NTNU
Norwegian University of
Science and Technology

Navigational Assistance for Mini-ROV

Andreas Løberg Carlsen

Master of Science in Engineering Cybernetics
Submission date: July 2010
Supervisor: Jo Arve Alfredsen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

VideoRay is a miniature underwater vessel especially designed for search- and inspection operations. The vessel is remarkably small, low-weight (~5 kg) and considerably mobile in areas with low accessibility. A disadvantage associated with the unit's size is its susceptibility to external forces and cable pull. When combined with poor water visibility the task of underwater navigation becomes increasingly difficult. There is also cost- and space related limitations to the accessory equipment that can be mounted on the ROV.

However, recent development within MEMS-technology has resulted in a rich selection of small and relatively low-cost accelerometers, gyroscopes and compasses which can serve as components in low-end inertial navigation systems. This project aims to explore the possibilities as well as limitations associated with this type of technology as a basis for time-limited inertial navigation for the VideoRay.

The project involves the development of a small embedded system including sensors for inertial measurements (IMU), heading (compass), depth (pressure) together with a suitable communication interface that allows easy installation in the vessel and that will transmit real-time motion data to the surface. A control scheme should be developed that assists the ROV in e.g. executing pre-programmed search patterns from a known location on the sea bed. Testing should be performed to document system properties and the results should be discussed with respect to the technology's feasibility in the present application.

Assignment given: 01. February 2010
Supervisor: Jo Arve Alfredsen, ITK

Abstract

In this thesis a simple and low-cost aided inertial navigation system is presented which can be used to automate underwater search operations. The system is designed and implemented to be compatible with a VideoRay Pro 3 tethered mini-ROV. Inertial-, compass- and pressure measurements are collected with a dedicated embedded system and transferred to the surface for processing. System states are estimated, compared against a user specified pattern and used to generate path-following control inputs. The system is tested on a small scale and the results obtained through camera-based measurements show acceptable performance for short time intervals.

Preface

This thesis concludes my master's degree in Engineering Cybernetics at the Norwegian University of Science and Technology. I would first of all like to thank Jo Arve Alfredsen for giving me the opportunity to work with an actual ROV and participate in a real search and rescue operation. He has always offered good advice when I have needed it. The personnel at the institute's workshop has done a great job in designing and building the module housing. I would also like to thank my co-students for contributing with advice and providing a social, but productive working environment.

Finally I would like to thank Ellen Kristoffersen for her continuous support and superb cooking.

NTNU, Trondheim 12.07.2010

Andreas Løberg Carlsen

Contents

List of Figures

Nomenclature

ADC	Analog-to-Digital Converter
AFS	Application Flash Section
BFS	Bootloader Flash Section
CAN	Controller Area Network
ECEF	Earth Centered Earth Fixed reference frame
ECI	Earth Centered Inertial reference frame
Firmware	A set of MCU instructions stored in non-volatile memory. Instructions, execution order and input parameters govern the behavior.
FOG	Fiber Optic Gyros
GIB	GPS Intelligent Buoy
GNSS	Global Navigation Satellite System
GUI	Graphical User Interface
ICB	Integrated Control Box
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
ISA	Inertial Sensor Assembly
ISP	In-System Programmable
JTAG	Joint Test Action Group, programming/debugging interface
LED	Light-Emitting Diode
LOS	Line-of-Sight path-following

LSB	Least Significant Bit
MCU	Microcontroller Unit
MEMS	Micro-Electro-Mechanical Systems
MIPS	Million Instructions Per Second
OpenCV	Open Computer Vision Library
PCB	Printed Circuit Board
RCA	Phono plug (Derived from Radio Corporation of America)
RLG	Ring Laser Gyro
ROV	Remotely Operated Vehicle
RS-232	Recommended Standard 232, serial binary interface.
SMD	Surface Mount Devices
SPI	Serial Peripheral Interface
TDS	Tether Deployment System
TTL	Transistor - Transistor Logic
UART	Universal Asynchronous Receiver Transmitter

Chapter 1

Introduction

1.1 Motivation

A Remotely Operated Vehicle (ROV) is an underwater vehicle that receives power, control signals and other necessities through a multifunctional umbilical cable (often called a tether). Such a vessel enables a user to conduct underwater exploration, observation and recovery missions with ease, while reducing the risk of human injury. ROVs come in many sizes and with a range of different auxiliary equipment including video cameras, grippers, sonars and other measuring instruments.

During underwater missions the camera output is often the most important data to an operator. Manual control of the vehicle is completely dependant on the user being able to see what is happening around the ROV. If a certain feature or object is of interest, he or she must continuously control the ROV to compensate for target movement or forces acting on the target and/or ROV. If an operator wishes to perform a search pattern, he or she must navigate consistently in an often featureless environment while combating disturbances.

To assist the operator, an automated path-following system can be introduced based on relative or absolute positioning. The ROV can then be controlled to explore a specified pattern while the user can focus on the video feed.

Manually executed search patterns are tedious, even simple operations as following a straight line for a short period of time can be unsuccessful based on personal experience.

1.2 Previous Work

A positioning system is based on the acquirement and fusion of sensor measurements to produce the best possible estimate of a vessel's position. In surface applications, the use of a Global Navigation Satellite System (GNSS) is highly attractive when combined with e.g. an Inertial Measurements Unit (IMU). Such a combination allows for sub-meter accuracy and availability during satellite occlusion.

Signals from the GNSS are absorbed close to the water surface and cannot directly be used in an underwater application. There is an acoustic alternative, which is based on the same concept as a GNSS. Instead of basing range measurements on electromagnetic propagation speed, it is based on the speed of sound through water. Depending on the system implementation a set of acoustic transmitters, transponders and transducers will be distributed across the sea bed, vessel or water surface.

One such acoustic implementation is the GPS Intelligent Buoy (GIB) system where a number of floating or moored buoys are fitted with acoustic transponders that react to ROV transmissions. A buoy is wirelessly connected to a central control unit or connected by wire. Each unit has an accurate position reading from the on-board GPS and will transmit this information when an acoustic pulse is received. By calculating the time between transmission and arrival the distance to each buoy can be determined relative to the GPS datum. Such a solution is practical because the necessary equipment can be brought to the area of operation during a mission and removed subsequently.

When using an acoustic system together with auxiliary measurements the results can be very accurate as shown in (Whitcomb et al. 1999) and (Alcocer et al. 2007).

Dead Reckoning is an alternative to acoustic positioning which estimates vessel displacement based on a previously determined position and run-time measurements. Velocimeter-, pressure transducer-, accelerometer- and gyroscope data are fused together to periodically update the vessel's system state.

A system based on dead reckoning and inertial measurements (accelerations and rotation rates) can be referred to as an Inertial Navigation System (INS). Such a system requires less external equipment, but depends on precise and expensive sensors to achieve an accurate position estimate. By extending a pure INS with additional sensors the complete accuracy of the system will increase and/or compensate for lack of inertial measurement precision. The

Kongsberg Maritime HUGIN project fuses a range sensor types, even acoustic measurements: GPS data (acoustically transmitted from an accompanying surface vessel), underwater transponder positioning (UTP) and Doppler velocity logging (DVL) (Jalving et al. 2003). INS combined with laser-based vision system (Karras & Kyriakopoulos 2007), INS combined with DVL and sonar (Lee et al. 2004).

1.3 Contribution

An aided-INS is designed and implemented for an observer class VideoRay Pro 3 S mini-ROV. The system consists of a small number of sensors that measure acceleration and rotation rate in three dimensions, magnetic compass heading in three dimensions and absolute pressure. Each sensor can be considered low-cost as they are of Micro-Electro-Mechanical Systems (MEMS) technology and cost less than 10% of the VideoRay system in total.

