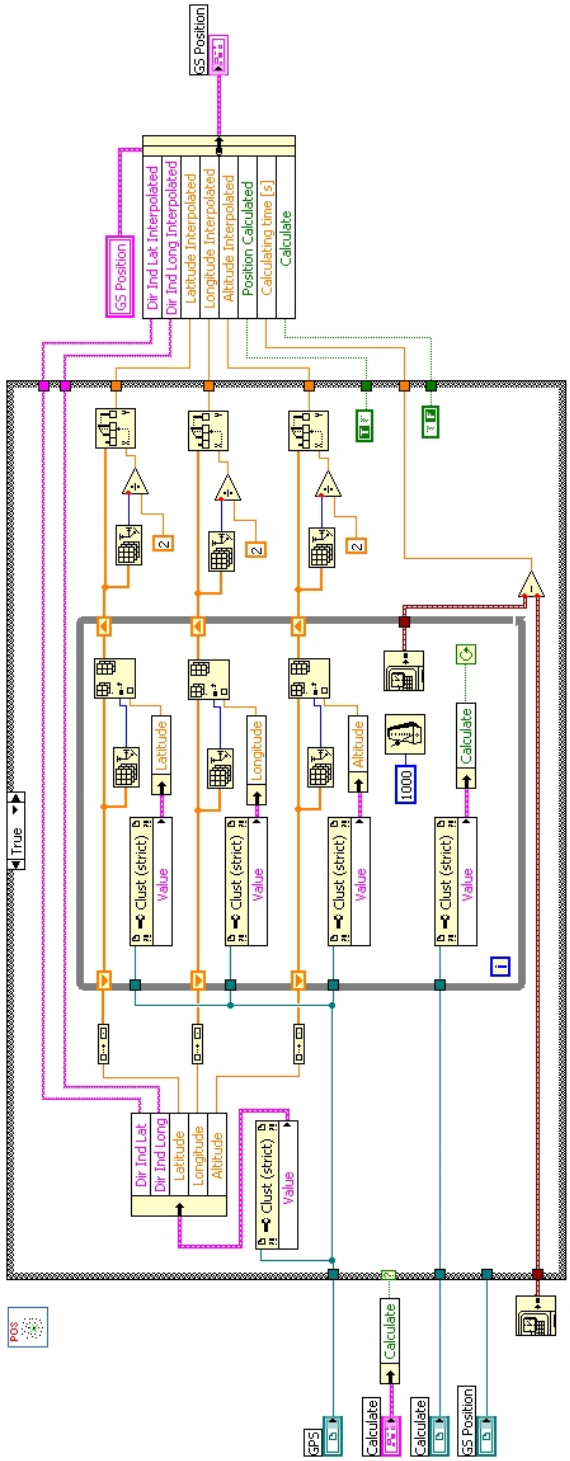


Figure 5.23: GS position calculation with Interpolate position subVI.

The $GPGGA generator subVI in Figure 5.24 uses the calculated GS position information to generate a NMEA $GPGGA sentence. The transciever unit will send this sentence to the CS. The subVIs used to format time, latitudal and longitudal representation, in addition to the format satellite string. These subVIs are not described here, but enclosed as pdf files in Appendix A.

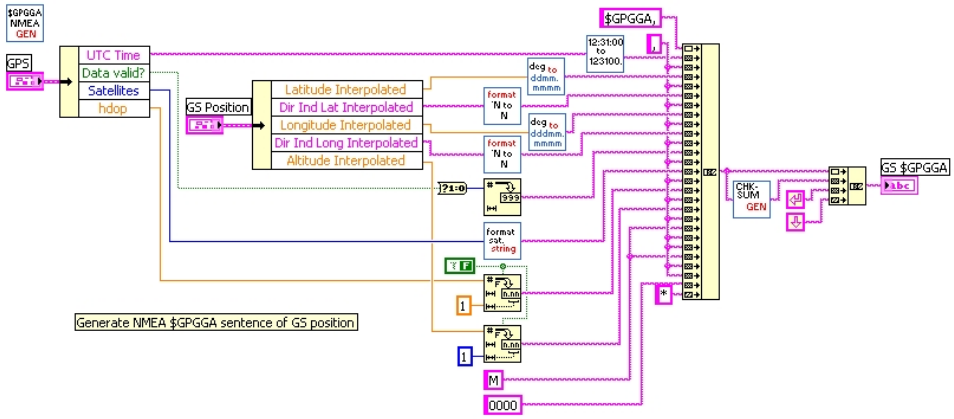


Figure 5.24: $GPGGA generator subVI.

Plotter VI

Figure 5.26 on the facing page shows the complete plotter VI block diagram. The GS needs to be engaged before the plotter can be used. Before entering the main while loop, shown in Figure 5.25, the map is drawn up, and the GS position is drawn on top of the map. Then, inside the while loop, if a navigation route is active, this will be drawn up. Next the CS plot will be draw, either as a countinously plot, or just the last position, dependent on the choice made on the front panel. If the GS position is recalculated the *Reset GS* button must be pushed in order to draw up its new position on the map. In order to plot the CS position, it has to be constantly updated. This is done in the while loop shown in Figure 5.33 on page 72. In the same while loop the distance between the CS and the GS is continously calculated and presented in an indicator on the front panel.

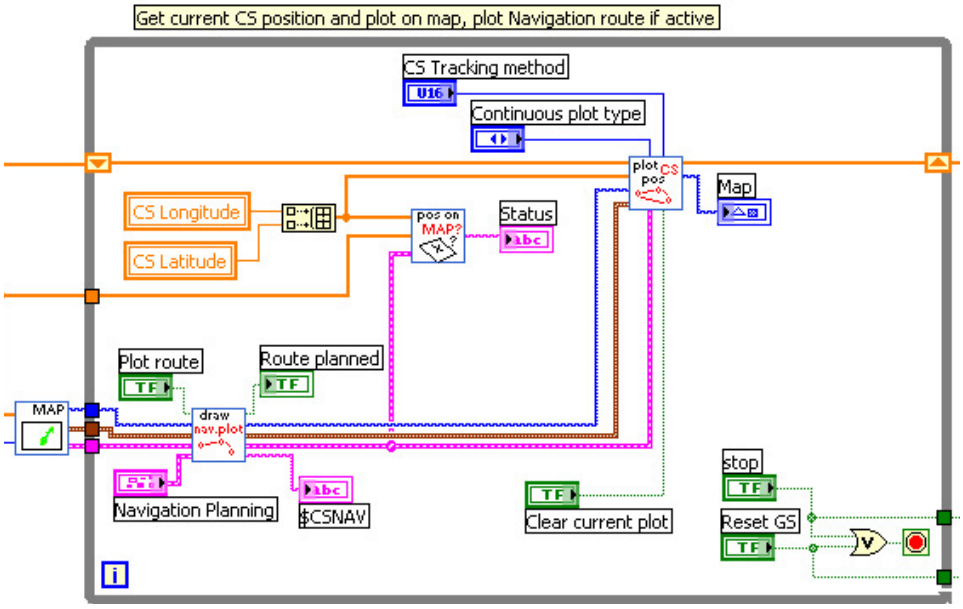


Figure 5.25: Plotting of map, route, GS and CS position.

The Draw map subVI in Figure 5.27 uses a case structure which provides the map to be drawn up. Additional maps can be added in consecutive cases. The map and the GS position plot is drawn up in three stages: The GS position is formatted, the map is loaded and drawn up, and the GS plot is drawn up. The GS position enters the case and are formated to the array format for plotting. The maps are created by using publicly available airial

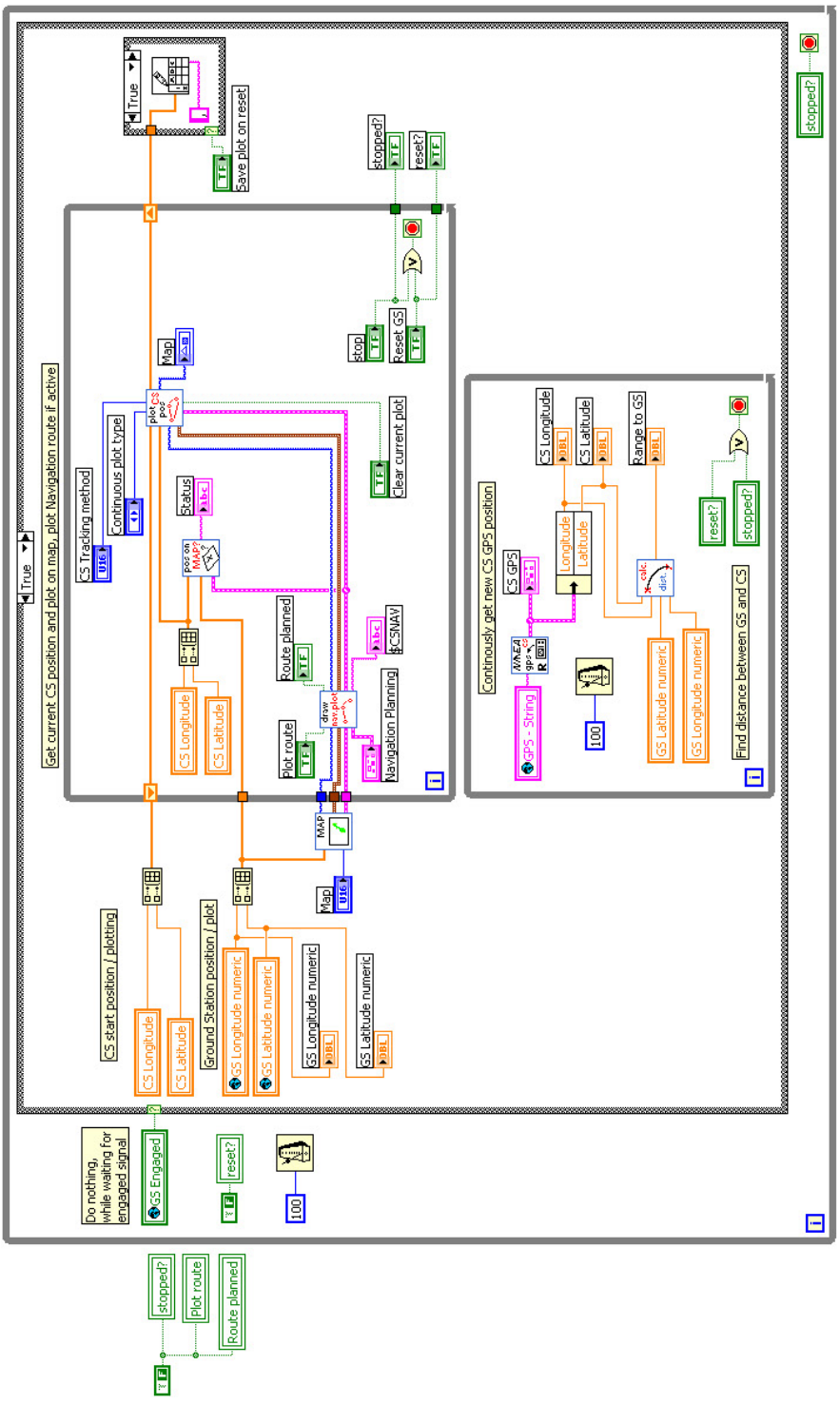


Figure 5.26: Plotter VI block diagram.

photos from gulesider.no. It was trimmed to 600 · 600 pixels and stored as a bmp picture. The map is loaded and drawn up as a flattened pixmap. The constants at the bottom of the figure defines the limits for the selected map, latitudal and longitudal, in addition to the plot type and colour. These limits were found using the interactive map Norgesglasset (Statens Kartverk 2007) for the map in question. The GS plot will then be plotted on top of the loaded map, as a small red square, provided its within the map's limits.