An ordinary Kalman filter is designed to estimate ROV attitude based on the available measurements. Position is estimated on the basis of the filtered attitude and the assumption of an unbiased forward speed. The position is meant to be consistent for a limited amount of time. Depth is calculated from pressure readings.

A simple guidance system and controller is designed to follow a user specified path consisting of n waypoints with x, y and z coordinates.

1.4 Outline

The thesis is roughly divided into the following chapters:

Theoretical Background: Relevant theory is presented. The different reference frames are defined while inertial navigation and discrete Kalman filtering are briefly introduced.

VideoRay Pro 3 S: VideoRay components are presented together with specifications of interest.

Specifications and Requirements: Specifications and requirements are stated for the hardware, firmware, software, state estimatior and control algorithms.

Design: The embedded system is designed together with the user interface and controllers. A discrete state space model and Kalman filter parameters are determined.

Implementation: The complete system implementation is described; hardware components, PCB, GUI, software libraries, function calls and development tools.

Experiments: Subsystems are tested seperatley to observe their isolated performance. The complete path-following system is tested on different search paths and the behavior is recorded through camera based measurements. Results er presented and discussed.

Conclusion: Overall system performance is commented and several improvements are suggested.

Chapter 2

Theoretical Background

2.1 Reference Frames and Transformations

An earth-fixed reference frame is not inertial because of the Earth's rotation, but for low-speed vessels this rotation can be neglected. This assumption justifies the use of Newton's laws in modeling and control when such a frame is considered (Fossen 2002).

When analyzing the behavior of a vehicle it is convenient to define multiple reference frames and describe the vehicle's motion as the sum of frame transitions. The reference frames chosen in this application are orthogonal, right-handed and based on definitions in (Farrell & Barth 1998).

ECI Earth Centered Inertial. A proper inertial frame with origin o_i at the center of the Earth, x_i towards the *vernal equinox*, z_i along the rotation axis and y_i completing a right-handed system. Used in accurate terrestrial or extraterrestrial navigation.

ECEF Earth Centered Earth Fixed. The origin o_e coincides with the ECI origin and x_e is defined from the Earth center to the location: latitude $0°$, longitude $0°$. z_e is along the rotation axis and y_e completes the right-handed system.

T Tangent frame. Defined by a plane, tangent to the Earth's reference ellipsoid (WGS-84) with its origin o_t locked at a specified longitude and latitude. x_t is defined in the direction of geographic north, y_t towards east and z_t downwards, normal to the tangent plane. These three axes complete a right-handed coordinate system that will be used for local navigation.

BODY This frame is fixed to the vehicle's body and used in expressing the vehicle's position and attitude relative to the navigation frame. Its origin o_b is placed at a location that simplifies modelling or transitions between frames. In this application the origin coincides with the S-frame origin o_s. The three axes form a right-handed system where the x_b axis is directed from aft to for, y_b from port to starboard and z_b from top to bottom.

S Sensor frame. Acceleration- and rotation rate sensors are aligned along three (approximately) orthogonal axes. Orientation depend on how the IMU is mounted to the circuit board. The axes are defined in relation to the circuit plane with y_s in the same direction as y_t (across the board), while x_s (along the board) and z_s (normal to the board) are in the opposite direction of x_t and z_t, respectively. The origin o_s resides at the axes' intersection inside the IMU.

Attitude and position are described by the vectors $\mathbf{\Theta}_{bt} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^{\mathrm{T}}$ and $\mathbf{p}^t_{bt} = \begin{bmatrix}x_t & y_t & z_t\end{bmatrix}^{\mathrm{T}}$ respectively. The attitude consists of three Euler angles relative to T as shown in Figure 2.1; roll ϕ, pitch θ and yaw ψ. Position is defined as the BODY origin relative to the T origin (subscript bt) expressed in T (superscript t). Force $\mathbf{f}^b$, moment $\mathbf{m}^b$, linear velocity $\mathbf{v}^b_{bt}$ and rotational velocity $\boldsymbol{\omega}^b_{bt}$ denotes that the measurements are decomposed in BODY coordinates.

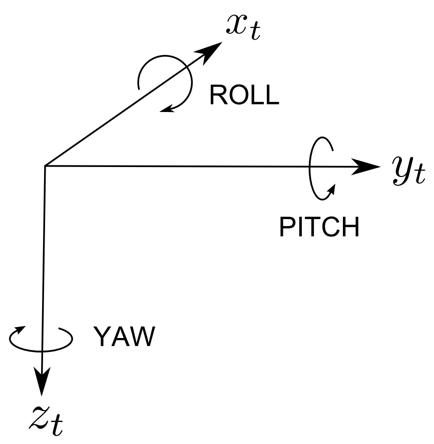


Figure 2.1: T reference frame. The positive direction of rotation is consistent with the right-hand rule.

Transformations between S and BODY

The transformation between the S- and BODY-frame is achieved through a rotation matrix based on Euler angles. These are denoted by $\mathbf{\Theta}_{sb} = \begin{bmatrix}\phi_b & \theta_b & \psi_b\end{bmatrix}^{\mathrm{T}}$ and indicate the attitude of S relative to BODY.

The rotation matrix is a product of three single rotations, one for each axis. These are multiplied in accordance with the *zyx*-convention producing a 3-by-3 orthogonal matrix that will transform a vector from S to BODY:

$$\mathbf{R}_s^b(\mathbf{\Theta}_{sb}) = \mathbf{R}_s^b = \mathbf{R}_{z,\psi_b}\mathbf{R}_{y,\theta_b}\mathbf{R}_{x,\phi_b}$$

$$\mathbf{R}_{z,\psi} = \begin{bmatrix}\mathrm{c}\psi_b & -\mathrm{s}\psi_b & 0\\ \mathrm{s}\psi_b & \mathrm{c}\psi_b & 0\\ 0 & 0 & 1\end{bmatrix}, \mathbf{R}_{y,\theta} = \begin{bmatrix}\mathrm{c}\theta_b & 0 & \mathrm{s}\theta_b\\ 0 & 1 & 0\\ -\mathrm{s}\theta_b & 0 & \mathrm{c}\theta_b\end{bmatrix}, \mathbf{R}_{x,\phi} = \begin{bmatrix}1 & 0 & 0\\ 0 & \mathrm{c}\phi_b & -\mathrm{s}\phi_b\\ 0 & \mathrm{s}\phi_b & \mathrm{c}\phi_b\end{bmatrix}$$

$$\mathbf{R}_s^b(\mathbf{\Theta}_{bs}) = \begin{bmatrix}\mathrm{c}\psi_b\mathrm{c}\theta_b & -\mathrm{s}\psi_b\mathrm{c}\phi_b + \mathrm{c}\psi_b\mathrm{s}\theta_b\mathrm{s}\phi_b & \mathrm{s}\psi_b\mathrm{s}\phi_b + \mathrm{c}\psi_b\mathrm{c}\phi_b\mathrm{s}\theta_b\\ \mathrm{s}\psi_b\mathrm{c}\theta_b & \mathrm{c}\psi_b\mathrm{c}\phi_b + \mathrm{s}\phi_b\mathrm{s}\theta_b\mathrm{s}\psi_b & -\mathrm{c}\psi_b\mathrm{s}\phi_b + \mathrm{s}\theta_b\mathrm{s}\psi_b\mathrm{c}\phi_b\\ -\mathrm{s}\theta_b & \mathrm{c}\theta_b\mathrm{s}\phi_b & \mathrm{c}\theta_b\mathrm{c}\phi_b\end{bmatrix}$$

For a further description of vehicle kinematics and rotation matrices refer to (Fossen 2002).

Transformations between BODY and T

The transformation between the BODY- and T-frame is similar to that of S and BODY, but uses the vehicle's attitude angles:

$$\mathbf{R}_b^t(\mathbf{\Theta}_{bt}) = \mathbf{R}_b^t = \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi}$$

2.2 Inertial Navigation

An INS uses Newton's laws of motion to calculate attitude and position. Based on how the sensor assembly is mounted an INS can be characterized as being a *gimbal* or *strapdown* system. A gimballed solution is mounted on a leveled platform oriented towards a desired heading (usually north). This configuration aligns the inertial sensors to a local navigation frame such that the measurements can be fed into an error model without prior transformations. The strapdown system is mounted directly to the vehicle's body such that inertial measurements must be transformed analytically with what is known as the *strapdown equations*.

The strapdown INS is smaller, cheaper and mechanically more simple, but requires the sensor assembly to have a larger measurement range (Vik 2009). The strapdown version will be the main focus of this section.

An INS can be defined as three nested systems as shown in Figure 2.2. The Inertial Sensor Assembly (ISA), consisting of accelerometers and gyroscopes, the IMU, a combination of the ISA and a low-level processing (e.g. ADC, down sampling of the measurements, coning/sculling compensation) and at the upmost level the INS that combines the IMU with the strapdown equations (Vik 2009).

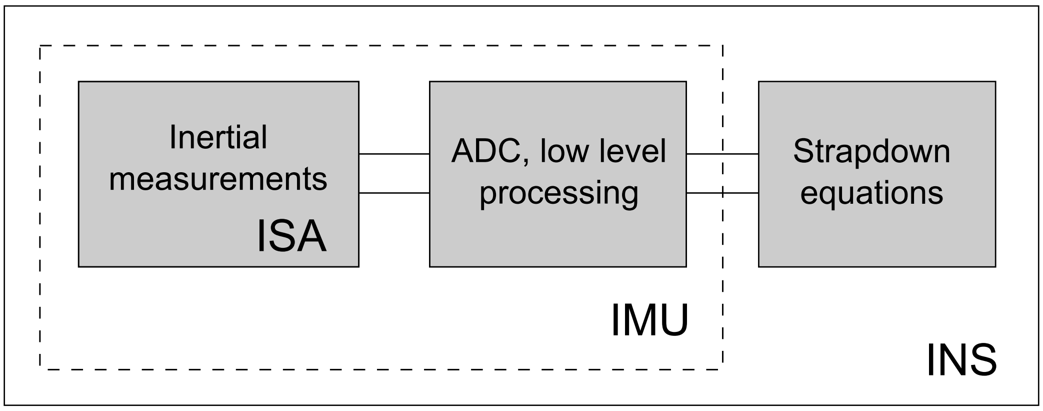


Figure 2.2: An overview of the INS subsystems. Adapted from (Vik 2009).

When used together with other instruments, the INS is often called an aided INS. If combined with a GNSS like Navstar GPS or GLONASS the result is a powerful tool in navigation. Measurements from the two systems are fused together and provide an accurate position and attitude. In case of a satellite outage, the IMU continues its operation and provides its own position estimate. Unfortunately, GNSS measurements are not directly available beneath sea level.

ISA

Accelerometers and gyroscopes can be based on several technologies, some more accurate than others. Ring Laser Gyros (RLG) and Fiber Optic Gyros (FOG) are accurate alternatives, but equally expensive. A gyro based on MEMS technology is the best alternative when a low-cost unit is required. These are usually based on the tuning fork gyro principle. A comparison of the three types is displayed in Table 2.1.

Parameter	**RLG**	**FOG**	**MEMS**
Input range (°/s)	> 1000	> 1000	> 1000
Bias (°/hr)	0.001-10	0.01-50	10-3600
Scale-factor error (%)	0.0001-0.01	0.0002-0.5	0.5-2
Bandwidth (Hz)	500	> 200	> 100

Table 2.1: Performance characteristics for different gyroscope technologies (Vik 2009).

Acceleration measurements are mainly based on two different principles. One such principle measures the amount of electricity needed to suspend a mass at a fixed position using electromagnets. The other principle is based on the frequency shifts of a vibrating string that experience a change in tension. These methods have successfully been ported to MEMS technology. A comparison is made in Table 2.2.

Parameter	**Closed loop pendulum**	**Vibrating quartz**	**MEMS**
Input range (g)	±100	±200	±100
Bias (mg)	0.1-10	0.1-1	< 25
Scale-factor error (%)	0.1	0.01	0.5-2
Bandwidth (Hz)	400	100	400
Threshold (μg)	10	< 10	1-10

Table 2.2: Performance characteristics for different accelerometer technologies (Vik 2009).

IMU

Depending on the technology, the size and weight of an IMU may vary. As a comparison, Figure 2.3 shows an expensive, FOG gyro, MEMS accelerometer IMU while Figure 2.4 shows a low-cost, all-MEMS IMU.

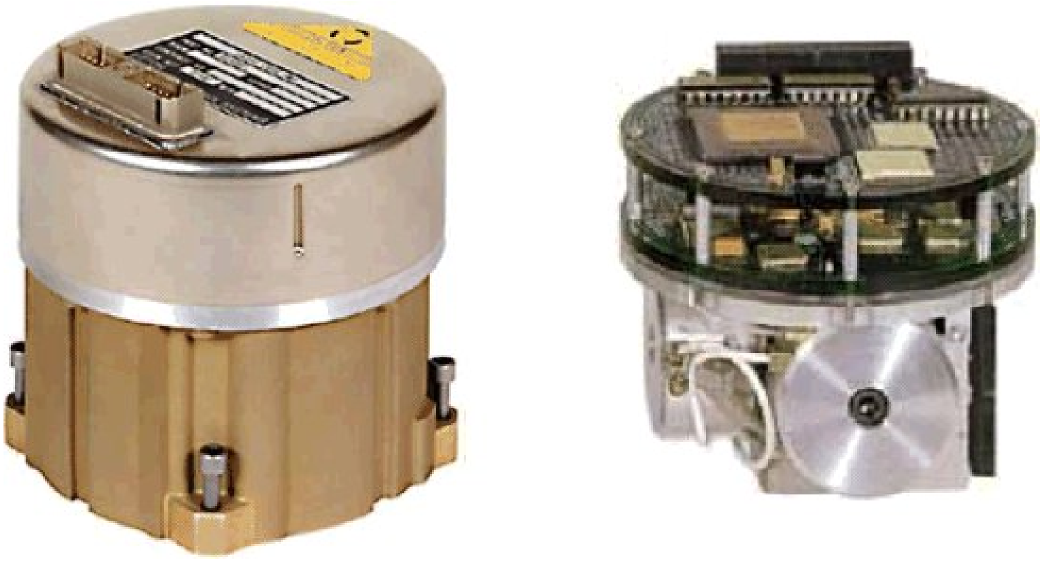


Figure 2.3: Northrop Grumman LN-200 IMU, (D 8.89 x H 8.51 cm) @ 750 grams.