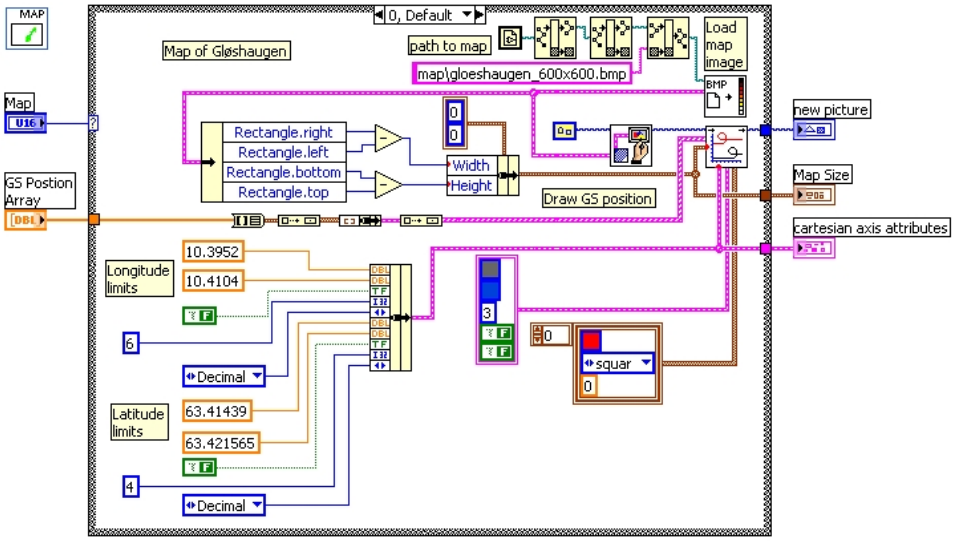


Figure 5.27: Draw map subVI.

The Draw navigation plot subVI in Figure 5.28 takes an active navigation route and plots it on top of the map. The case shown is executed only if the *Plot route* button on the front panel is pressed. The routes three waypoints are fed to the create plot subVI which formats it to the array needed for plotting. The route is then plotted on top of the map, the waypoints as small red squares with a red line between them. The route is also fed to the $CSNAV generator subVI in order to generate the string for transmission to the CS using the tranceiver front panel. For this the string are stored to the global variable $CSNAV string and the Route planned global variable are set to "true". In the "false" case the Route planned global variable is set to false and the $CSNAV string are cleared.

The $GSNAV generator subVI in Figure 5.29 takes the waypoint information entered in the front panel and formats this to the defined $CSNAV string.

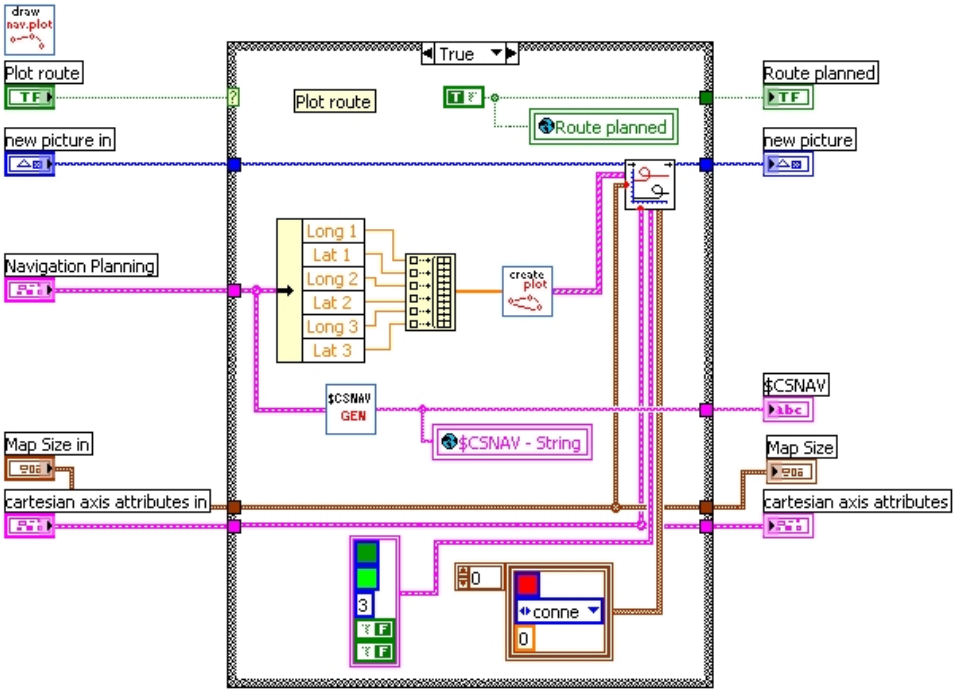


Figure 5.28: Draw navigation plot subVI.

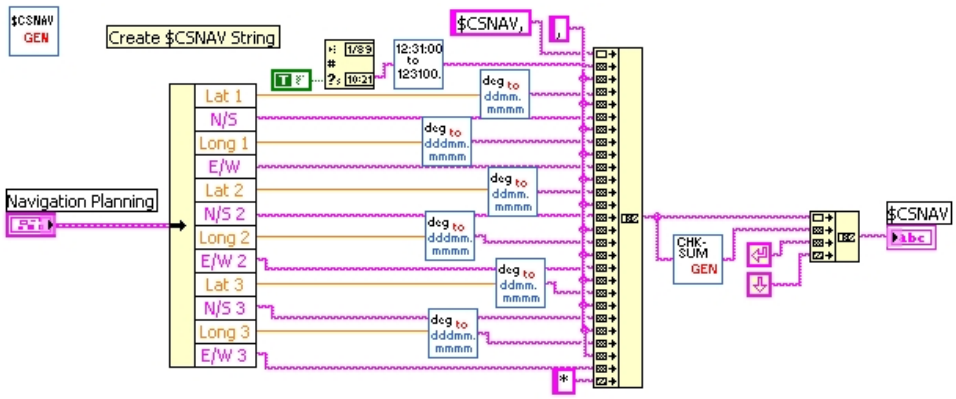


Figure 5.29: $CSNAV sentence generator subVI.

The Create array to plot subVI in Figure 5.30 changes the structure of the array of plots. The input array only has one long row with latitude and longitude coordinates. When entering the while loop, the two first values, the first plot, are added as an initial value for the new array. Then, as the while loop iterates, the two next values, the next plot, will be added in a new row. When the whole input array is arranged the same way, the output array will have as many rows as it has plots.

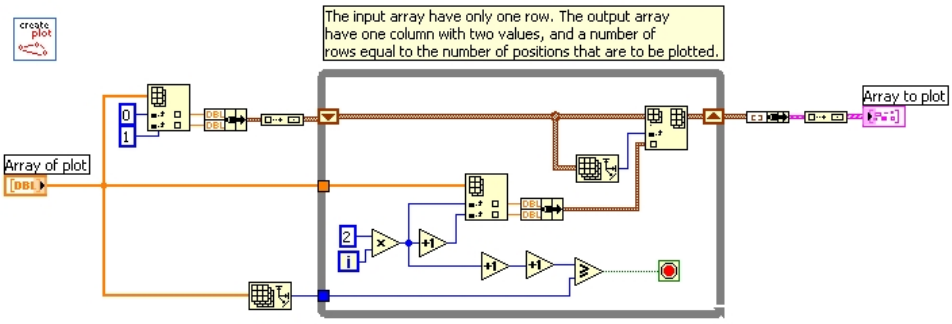


Figure 5.30: Create array to plot subVI.

The GS & CS on map? subVI in Figure 5.31 takes the map limits, and checks if the input GS and CS positions are within these limits. If it is, the case structures "false" case will return an "OK..." message, if not the "true" case will return a "Warning! GS/CS outside map area" message, which is printed on the plotters status display on the front panel.

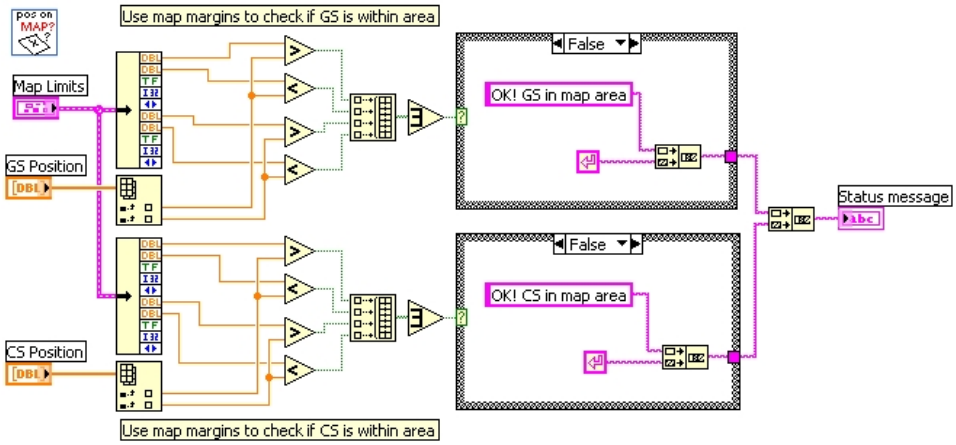
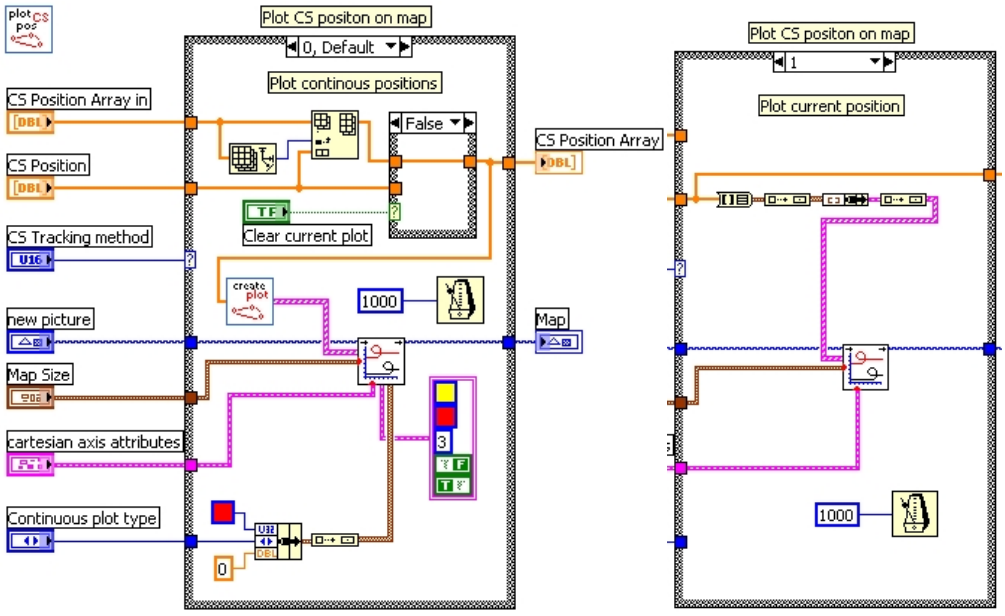


Figure 5.31: GS & CS within map limits check subVI.