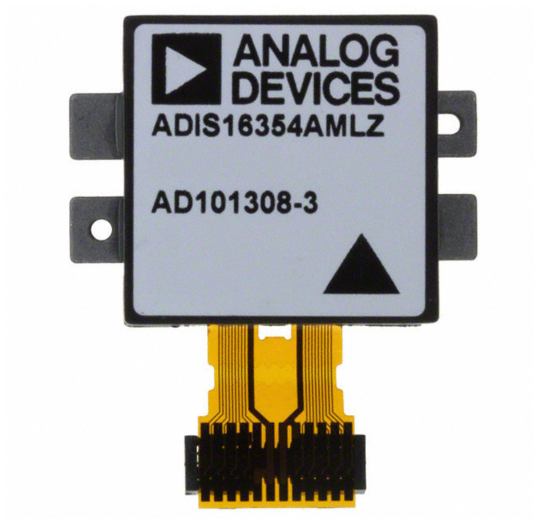


Figure 2.4: Analog Devices ADIS16354AMLZ IMU, (2.3 x 2.3 x 2.3 cm) @ 16 grams.

Strapdown Equations and Error Models

The strapdown equations are used to transform acceleration measurements into the local navigation frame. Gravity compensation is necessary because the IMU senses the vehicle's acceleration superimposed on gravitational acceleration. The presented equations take a number of factors into account, some less important than others when considering low-cost equipment. Equation (2.1) is derived in (Vik 2009).

$$\begin{aligned}\dot{\mathbf{v}}^t &= \mathbf{R}_b^t\mathbf{f}^b + \mathbf{g}^t - [2\mathbf{S}(\boldsymbol{\omega}_{ei}^t) + \mathbf{S}(\boldsymbol{\omega}_{te}^t)]\mathbf{v}^t \\ \dot{z}_t &= \mathbf{c}^{\mathrm{T}}\mathbf{v}^t \\ \dot{\mathbf{R}}_e^t &= -\mathbf{S}(\boldsymbol{\omega}_{te}^t)\mathbf{R}_e^t\end{aligned} \tag{2.1}$$

$\mathbf{R}_b^t\mathbf{f}^b$	Force transformed from BODY to T $\in \mathbb{R}^3$
$\mathbf{g}^t$	Gravity vector (G-vector) $\in \mathbb{R}^3$
$2\mathbf{S}(\boldsymbol{\omega}_{ei}^t) + \mathbf{S}(\boldsymbol{\omega}_{te}^t)$	Coriolis effect and centripetal force $\in \mathbb{R}^{3\times 3}$
$\boldsymbol{\omega}_{ei}^t$	Earth's rotation rate $\in \mathbb{R}^3$
$\boldsymbol{\omega}_{te}^t$	Rotation rate of T relative to ECEF $\in \mathbb{R}^3$
$\mathbf{c}$	$\begin{bmatrix}0 & 0 & -1\end{bmatrix}^{\mathrm{T}}$

The IMU measurements do not reflect the true behavior of the vehicle because of sensor imperfections. To compensate for this the different sensors can be modelled with several error terms; biases, scale-factors, higher order terms and misalignment angles. The relationship between sensor input and output can be expressed as a polynomial:

$$f_{\text{out}} = f_{\text{in}} + c_0 + c_1 f_{\text{in}} + c_2 f_{\text{in}}^2 + \cdots$$

The bias c_0 and scale-factor c_1 are depicted in Figure 2.5. Depending on an application's accuracy the number of modelled factors will vary. The gyroscope- and accelerometer output from the IMU can be modelled as Equation 2.2 and 2.3 hereby known as the *IMU error model.* All error signals are decomposed in BODY.

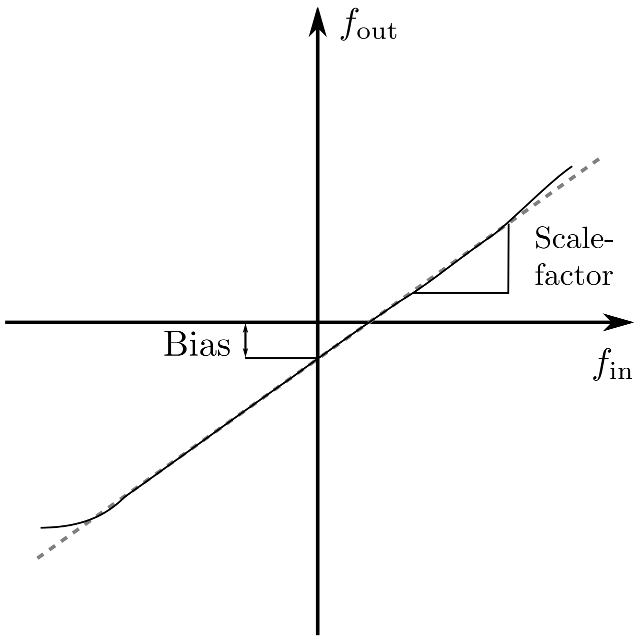


Figure 2.5: Definition of error in gyroscopes and accelerometers. The perfect sensor response would be a straight line with slope $+1$ going through the origin (Vik 2009).

$$\begin{aligned}
\boldsymbol{\omega}_{bi}^{b} &= [\mathbf{I} + \boldsymbol{\Delta}(\boldsymbol{\kappa}, \boldsymbol{\alpha})]\boldsymbol{\omega}_{\text{imu}} + \mathbf{b}_g + \mathbf{w}_1 \\
\dot{\mathbf{b}}_g &= -\mathbf{T}_1^{-1}\mathbf{b}_g + \mathbf{w}_2 \\
\dot{\boldsymbol{\kappa}} &= -\mathbf{T}_2^{-1}\boldsymbol{\kappa} + \mathbf{w}_3 \\
\dot{\boldsymbol{\alpha}} &= -\mathbf{T}_3^{-1}\boldsymbol{\alpha} + \mathbf{w}_4
\end{aligned} \tag{2.2}$$

$$\begin{aligned}
\mathbf{f}^{b} &= [\mathbf{I} + \boldsymbol{\Delta}(\boldsymbol{\epsilon}, \boldsymbol{\beta})]\mathbf{f}_{\text{imu}} + \mathbf{b}_a + \mathbf{w}_5 \\
\dot{\mathbf{b}}_a &= -\mathbf{T}_4^{-1}\mathbf{b}_a + \mathbf{w}_6 \\
\dot{\boldsymbol{\epsilon}} &= -\mathbf{T}_5^{-1}\boldsymbol{\epsilon} + \mathbf{w}_7 \\
\dot{\boldsymbol{\beta}} &= -\mathbf{T}_6^{-1}\boldsymbol{\beta} + \mathbf{w}_8
\end{aligned} \tag{2.3}$$

where

$$\boldsymbol{\Delta}(\mathbf{s}, \boldsymbol{\phi}) = \begin{bmatrix} s_x & \phi_{xy} & \phi_{xz} \\ \phi_{yx} & s_y & \phi_{yz} \\ \phi_{zx} & \phi_{zy} & s_z \end{bmatrix}$$

$\boldsymbol{\omega}_{bi}^{b}$	Rotation rates of the vehicle's BODY relative to ECI $\in \mathbb{R}^3$
$\mathbf{f}^b$	Forces of the vehicle's BODY relative to ECI $\in \mathbb{R}^3$
$\mathbf{I}$	Identity matrix $\in \mathbb{R}^{3\times 3}$
$\boldsymbol{\omega}_{\text{imu}}^{b}, \mathbf{f}_{\text{imu}}^{b}$	IMU measurements $\in \mathbb{R}^3$
$\boldsymbol{\kappa}, \boldsymbol{\epsilon}$	Scale-factor errors $\in \mathbb{R}^3$
$\boldsymbol{\alpha}, \boldsymbol{\beta}$	Misalignment angles $\in \mathbb{R}^6$
$\mathbf{b}_g, \mathbf{b}_a$	Gyroscope and accelerometer biases $\in \mathbb{R}^3$
$\mathbf{T}_{1,\ldots,8}^{-1}$	Time constants, diagonal $\in \mathbb{R}^{3\times 3}$, $\mathbb{R}^{6\times 6}$ and $\mathbb{R}^3$, $\mathbb{R}^6$
$\mathbf{w}_{1,5}$	Bounded and unmodeled errors and measurement noise $\in \mathbb{R}^3$
$\mathbf{w}_{2,3,4,6,7,8}$	Gaussian white noise $\in \mathbb{R}^3$ and $\mathbb{R}^6$

An *INS error model* is used to describe the overall position,- velocity- and attitude error of the INS states. Such a model is used to correct the measurements before integration. The model described here is referred to as the *psi-angle error model* and is presented in (Leondes 1963).

$$\begin{aligned}
\delta\dot{\mathbf{v}}^t &= -\mathbf{S}(2\boldsymbol{\omega}_{ei}^t + \boldsymbol{\omega}_{te}^t)\delta\mathbf{v}^t - \mathbf{S}(\boldsymbol{\psi})\mathbf{R}_b^t\mathbf{f}_{\text{imu}} + \delta\mathbf{g}^t + \mathbf{R}_b^t\boldsymbol{\nabla} \\
\delta\dot{\mathbf{p}}^t &= -\mathbf{S}(\boldsymbol{\omega}_{te}^t)\delta\mathbf{p}^t + \delta\mathbf{v}^t \\
\dot{\boldsymbol{\psi}} &= -\mathbf{S}(\boldsymbol{\omega}_{ti}^t)\boldsymbol{\psi} + \mathbf{R}_b^t\boldsymbol{\gamma}
\end{aligned} \tag{2.4}$$

$\delta\mathbf{v}^t$	Velocity error $\in \mathbb{R}^3$
$\delta\mathbf{p}^t$	Position error $\in \mathbb{R}^3$
$\boldsymbol{\psi}$	Attitude error $\in \mathbb{R}^3$
$\boldsymbol{\nabla}, \boldsymbol{\gamma}$	Scaling and misalignment errors of gyro and accelerometer $\in \mathbb{R}^3$
$\delta\mathbf{g}^t$	Gravity errors; gravity anomaly and deflections of the vertical $\in \mathbb{R}^3$

INS Alignment

When initializing an INS the attitude must be determined such that the system has a valid initial state. Such a procedure is usually performed when the system is at rest.

Gravity is used to find the roll and pitch angles by measuring the G-vector along each accelerometer axis. The sensed forces are denoted f_x, f_y and f_z. By using trigonometry, the direction of the vector can be determined and conversely the roll/pitch angles relative to the T-frame:

$$\phi = \text{atan2}(f_y, f_z) \tag{2.5}$$

$$\theta = \text{atan2}(-f_x, \sqrt{f_z^2 + f_y^2}) \tag{2.6}$$

where atan2 is an implementation of the arctan operator with output in all four quadrants of the unit circle.

The Earth's rotation can be used to determine the yaw angle relative to north. This requires a highly accurate set of gyroscopes that can measure a rotation rate of $7292115 \cdot 10^{-11}$ rad/s. A compass is a low-cost alternative.

When the attitude is approximately known the gyro measurements can be transformed to T by compensating for roll and pitch. $\mathbf{R}_s^b$ is used with yaw equal to zero (* denotes corrected rotation rate):

$$\begin{bmatrix} \omega_x^* \\ \omega_y^* \\ \omega_z^* \end{bmatrix} = \mathbf{R}_s^b \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2.7}$$

The initial attitude alignment is considered a coarse, first step. By using a Kalman filter combined with an IMU error model, a finer alignment can be achieved. To estimate roll and pitch errors the filter uses the difference between gyro- and accelerometer based angles as a measurement. Equivalently the yaw error measurement is found by taking the difference between the gyro based angle and e.g. a compass reading.

The operations involved in attitude and position determination can be summarized in the following steps:

1. Coarse align INS
2. Integrate measurements
3. Estimate errors (fine alignment)

4. Correct measurements
5. Transform measurements

2.3 The Magnetic Compass

A magnetic compass indicates north by aligning itself with the surrounding magnetic field. Such a field is ideally only influenced by the Earth's magnetic field, but is in reality disturbed by a number of objects that induce their own or manipulate others. Such influences are categorized as being hard- or soft iron effects, respectively.

When mounting a compass to a rigid body, the neighboring components and parts will alter the sensed field as a function of mounting distances and body orientation. This is based on how field intensity weakens with distance and how the soft iron effect varies with its surrounding field. Such disturbances must be removed to obtain valid measurements.

In the interest of navigation the magnetic north pole is not coincident with the geographic. The deviation between the two changes slowly, and varies from location to location. An angle of deviation can be entered to a navigation system before an operation.

The resultant field can be measured electronically by a magnetometer. By orthogonally aligning three such sensors the heading can be determined at any roll- and pitch angle.

2.4 Pressure-based Depth Measurements

Pressure-based depth measurements are related to the weight of a column filled with fluid. The height of the column is determined by measuring the pressure difference between the column's top/bottom and combining this information with the specific weight of the fluid. Hydrostatic pressure in seawater can be related to depth with the formula (White 1998, p.65) :

$$p = p_{\text{atm}} + \gamma z_t$$

p	Pressure calculated in ${}^{\mathrm{N}}/_{m^2}$(Pa)
p_{atm}	Surface atmospheric pressure
γ	Specific weight of fluid. Seawater: $\gamma = 10050\, {}^{\mathrm{N}}/_{m^3}$

2.5 The Discrete Kalman Filter

The Kalman filter is an optimal recursive data processing algorithm. Since its publication in 1960 by Rudolf E. Kalman, it has become an invaluable tool in navigation and general state estimation. The filter combines a set of noisy measurements and generates an output that minimizes the estimation error. This is done on the basis of a blending factor $\mathbf{K}$ that is optimally updated in each iteration.

The filter uses a linear, discrete state space model during its calculations. Subscript $_k$ denotes value at time t_k:

$$\begin{aligned} \mathbf{x}_{k+1} &= \boldsymbol{\phi}_k \mathbf{x}_k + \boldsymbol{\Delta}_k \mathbf{u}_k + \mathbf{w}_k \\ \mathbf{z}_k &= \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \end{aligned} \tag{2.8}$$

When initializing the algorithm, the noise covariance of $\mathbf{w}_k$ and $\mathbf{v}_k$ is specified. These values indicate the reliability of a state and are used during filter updates. The algorithmic steps are summarized in Figure 2.6.