The Draw CS position plot subVI in Figure 5.32 uses a case structure with two cases. On the plotter front panel one can chose if the CS position is to be plotted continously or just plot the current position. The continous plot case, in Figure 5.32a gets the map as an input, in addition to the CS position, which is continously added to an array for plotting. This is then fed to the create plot subVI and plotted on the map once a second. The next case, in Figure 5.32b, takes the CS position and plots it directly on the map once a second.



(a) Plot CS position continously. **(b)** Plot current CS position.

Figure 5.32: Plot CS position on map.

The Read & format $CSGPS subVI in Figure 5.34 get the $GSGPS message as an input, its parsed in the read $CSGPS subVI and then formatted to be presented in the CS GPS diplay on the front panel.

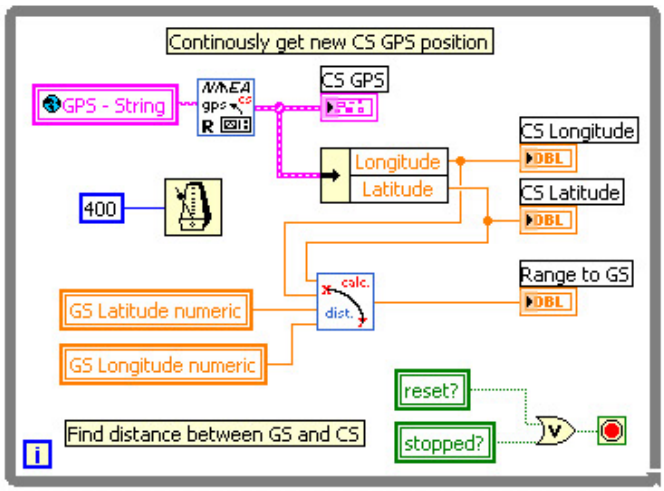


Figure 5.33: CS position update and range to GS calculation.

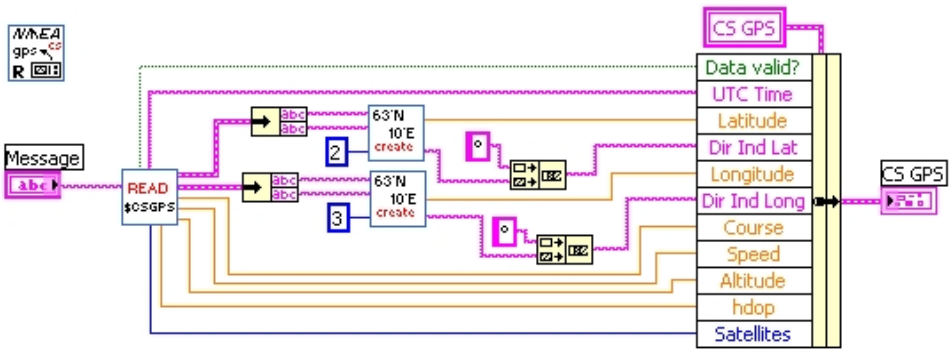


Figure 5.34: Read & format $CSGPS sentence subVI.

The Calculate distance subVI in Figure 5.35 is created using LabVIEW MathScript. It calculates the great-circle distance between two points using the Haversine formula (Chamberlain 2007). The distance returned is the shortest distance over the Earth's surface.

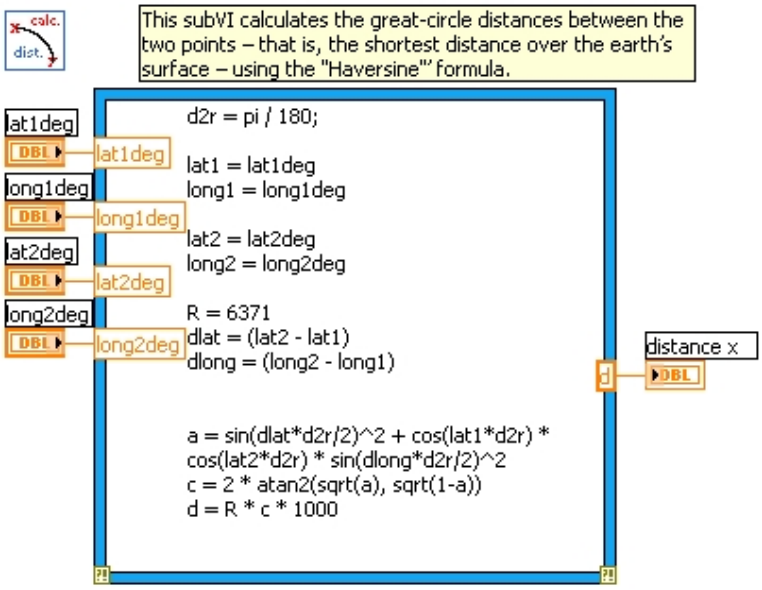


Figure 5.35: Calculate distance between GS and CS subVI.

IMU display VI

The IMU display block diagram in Figure 5.37 on the facing page, will when started, wait and do nothing, until the GS is engaged. Then it will enter the "true" case and read the global IMU string. Now, entering the while loop, the IMU sentence is read from the global variable IMU string. It is compared to the one read in the last interation of the loop, or in the first run, with the sentence read before entering the while loop. The message are passed through a shift register in the while loop. Only when a new message is read will the IMU orientation data be parsed and the 3D model is drawn into the display on the front panel. The CyberSwan 3D subVI contains a subVI that generates the CS 3D model, draws it into the display on the front panel, and rotates the model according to the roll, pitch and yaw angles read from the $CSIMU sentence. The block diagrams for the CyberSwan 3D subVI and CyberSwan 3D-model subVI were to big to be printed in this report, and are available in Appendix A.

When the *Reset display* button on the front panel is pressed will the "true" case in the case structure to the left in the while loop be active and reset the display to its original setup, changing the orientation of the CS 3D model if it has been rotated manually in the display. The "false" cases in this VI are all empty.

The Read IMU data subVI in figure 5.36 reads the inputted $CSIMU sentence and parses it into its respective variables for by the CyberSwan 3D subVI and to be presented in the display on the front panel.

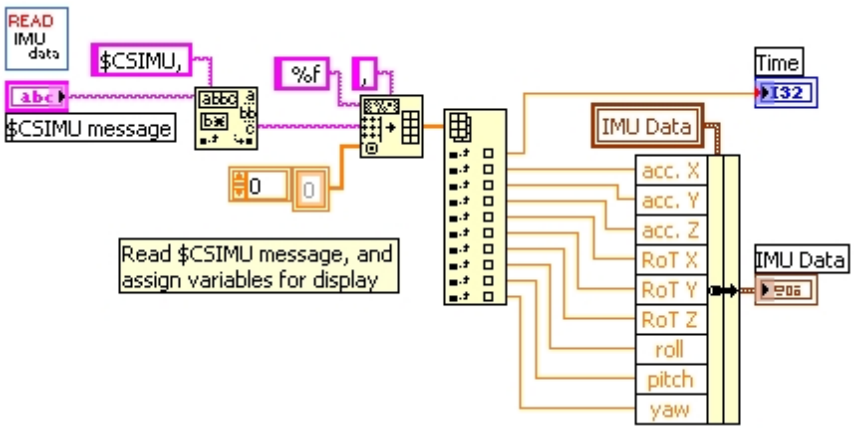


Figure 5.36: Read IMU data subVI.

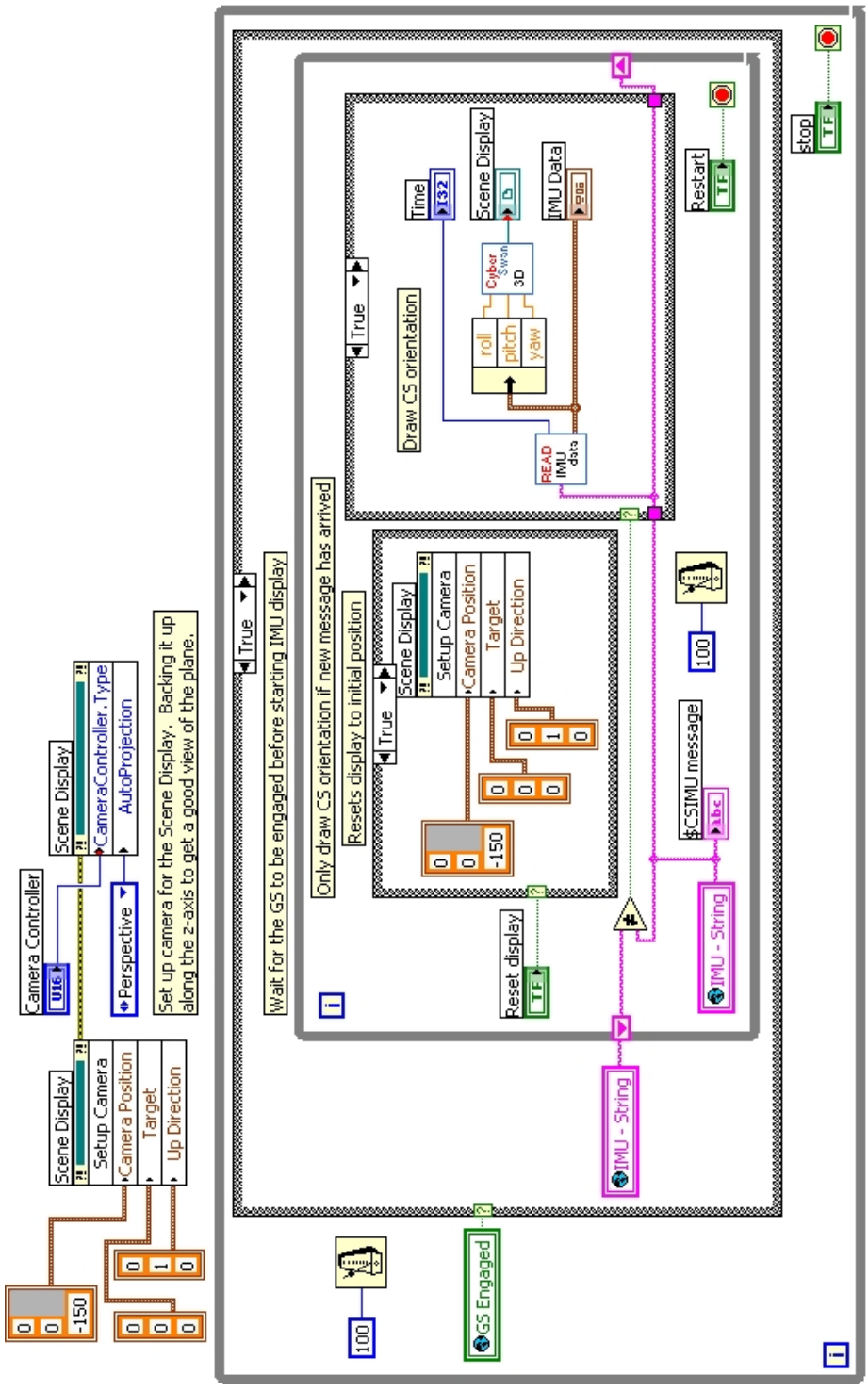


Figure 5.37: IMU VI block diagram.