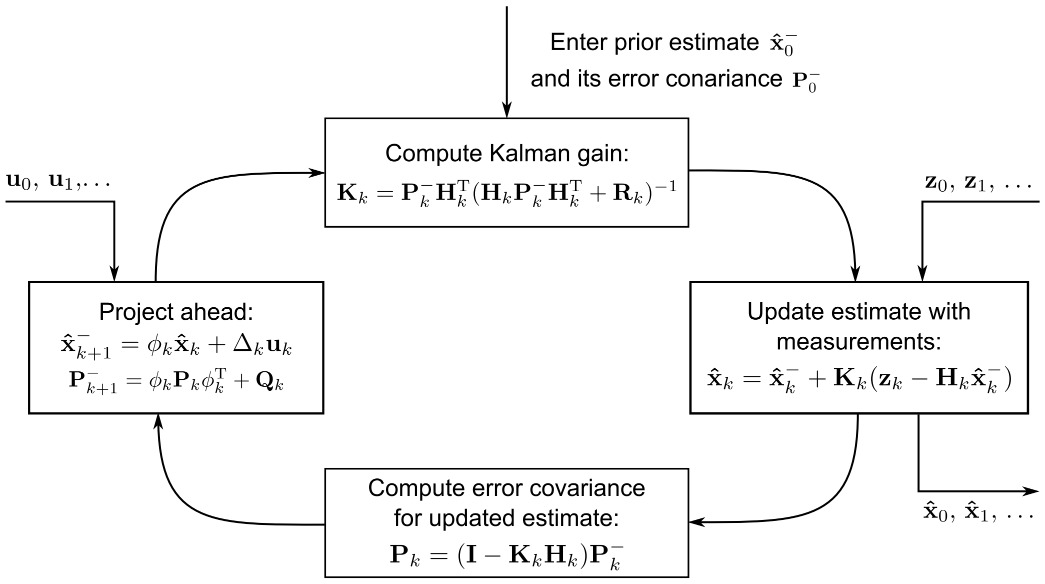

Enter prior estimate $\hat{\mathbf{x}}_0^-$
and its error conariance $\mathbf{P}_0^-$
Compute Kalman gain:
$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^{\mathrm{T}} (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}$
$\mathbf{u}_0, \mathbf{u}_1, \ldots$
$\mathbf{z}_0, \mathbf{z}_1, \ldots$
Project ahead:
$\hat{\mathbf{x}}_{k+1}^- = \phi_k \hat{\mathbf{x}}_k + \Delta_k \mathbf{u}_k$
$\mathbf{P}_{k+1}^- = \phi_k \mathbf{P}_k \phi_k^{\mathrm{T}} + \mathbf{Q}_k$
Update estimate with measurements:
$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-)$
Compute error covariance for updated estimate:
$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^-$
$\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, \ldots$

Figure 2.6: The Kalman filter loop. Prior error estimates, error covariance and measurements are the filter input while the estimated states are the output (Brown & Hwang 1997).

$\mathbf{x}_k$	Process state vector.
$\boldsymbol{\phi}_k$	State transition matrix.
$\mathbf{u}_k$	Control input vector.
$\boldsymbol{\Delta}_k$	Input matrix.
$\mathbf{w}_k$, $\mathbf{v}_k$	Process- and measurement white noise vector.
$\mathbf{z}_k$	Measurement vector.
$\mathbf{H}_k$	Measurement matrix.
$\hat{\mathbf{x}}_k$	Estimated process state vector.
$\mathbf{K}_k$	Kalman gain, blending factor.
$\mathbf{Q}_k$, $\mathbf{R}_k$	Process- and measurement noise covariance matrix.
$\mathbf{P}_k^-$, $\mathbf{P}_k$	A priori and a posteriori error covariance matrix.

To run the discrete Kalman filter on a continuos process, the system matrices must be discretized. A procedure is described in (Chen 1999) where ΔT denotes the sample time of the discretization:

$$\boldsymbol{\phi}_k = e^{\mathbf{A}\Delta T} \qquad \boldsymbol{\Delta}_k = \left(\int_0^{\Delta T} e^{\mathbf{A}\tau}\, d\tau \right) \mathbf{B} \qquad \mathbf{H}_k = \mathbf{C} \tag{2.9}$$

where A, B and C are the continuos time equivalents of the transition-, input- and measurement matrix respectively:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{w} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{v} \end{aligned} \tag{2.10}$$

One way of calculating $e^{\mathbf{A}\Delta T}$ is via the inverse Laplace transform:

$$e^{\mathbf{A}\Delta T} = \mathcal{L}^{-}(s\mathbf{I} - \mathbf{A})^{-1} \tag{2.11}$$

Chapter 3

VideoRay Pro 3 S

The VideoRay Pro 3 S mini-ROV system is a commercial product developed by VideoRay LLC located in Phoenixville, Pennsylvania, USA. Since the company was founded in 1999, it has become a global supplier of mini-ROV units.

The VideoRay system owned by the Department of Engineering Cybernetics is the same as the one shown in Figure 3.2. It consists of the mini-ROV unit, the ICB and the Tether Deployment System (TDS). These components are connected such that the ROV and ICB communicate through the tether housed in the TDS.

Actuators are placed so that the ROV can move in the surge, heave and yaw direction as seen in Figure 3.1. This is achieved with three bidirectional thrusters: one placed in a vertical tunnel, one placed on the starboard side and one placed on the port side.

Two cameras are mounted inside the ROV and have the specifications given in Table 3.1 (VideoRay 2010).

	Front Camera	**Rear Camera**
Lines of Resolution	570	430
Illuminance	0.3 lux	0.1 lux
Color Encoding	PAL	Black & White

Table 3.1: Camera specifications.

The frontal camera has the ability to tilt within a 180° vertical field of view and can be adjusted within a wide focus range. Camera exposure is

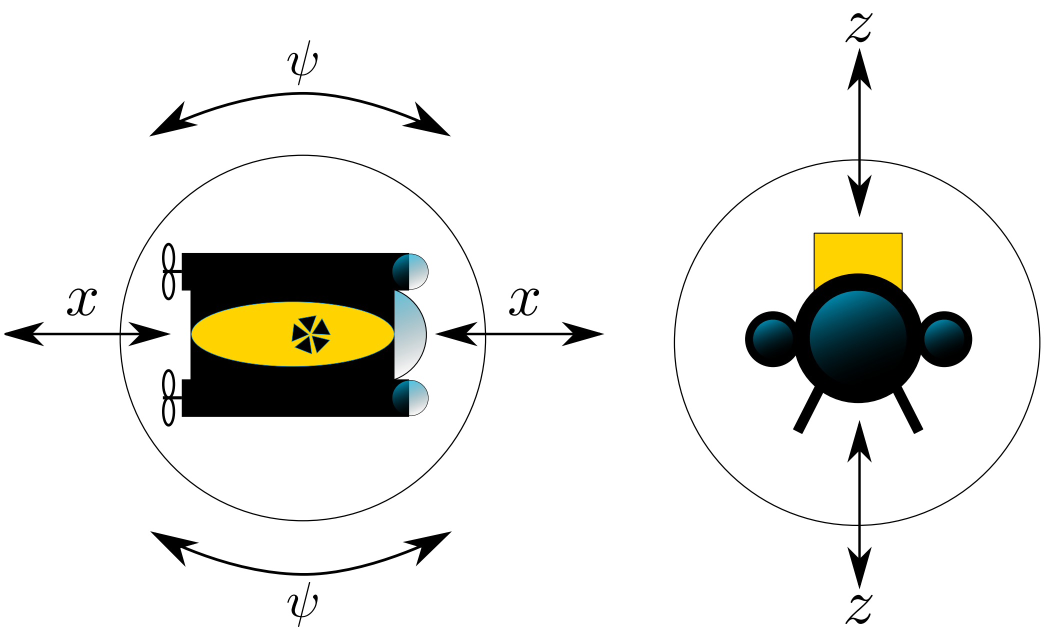


Figure 3.1: Surge(x), yaw(ψ) and heave(z) relative to the ROV.

automatically controlled. Captured video is sent to the ICB through the tether.

Actuators and cameras can all be controlled with the ICB dials, switches and joystick. Optionally, control signals can be given from an external source through a serial connection located on the inside of the ICB.

ROV ratings are presented in Table 3.2.

For more technical details, features and maintenance instructions, refer to the user manual shipped with the VideoRay system.

A mathematical model has been developed for the VideoRay ROV in (Miskovic et al. 2007).