Sensor display

After the GS has been engaged, the sensor display VI continously reads global variables (Figure 5.38) and presents the information on the front panel. The information read are the last received message of the CS's *Servo* data, the *Receiver* data and the *Other* data, which holds the height measurement from the ultrasound ranging device and the pitot tube airspeed measurement.

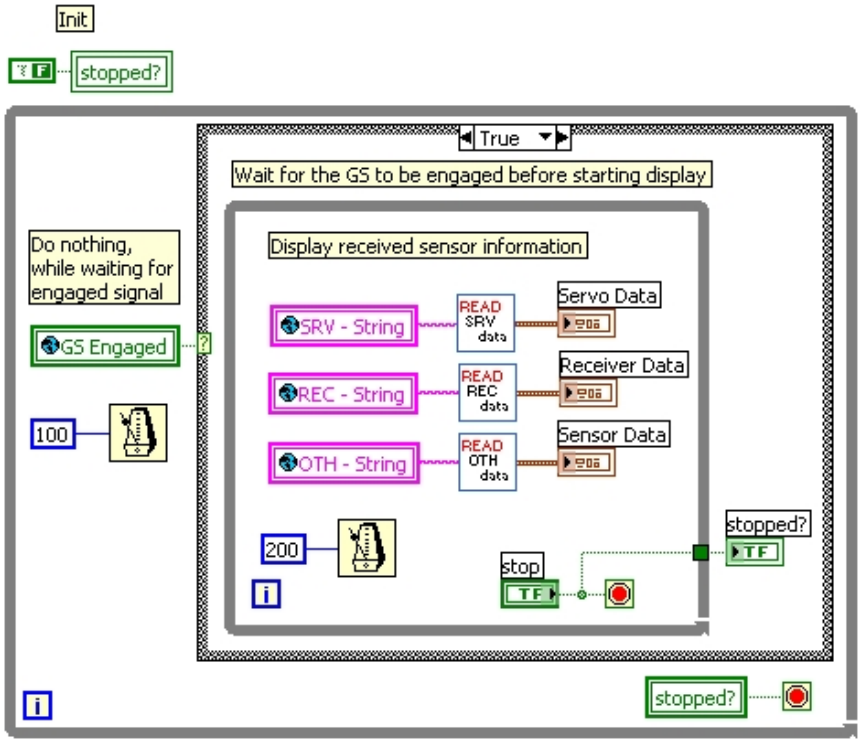


Figure 5.38: Sensors VI block diagram.

Figures 5.39 to 5.41 shows the parsing of the received $CSSRV, $CSREC and $CSOTH messages.

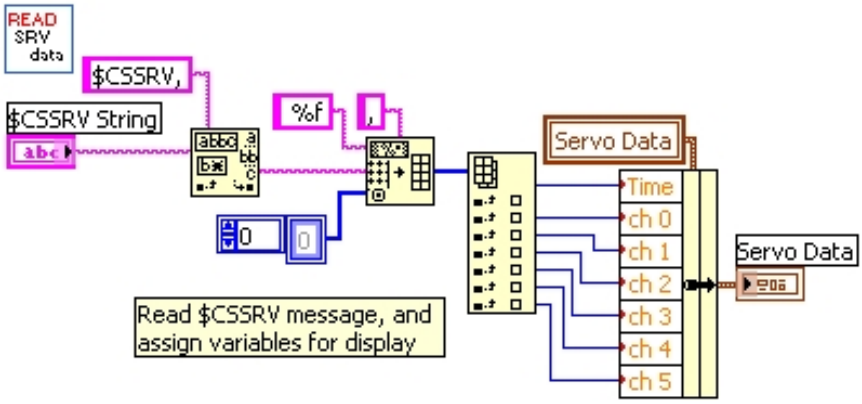


Figure 5.39: Read $CSSRV data subVI.

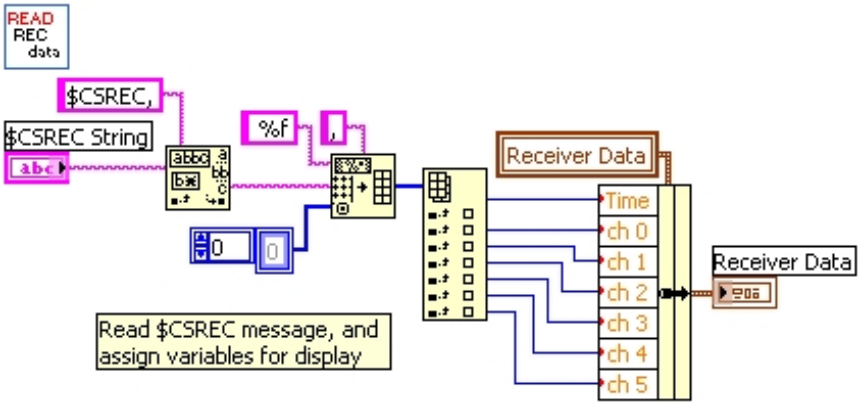


Figure 5.40: Read $CSREC data subVI.

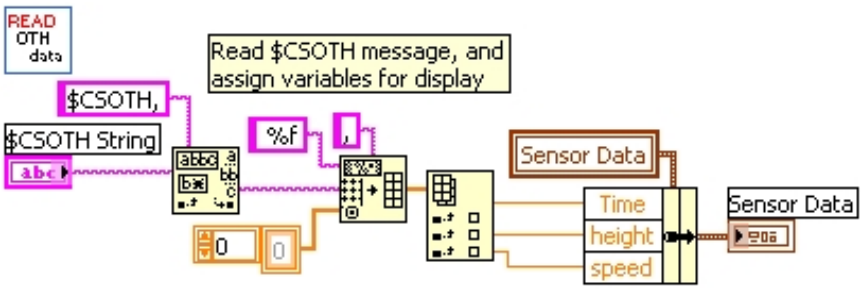


Figure 5.41: Read $CSOTH data subVI.

CS simulator

Functional description

The program structure consist of one main program, the CS simulator VI, which has the same structure as the GS VI. It loads the diffent VIs developed into panels on its front panel. Chosing a tab in the top left corner of the front panel (Figure 5.42 on the next page) changes the displayed VI. Here each VI works individually, only shearing its simulated value with the tranceiver VI for transmission. The passing of information between VIs are realised using global variables.

The CS simulator concists of five panels:

- Transceiver
- IMU
- GPS
- Servo

In the next pages each of these VIs will be described. It is chosen to describe the used subVIs with text only, with one exeption, where a figure also is used. The complete documentation can be found in Appendix A.

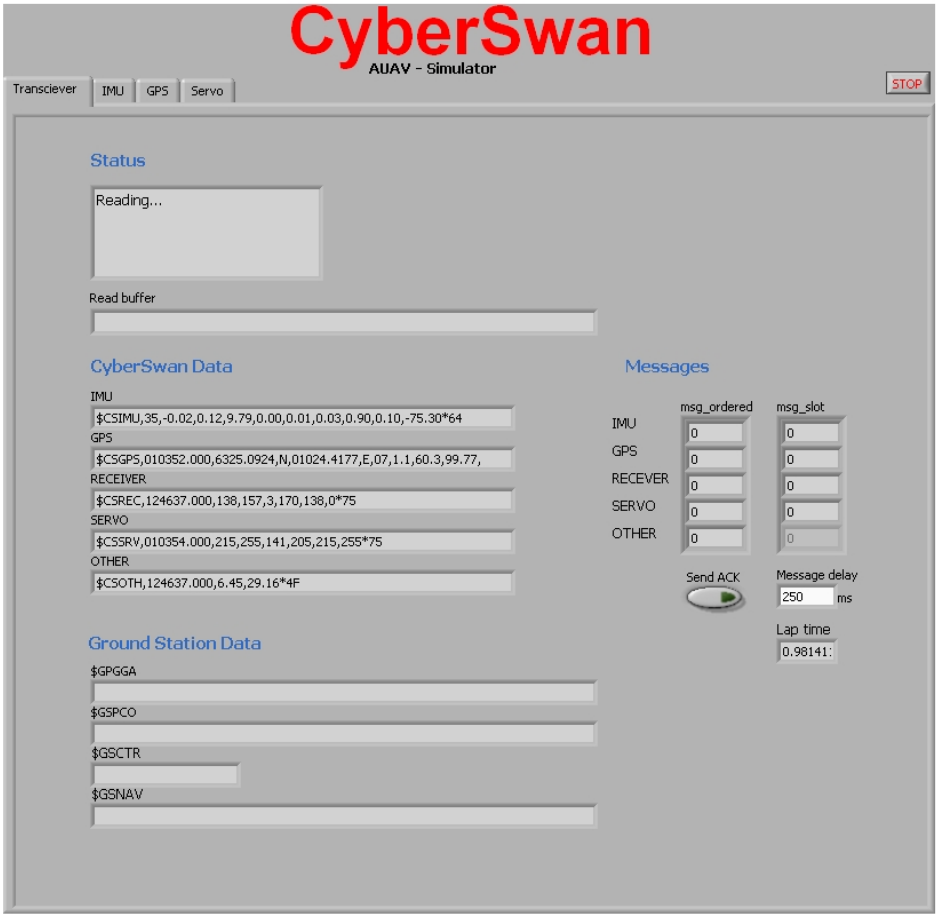


Figure 5.42: CS simulator front panel showing the tranceiver panel.

Transceiver

The transceiver VI block diagram (Figure 5.43 on the facing page) consists of three while loops working in parallel. One for message reception, one to read global variables, and one for message transmission.

The reception while loop reads incoming messages from the configured serial port. The received messages are diplayed, and the $GSCTR message is parsed. In the case of an error, this is diplayed in the *Read status* display.

The global variables updated in the IMU, GPS and servo VIs are read and its respective display on the front panel are updated. The *Receiver* and *Other* displays are not updated, and contains only a static value displayed at the inital start of the VI.

In the transceiver while loop the transmission of messages takes place. The acknowledge for the received navigational route message are sent when the *Send ACK* button is pressed. The transmission of IMU, GPS, servo, receiver and other messages takes place in five message slots placed in a stacked sequence structure. Dependent on the received $GSCTR message, will messages be assigned to be sent from these slots. Each slot contains a case structure defining which message to send in this slot. The first case, 0, is empty, meaning that no message are transmitted. The second case, 1, transmits the IMU message, and so on. In between the slots are a delay, ensuring that the message are transmitted before sending the next.

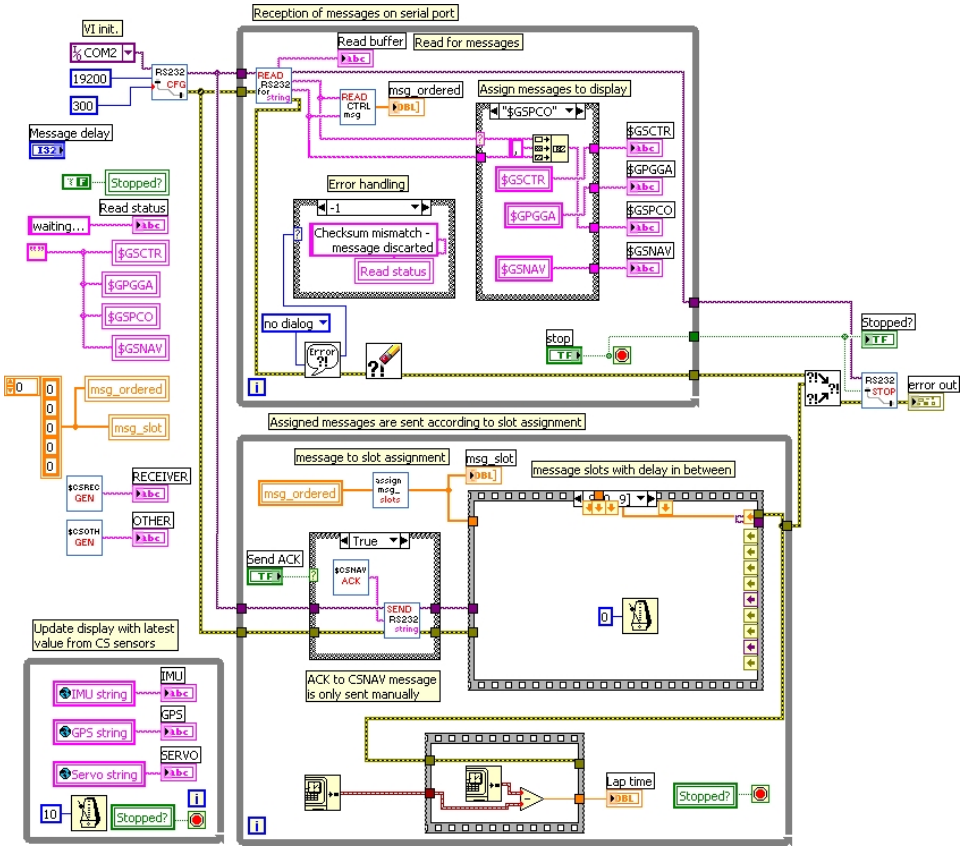


Figure 5.43: CS simulator transceiver block diagram.

The read $GSCTR message subVI parses the received $GSCTR message and writes it the *msg_ordered* array.

The $CSNAV ACK subVI contains the acknowledge message that needs to be returned in order to confirm that the $GSNAV message was received. This acknowledge will always be the same, and is therefore hard coded to $CSNAV,ACK*2C. It will be sent any time the *Send ACK* button is pressed.

The assign messages to slot subVI in Figure 5.44 uses the received $GSCTR message to assign which message are to be transmitted to the GS. The alorithm in the MathScript node assigns transmission from four of the slots, in an order decided by the *q_order* array, here: 1,3,2,4. So if two messages of the IMU type is ordered, these will be sent from slot 1 and 3. The slots are assigned with a message priority decided by its place in the $GSCTR message. The IMU message has the highest priority, and the other message has the lowest. Ordering four messages of the IMU type, will then only assign the IMU message to be transmitted. Since the four available slots now are filled, any other ordered lower priority message will not be sent.

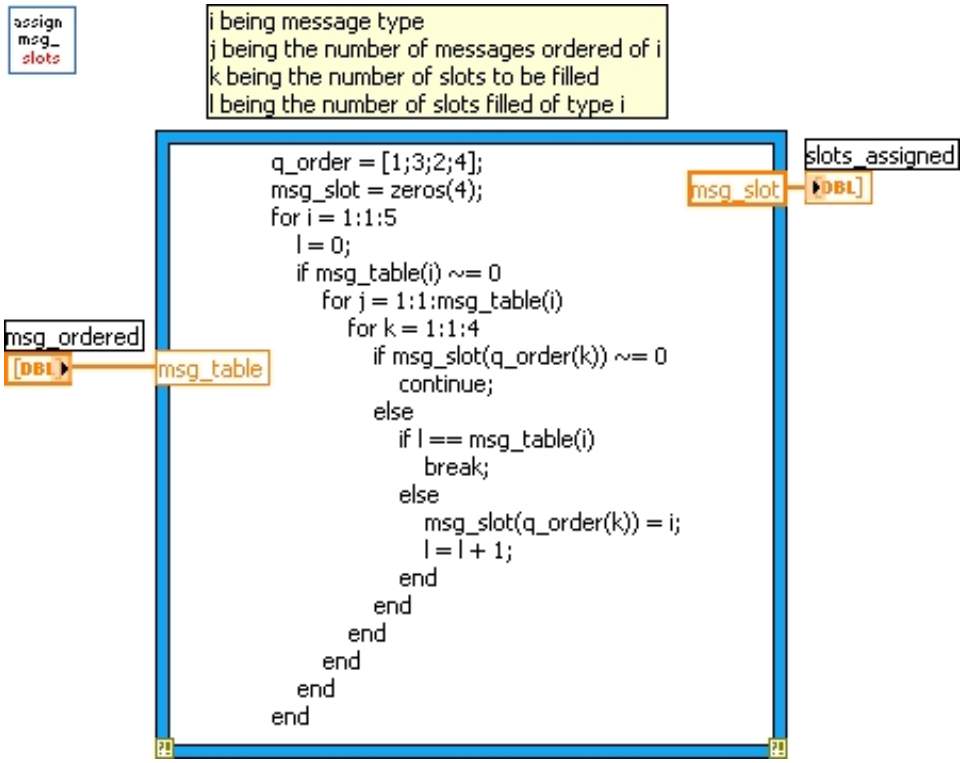


Figure 5.44: Assign messages according to $GSCTR message.

IMU

The IMU simulator front panel in Figure 5.45 shows a 3D model of the CS. It also has a display for the IMU data values, a *Restart* button, a button to reset the 3D models perspective, the *Reset display* button. The source for the 3D models motion are actual data collected from the IMU used in the CS. A test program was developed to be able to collect the data to a file. Then using this progam three files were collected adjusting the IMU in roll, pitch and yaw direction. Each of these files can be chosen using the *Direction* menu button on the front panel. The developed program is available in Appendix A.

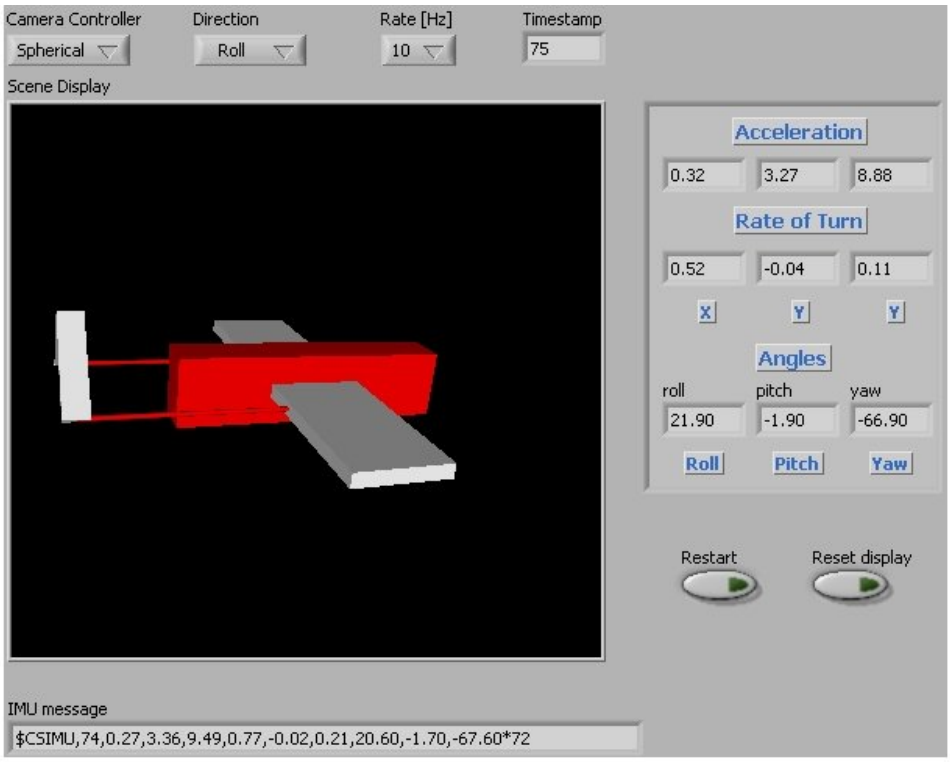


Figure 5.45: IMU simulator front panel.

Figure 5.46 show the block diagram. First the inital perspective is arranged, and then the chosen log file is read. The while loop iterates reading the table row for row, drawing the 3D model, and rotates it respective to the read IMU data's roll, pitch and yaw angle. This read data is also written to a display

on the front panel, and to a global variable, accessable for the tranceiver VI. The CyberSwan 3D model generation and rotation subVIs are available in Appendix A.