Depth Rating	152.4 m (500 ft)
Tether Length	76.2 m (250 ft)
Dimensions (LxWxH)	30.48 cm x 22.5 cm x 22.9 cm (12" x 8 – 7/8" x 9")
Length Between Thrusters	17 cm (6.7")
Weight	3.6 kg (8 lb)
Operating Voltage	100 V – 240 V @ 50 Hz – 60 Hz AC
Internal Voltage	48 V DC Maximum
Speed	0 knots to 2.5 knots
Operating Temperature	0°C – 50°C (32°F – 122°F)
Medium	Fresh or Salt Water (ROV cleaning is recommended after use in salt water)

Table 3.2: ROV specifications.

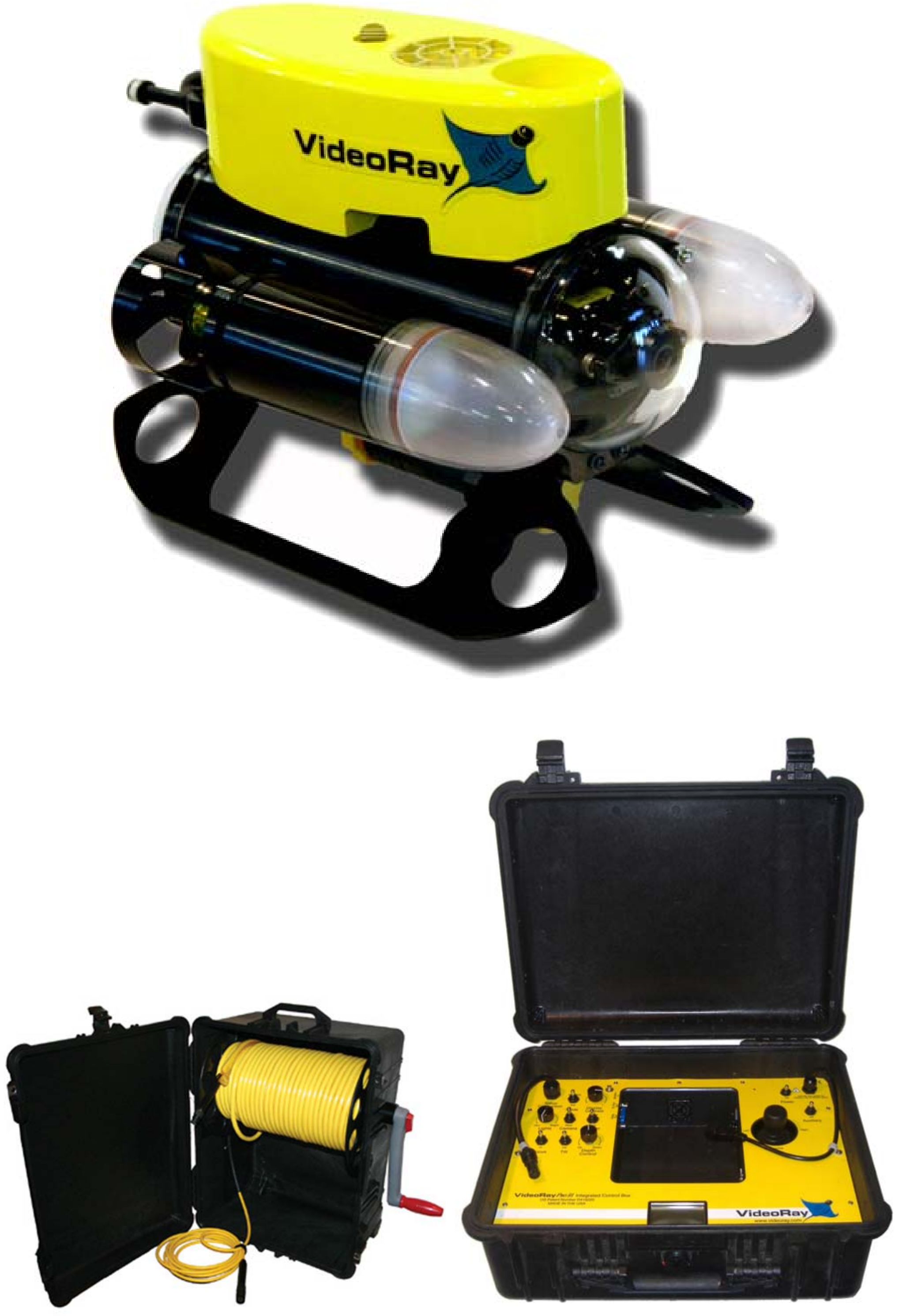


Figure 3.2: Top: VideoRay Pro 3 S ROV. Bottom-left: Tether Deployment System. Bottom-right: Integrated Control Box. VideoRay LLC ©videoray.com

Chapter 4

Specifications and Requirements

4.1 Hardware

The complete path-following system is based on the VideoRay ROV's ability to receive external control signals. Thrusters, lights and cameras can all be controlled through an RS-232 interface, giving the operator much freedom in implementing desired control features.

The hardware setup (Figure 4.1) consists of four modules and their interconnections. Sensor data, video and the VideoRay's internal voltage are all available through the tether, supplying the ROV and navigation module with power while making data available at the ICB. These signals are interfaced to the computer, processed and used in the calculation of new ROV control inputs. Control signals are transmitted back to the ICB and formatted as ROV input.

4.1.1 Embedded System

Raw sensor data must be gathered, preprocessed and packaged before tether transmission. This work can be done by a Microcontroller Unit (MCU) with the sufficient number of input/output pins and the necessary processing power.

The size of the embedded system must be limited to fit inside a housing that can be mounted on the ROV's belly. Housing volume is proportional to the Printed Circuit Board (PCB) area and will create buoyancy forces that will affect ROV properties. A small PCB layout is therefore desirable.

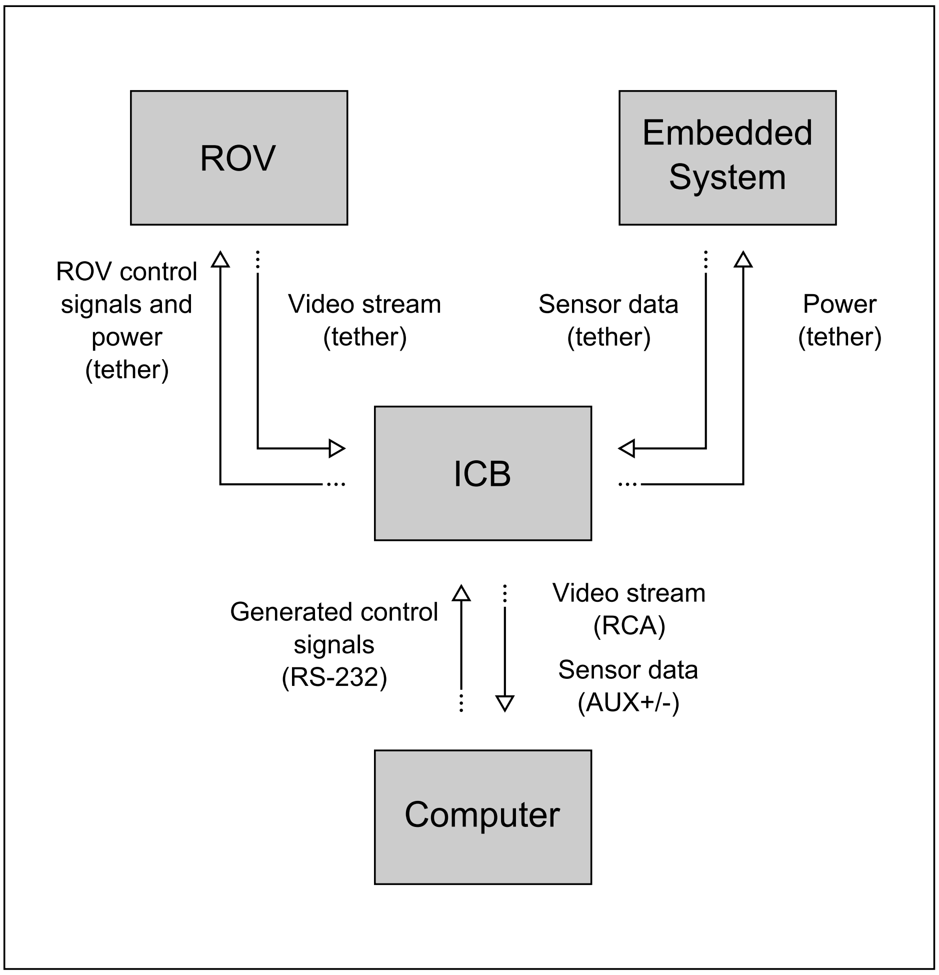


Figure 4.1: Overview of hardware setup. All subsystems communicate through the ICB.