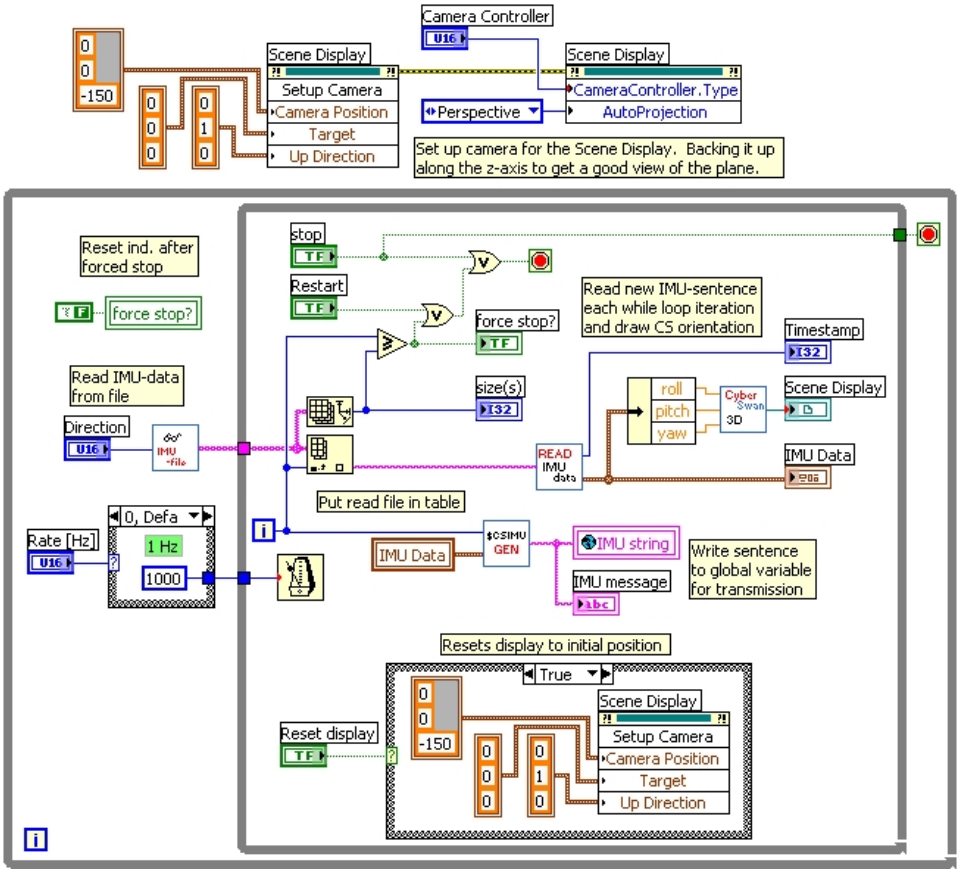


Figure 5.46: IMU simulator block diagram.

The Read IMU file subVI reads the chosen IMU log file and places the data in an array.

The Read IMU data subVI reads one IMU message string from the array, generated by the loaded IMU log file, indexed with the while loops iteration number. The message is then parsed and diplayed.

The $CSIMU generator subVI reads the IMU display and generates the $CSIMU sentence.

GPS

Figure 5.47 shows the GPS simulator's front panel. The simulator circles around an origin with a radius of approximately 250 meters. Then the information defined by the $CSGPS sentence are gathered and written to its display and the global variable accessable by the transceiver VI.

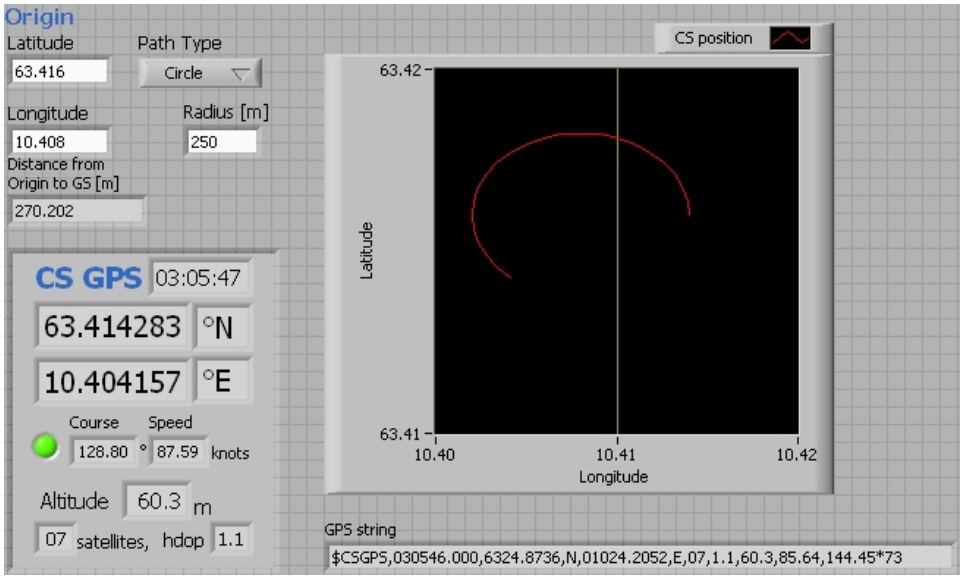


Figure 5.47: GPS simulator front panel.

Figure 5.48 shows the VI's block diagram. The inital GPS position are first calculated in the Initial path subVI. Then a circular path is calculated using the step 360° and CS path generator subVIs. This path is also plotted in a display on the front panel. The simulated CS's speed and course are calculated in the Calculate GPS speed and course subVI. The calculated values, along with a few constanst are then bundeled together and displayed in the CS GPS display on the front panel. From these values the $CSGPS string is generated with the $CSGPS generator.

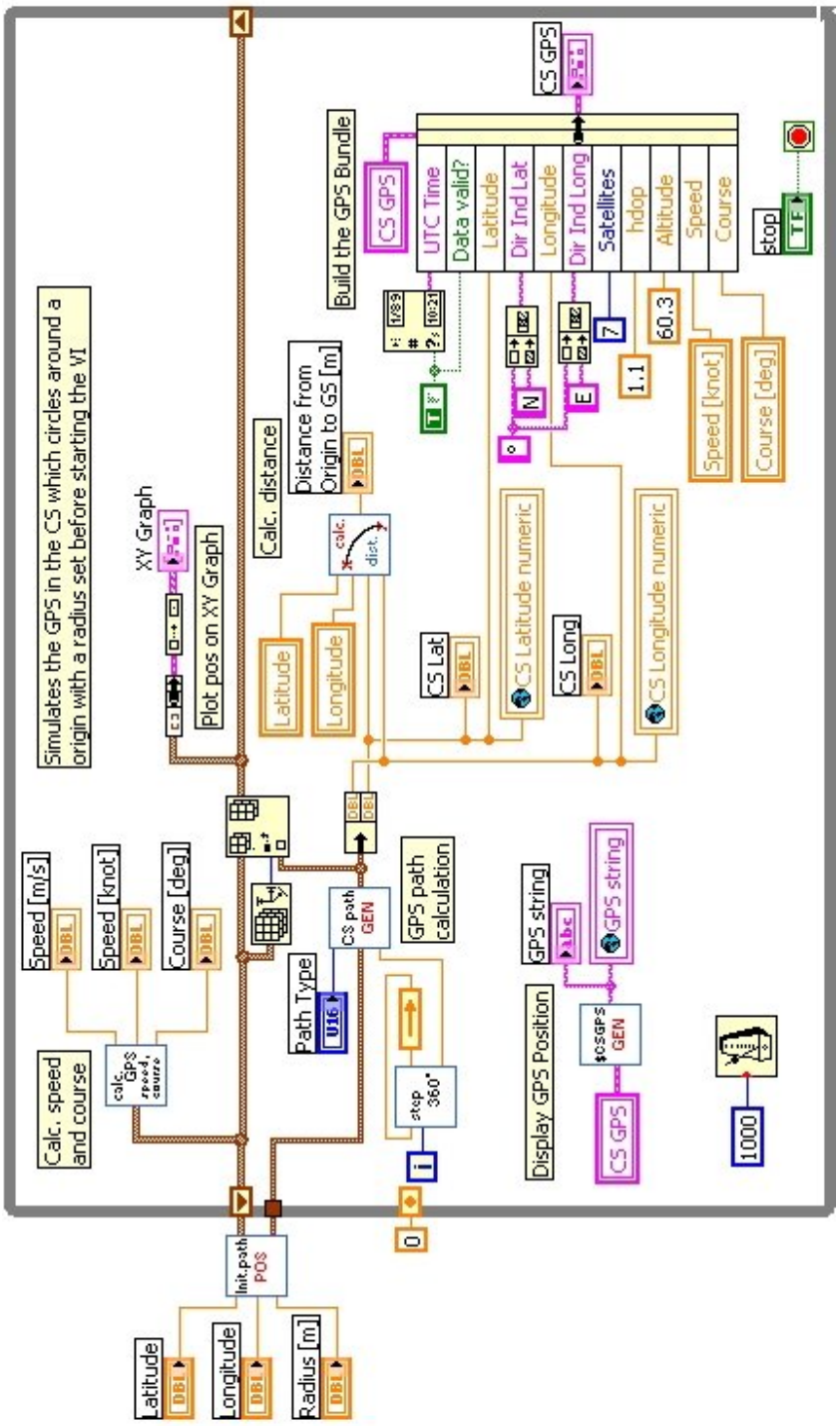


Figure 5.48: GPS simulator block diagram.

Servo

Figure 5.49 shows the Servo VI's front panel, and Figure 5.50 shows its block diagram. The servo values are generated by using random number generators, giving them a value between 0 and 255. Then the $CSSRV string is generated and displayed on the front panel, and written to the global variable accessable by the transceiver VI.

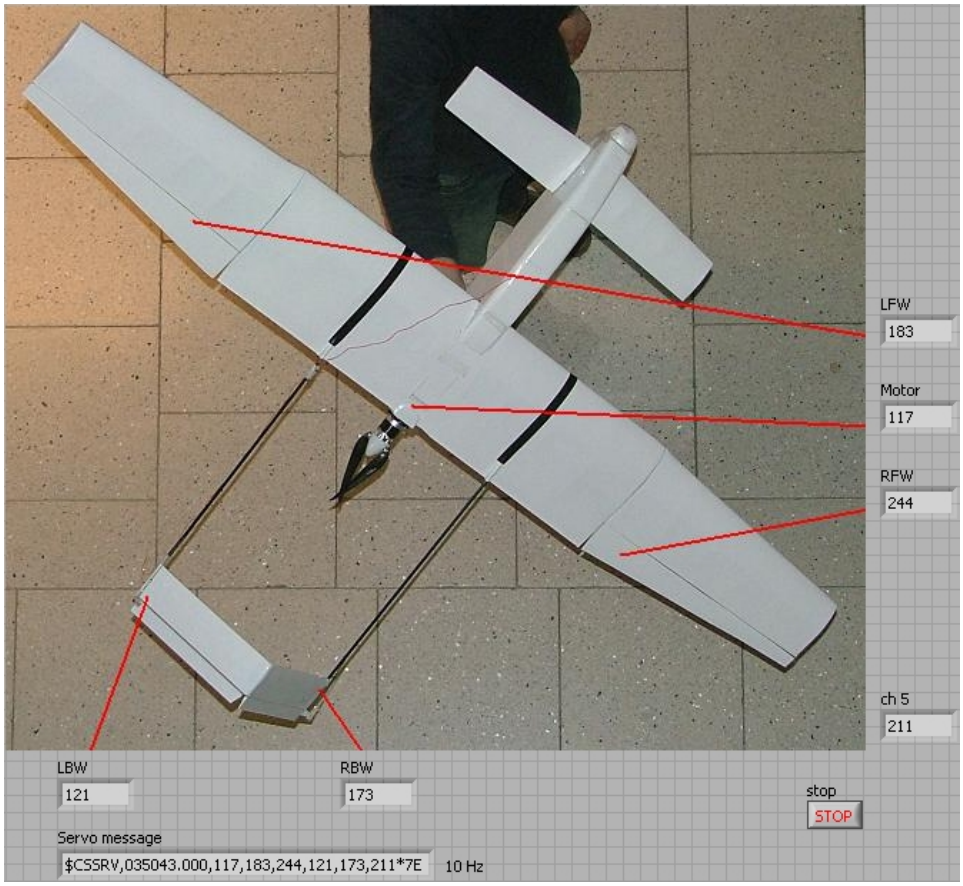


Figure 5.49: Servo simulator front panel.

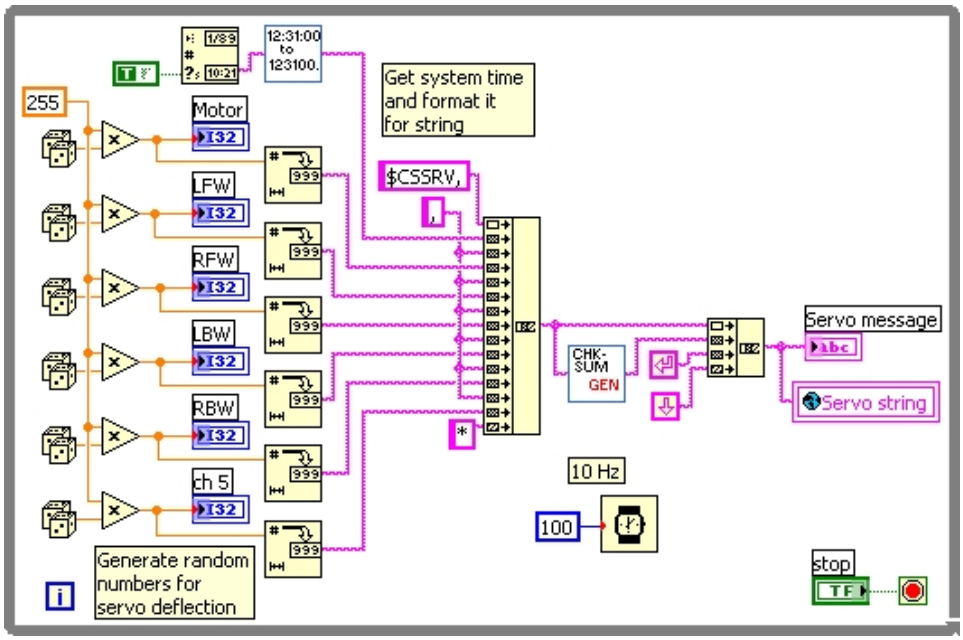


Figure 5.50: Servo simulator block diagram.

Ultrasound ranging device

There are three ways to get a range measurement from the MaxBotix LV-MaxSonar-EZ1. It has a pulse width output, analog voltage output and a serial port. The sensor is to be connected to the I/O-card connected to the CS's main computer. Since this has a limited amount of serial ports, the analog voltage option was taken.

The CS main computer system's I/O-card features an Atmel ATmega128 microcontroller (Bjørntvedt 2006). A routine to get a range measurement was programmed in C, using a built in analog-to-digital converter (ADC) to sample the analog voltage, representing the measured range. A test program, running the programmed range-routine once a second, was developed using the Atmel STK500 development board, with the STK501 add-on card which features the ATmega128. It was programmed with Programmers Notepad, provided with the WinAVR (2006)-package.

The test-program is enclosed in Appendix A.

Chapter 6

Results

Due to the unavailability of the CS's computer system (Bjørntvedt 2007) it was not possible for any in-flight test. A hardware in the loop (HIL) test was therefore performed using the CS simulator developed in this thesis, instead of the CS itself. In addition, a manual test of the ultrasound ranging device and a test of the camera system in-flight was performed.

6.1 HIL test

Test setup

Figure 6.1 shows the HIL test setup, consisting of the following hardware:

- Dell Inspiron Laptop Centrino 1.4 GHz with 512 MB of memory running the GS software
- Dell Optiplex GX620 Stationary PC Pentium 4 Hyper-Treading 3 GHz with 1 GB of memory running the CS simulator software
- Radiocrafts RC1240 demonstration board connected to the GS Laptop
- Developed CS RF card connected to the CS simulator PC

Figure 6.1: Hardware setup for HIL test.

Objective

The objective of the tests was to find the GS performance with regards of communication for presentation of data and navigation.

Test description

The following tests were performed:

1. Transmission of GPS correction sentence from the GS to the CS simulator.

2. Transmission of GPS position sentence from the CS simulator to the GS.

3. Simultaneous performance of test 1 and 2.

4. Identification of the maximum throughput of the IMU sentence from the CS simulator to the GS.

Each test, exept test 4, were performed for three minutes and repeated three times. The results are given with a percentage of how many messages that were lost.

Test 1

The results from test 1, transmission of GPS correction sentence, $GSPCO, from the GS to the CS simulator, are shown in Table 6.1.

Run	Message loss [%]
1	0
2	1.1
3	0

Table 6.1: Results from HIL test 1.

Test 2

The results from test 2, transmission of GPS position sentence, $CSGPS, from the CS simulator to the GS, are shown in Table 6.2. Here run four is a reference test using a null modem cable.

Run	Message loss [%]
1	4.4
2	6.1
3	5.0
4	5.0

Table 6.2: Results from HIL test 2.

Test 3

The results from test 3, simultaneous performance of test 1 and 2, are shown in Table 6.3. Run four is a reference test using a null modem cable.

Run	$CSGPS loss [%]	$GSPCO loss [%]
1	8.3	0
2	7.2	0
3	6.7	0.5
4	0	0

Table 6.3: Results from HIL test 3.

Test 4

Test 4, identification of the maximum throughput of the IMU sentence from the CS simulator to the GS, where performed using a terminal program and confirmed with the GS software.

In this test the CS simulator's transmit sequence is to be tuned to its maximum, limited by the data rate of the RF link. It is defined to take once second to send one message from one of the simulator's slot. That is, if for example one GPS sentence is ordered, is to be sent once a second, from one slot. This was tested using 3, 4 and 5 active slots. 3 slots gives a delay of 333 ms between the slots, 4 slots a delay of 250 ms between the slots, and 5 slots gives a delay of 200 ms between the slots.

The RF link were not able to transmit five consecutive IMU sentences in one second. The maximum throughput were found using 4 slots, giving a frequency of 4 Hz.

6.2 Ultrasound ranging device

The ultrasound ranging device, MaxSonar-EZ1, was manually tested using a mulimeter measuring the analoge voltage, representing the range to the ground. A surface similar to a surface used for landing was chosen: a lawn at the university campus. It gave a clear reading from 0 to 2.5 m. This is considered to be enough and it was not tested how the reading was close to its proposed range of 6.45 m it gave a reading.

6.3 Video test

Figure 6.2: Antenna for video reception.

Video from the wireless camera, fixed on the CS's tail, was recorded in-flight while performing tests on the CS's controllers. Figure 6.2 shows the antenna used for receiving and the video camera used for recording. The video, edited by Høstmark (2007), is enclosed in Appendix A.

Chapter 7

Discussion

This chapter is divided into four parts. First the results from the HIL tests are discussed. Further, the hardware choices, the software choice and system design are evaluated. At last are some recommendations for further work.

7.1 Results

The tests on the communication equipment showed some interesting characteristics. The first test, transmitting the $GPPCO sentence to the CS simulator was practically perfect, with loss of only two of 540 messages. Of course, the sentence is sent only once a second, and not using much of the capacity, one should expect the transmission to be flawless. This would also be expected from the second test, receiving the $CSGPS sentence on the GS. Since quite a few messages were lost, a test with a null modem cabel were also performed, giving similar results. This excludes the RF link of beeing the faulty factor. The cause of the errors were examined, and can be explained to be caused by a combination of transmission latency, timeout configuration and buffer problems. The errors that occured were due to that just a part of the sentence entered the read buffer, causing a checksum error. This was tested with a terminal program, receiving the same as the GS did. Here the whole sentence appeared, but some of them showed a small latency, where a part of the sentence appeared just a little bit later than the first part, exactly the way they failed in LabVIEW. The solution to this problem was to increase the timeout in the serial port configuration to 2 s from the

100 ms from before. Now all the sentences were received without any faults, but this just created a new fault as the GPS receiver became slower. Quite some time was spent examining the properties of the serial ports, and why they would influence each other, but no definitive reason was found. It was not pursued further, since the problem was discovered in a late stage of the project.

In test 3 multiple collisions were expected, due to no flow control, and transmission in both directions. Still, also here the faults were confined to the reception of the $CSGPS sentence. The interesting property here is that, in run 4, with the null modem cable, the result were perfect both ways. When the GS transceiver was both sending and receiving, the reception buffer problem from test 2 was removed. The problem were not found, but the results pointed to that resources were spent somewhere in the block diagram during reception only, slowing it down.

Test 4 resulted in a 4 Hz throughput, tested with the IMU sentence. The IMU display managed to follow the CS simulator's IMU display, only lagging behind due to the transmission latency.

It was, after reviewing these tests, that the RF link could be a successful implementation when connected to the CS computer system. But before this, the receiver problem in the GS's transciever would have to be adressed. In addition should hardware flow control concidered to be implemented.

7.2 Hardware choices

The hardware selection applies to the GPS receiver, the RF receiver, the wireless camera and the ultrasound ranger.

The GPS receiver had all the properties one would want, and have not posed any particular problems. The only complaint would be the poor documentation. The development of the circuit board progressed without any problems.

The ultrasound ranger was not tested during a landing situation, where it would be used in a autonomous mode. The CS's computer system was not finished in order for the ranger to be implemented on the I/O board. Still, the ranger was manually tested on a surface similar to a landing site, giving a clear range reading of at least 2.5 m. This gave an indication that the ranger could be used in an autonomous landing mode.

Concidering that the camera that was chosen, the Luda Minikameran, are made for home security, its quality of video are acceptable. Still, it is not close to the quality seen in Hi Cam's Pro X2 system example videos. This system have superior resolution and a noise free picture, while Minikameran's image are easily distorted. The Pro X2 also enables any video source to be connected. But since this system was unavailable Minikameran was the next best choice, and delivered unique in-flight images from the CS.

The RF receiver was chosen based on a few criteria: range, current consumption and data rate. The data rate was thought of beeing a little low, but the range was given priority, and the Radiocrafts RC1240 was therefore chosen. The development of the circuit board went well, exept an erroneous footprint, which was corrected using one short wire. The choice of communication device was not appraised to be an especially good choice during the project period, because of problems with errors and low throughput. Some problems with the configurations were discovered and corrected. And in the end, achieving 4 Hz with the IMU message was more than expected on a 4800 baud connection. It was also experimented with flow control, though not sucessfully implemented. Chosing RC1240 as the communication device was eventually proven to be a good one. Still, a higher data rate would be needed if there is a need to transfer data like IMU, servo information and height, at a higher combined rate than 4 Hz. Then the MaxStream modems, used by the Aerobotics Reaserch Group at Monash University (2007), could be investigated. A higher data rate would also be necessary if it is of interest to transfer other data, like digital photos, through the same communication device.

7.3 Software choice and system design

Realizing the GS in LabVIEW was chosen mostly to increase the chance of getting a result, a working GS. An influencing factor was also the fact that many control operations use LabVIEW. Having reached the goals set at the start, developing a functional product, made it difficult to contradict this choice. The alternative was to develope the software at a much lower level, and hereby lowering the possibilites to make a functional product.

It was focused to make the GS implementation structure easy and lucid. It uses sharing of information through global variables. This made it possible to develope the different functions needed individually, and test their functions, before putting them together as a whole program. This was a great

advantage making the development efficient. The transceiver unit is used to store all the information that is received, sent or ready to be sent, all available through global variables. This makes it possible for any function to use any information received, and easy to implement new functions.

It was decided to use LabVIEW to visualize the CS's position. Commercial options are available but traditionally based on the positioning of just one unit. In this case it was, in addition to the CS's position, also of interest to mark the GS's position. The map used in the implemented plotter is based on publicly available airial photos. The coordinates were manually entered, using the interactive map Norgesglasset (Statens Kartverk 2007). The map needed only minor adjustments in order for the coordinates to match. It does not support maps going across the datum line, and the plots and lines tickness are not adjustable. Still, this option worked good enough for its entended use. A cheap alternative could be to use Google Earth Pro. The Pro version has plotting possibilities, but this option does not yet have good enough resolution in Norway to make it a proper alternative.

Problems

There were not only advantages. LabVIEW uses a lot of resources, affecting the systems performance. The CS simulator had a 60 - 70 % processor load on a fairly new computer. The reason for this was the CS 3D model showing the logged IMU data, and the servo simulated values had 10 Hz update rate. Especially was the 3D model resource demanding, resulting it to sometimes lag behind, not always beeing able to work at the specified speed. This was not a major problem as this only happened when this panel was active, and all data was still available from the transceiver panel.

7.4 Recommendations for further work

The recommendations for further work can be divided into two categories: improvements to the developed system and additional work.

The developed system works but the communication lacks flow control. This is available in the chosen communication link, and the serial configuration in the software is equipped with this option. In order to implement this in the software, the transceivers structure should be evaluated and perhaps restructured. The problem regarding message reception should also be adressed.

The developed plotter has a few limitations. New maps have to be manually

developed, the plots and lines are difficult to see due to their size, and the current design does not support the crossing of datum lines. The plotter works, but is not concidered to be versatile. It is recommended to research for a third party plotter capable of implementation with the rest of the GS software.

Currently only the navigation route message demands an acknowledge, this should also be implemented for the GS position transmission. Implementation can be done in the same manner as the navigation route transmission structure, sent only when the GS is not engaged. It should be concidered if this is necessary for the *Order* message, as one can see what message is beeing received, and thereby can conclude if the message transmitted to the CS was received or not.

Chapter 8

Conclusion

The main objective in this thesis has been to develope a ground station (GS) for the CyberSwan (CS) fixed-wing UAV. In addition a CS simulator has been developed, in order to test the GS, due to unavailability of the real system. Two electronic circuits has been developed: a RF card for communication between the CS and the GS, and a GPS receiver board. Supporting equipment for the CS has also been collected: a wireless camera system and a ultrasonic ranging device.

Video have been transferred from a test flight, with acceptable quality. The GS and the CS simulator were developed in LabVIEW. A hardware in the loop (HIL) test was performed using these systems. The HIL test brought a reception problem in the GS to the surface. It was discovered that the timeout setting in the configuration of the two serial devices connected to the GS influenced each other, causing a loss of 5 % of the messages on the communication device. Transmission was not affected by this problem, nor was the CS simulator. It was not possible to solve this problem before the deadline.

Beyond the mentionded problem, the tests proved that the selected communication device has a data rate high enough for the intended use. A throughput of 4 Hz was successfully achieved, making it possible to follow the CS's state through the GS.

Appendix A

DVD

A.1 Contents

The attached DVD contains four folders:

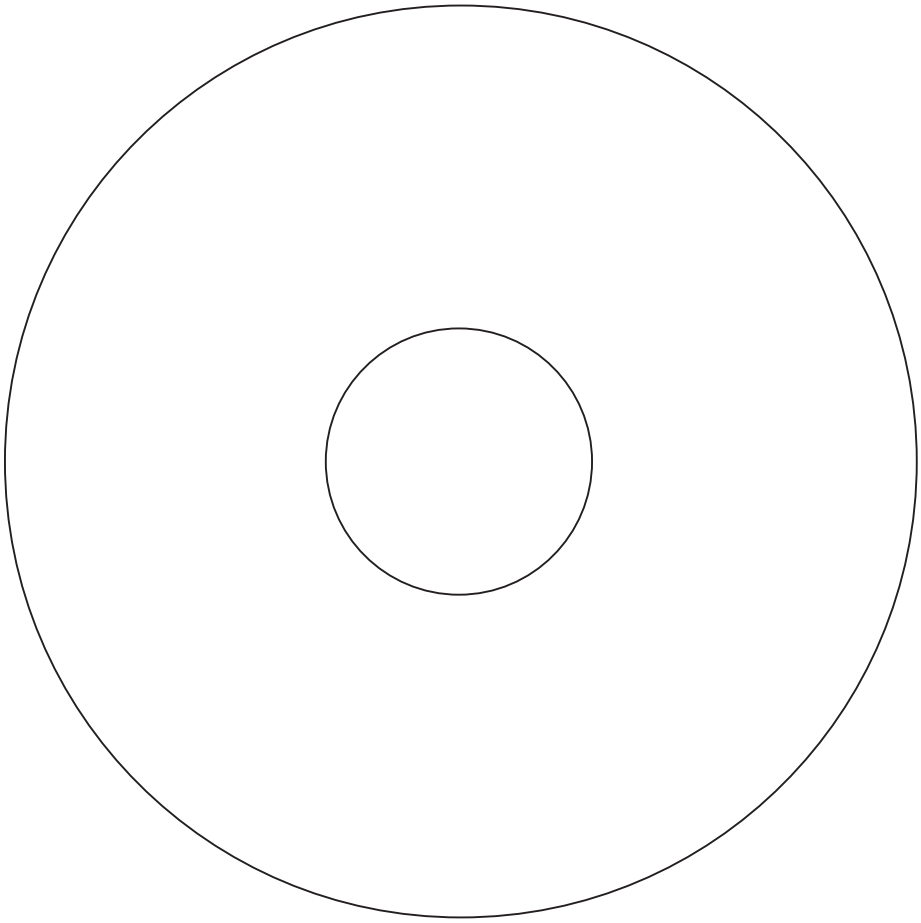
Development contains documentation of the developed hardware, the developed LabVIEW code with documentation, the software for the ultrasound ranging device and the IMU data collection program.

References contains references, datasheets and pdf documents of sources found on the Internet.

Report contains the LaTeX source of this report.

Video contains videos from the CS test flight, a quick tour of the GS and the CS simulator. One video from the collection of IMU data, and three examples of video received with the Hi Cam Pro X2 system (Hi Cam 2007) are also attached.

A.2 DVD



Bibliography

Baykar Machine Inc (2007), 'Bayraktar mini uav system', web page: www.baykarmakina.com.

Bjørntvedt, E. (2006), Instrumenteringsdesign for autonomt ubemannet fly, Technical report, Institutt for teknisk kybernetikk, NTNU.

Bjørntvedt, E. (2007), Instrumentering av selvstyrt ubemannet fly, Master's thesis, NTNU.

CadSoft Online (2006), 'Eagle v4.16 release 2', web page: www.cadsoft.de/index.htm.

Chamberlain, R. G. (2007), 'Great circle distance between 2 points', web page: www.movable-type.co.uk/scripts/gis-faq-5.1.html.

Deitel, H. M. & Deitel, P. (2001), *C++ How to Program*, 3rd edn, Prentice Hall.

Ellingsen, O.-J. (2002), Utvikling av et styresystem for et modellhelikopter (CyberEagle), Master's thesis, Institutt for teknisk kybernetikk, NTNU.

Environmental Studies (2007), 'Global Position System Precision Farming', web page: www.environmental-studies.de.

ESA (2005), 'EGNOS Fact Sheet 1: EGNOS explained'.

ESA (2007), 'European Space Agency', web page: http://www.esa.int/esaNA/egnos.html.

Evjen, P. M. (2006), 'New radio modules makes "Long range devices"'.

Franchi, P. L. (2006), 'Turkish army orders mini-uavs', *web page: www.flightglobal.com/articles/2006/08/08/208327/turkish-army-orders-mini-uavs.html* .

gps.gov (2007), 'Global Positioning System', web page:gps.gov.

Green Bay Professional Packet Radio (2007), 'Microstripline Analysis & Design', web page: http://my.athenet.net/~multiplx/cgi-bin/strip.main.cgi.

Hi Cam (2007), 'Pro X2 systems', web page: www.hicam.com.au/pro_x2.htm.

Høstmark, J. B. (2006), Design og konstruksjon av ubemannet fly for visuell overvåkning, Technical report, Institutt for teknisk kybernetikk, NTNU.

Høstmark, J. B. (2007), Modelling, Simulation and Control of Fixed-wing UAV: CyberSwan, Master's thesis, NTNU.

Monash University (2007), 'Aerobotics @ monash', web page: www.ctie.monash.edu.au/AEROBOTICS/.

National Instruments (2007), 'LabVIEW 8.2 20th Anniversary Edition'.

pnt.gov (2007), 'TheNational Space-Based Positioning, Navigation, and Timing (PNT) Executive Committee', web page:pnt.gov.

Sølvberg, A. (2007), Cyberbike, Master's thesis, NTNU.

Statens Kartverk (2007), 'Norgesglasset - interaktiv norgeskart', web page: ngis2.statkart.no/norgesglasset/default.html.

Trolltech (2007*a*), 'Qt customers', web page: troll.no/customers.

Trolltech (2007*b*), 'Qt features', web page: troll.no/products/qt/features.

Ventura-Traveset, J., Gauthier, L., Toran, F., de Lesthievent, C. & Bedu, J. (2005), 'EGNOS Status and Evolution ENC'.

WinAVR (2006), 'sourceforge.net/projects/winavr/'. version 20060421.

Wormley, S. (2007), 'Gps errors & estimating your receiver's accuracy', *web page: edu-observatory.org/gps/gps_accuracy.html* .

www.aero.org (2007), 'The Aerospace Corporation', web page: www.aero.org.