Power

As described in Table 3.2, the internal ROV operating voltage is 48 Volts DC. This value indicates the maximum voltage that will be available to the embedded system. A DC-DC converter is necessary since system components normally use lower supply voltages and because a steady power supply is crucial for stable operation.

IMU

The IMU must have three accelerometers and three gyroscopes aligned with the orthogonal axes x_s, y_s and z_s. Measurement ranges must comply with the planned motion of the ROV, the acceleration range will only need to be within a few g (gravitational acceleration) and the gyroscope range less that $360°/\mathrm{s}$.

This component is central to the navigation system and should have error specification that reflect the significance of the unit. The IMU must at the same time fit the low-cost classification.

Compass

The compass should have three magnetometers to correctly sense the Earth's magnetic field during non-zero roll and pitch angles. The unit should have calibration routines to compensate for soft- and hard iron effects.

Pressure Transducer

Pressure measurements should be within the depth ratings of the ROV. As shown in Table 3.2, the ROV is primarily limited by the length of the tether, thus giving an operating range of 76 m below the sea level.

Data Transmission

A communication interface must be chosen that only needs two wires for robust data exchange. The tether has two free wires that are not used by the standard VideoRay system, the AUX+ and AUX-. The ground wire of the tether should not be used because of power noise and power transients.

Programming and Debugging

The MCU must have interfaces for firmware programming and debugging.

4.1.2 Tether Interface

The ROV is connected to the tether with the connector shown in Figure 4.2.

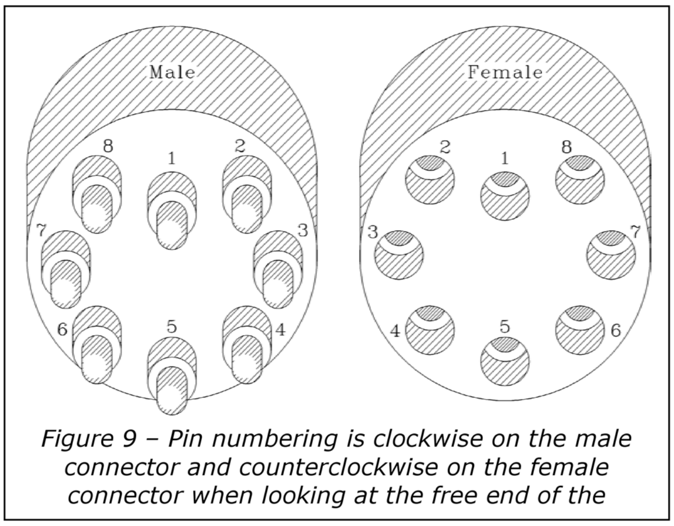

Male
Female
8
1
2
7
3
6
5
4
2
1
8
3
7
4
5
6
Figure 9 – Pin numbering is clockwise on the male connector and counterclockwise on the female connector when looking at the free end of the

Figure 4.2: Round tether connector VideoRay LLC ©videoray.com

Beneath the ROV's tether connection there is a female "SubConn Micro Low Profile 9" connector plug. By inserting the male equivalent (Figure 4.3) the embedded system can be connected to the tether and have access to the 48 V internal power source, ground and the AUX +/- wires. Pin numbers and color codes are shown in Table 4.1.

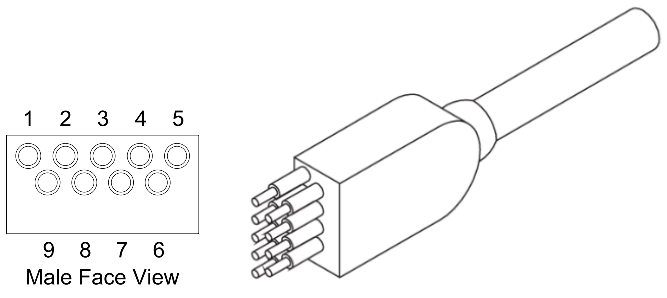


Figure 4.3: Male SubConn Micro Low Profile 9 tether connector SubConn Inc. ©subconn.com

Pin	Function	SubConn Color	Connector
1	Video -	Black	M2
2	Video +	White	M1
3	48 VDC +	Red	FM1
4	Aux +	Green	M1
5	Ground	Orange	FM2
6	Aux -	Blue	M2
7	Internal CAN +	White/Black	M1
8	Internal CAN -	Red/Black	M2

Table 4.1: Pin assignment by pin number for round tether connector and SubConn connector.

Communication via the AUX +/- wires must be designed to handle long transmission lines, noise and cross talk between the different wires. Each lead is insulated from each other, but not shielded. Transmission speed must be fast enough to satisfy the sensor sampling rates but at the same time consider the tether length (76 m).

4.1.3 ICB Interface

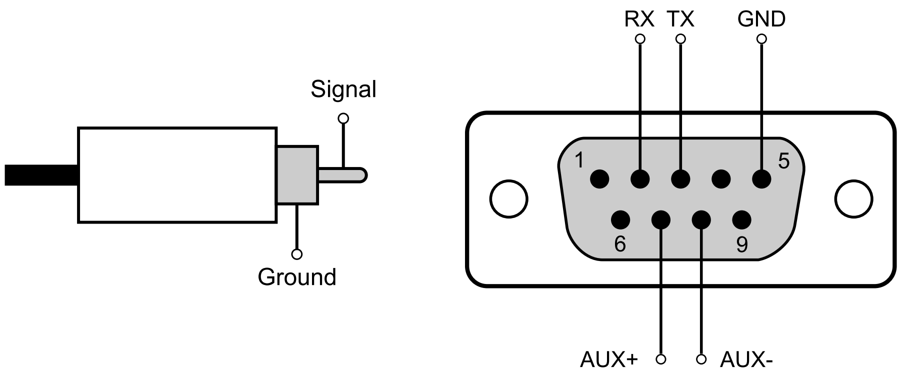
Video feed, sensor data and control signals must all be sent between the ICB and computer. Different solutions must be designed to interface the different standards.

Video Feed

The analog video feed is accessed from the ICB through a Phono plug (RCA) (Figure 4.4(a)) and must be digitized before being transmitted to the computer. The digitization hardware must comply with the following requirements:

- Capture video from a single RCA connector (composite video).
- Capture video from a 25 Hz PAL video source.
- The ability to run on computers with “normal” system performance specifications.
- Be portable/external and have a standard computer interface found on end user computers (e.g. USB 2.0 or FireWire).

A portable solution is desirable since ROV fieldwork will be done with a laptop computer.



(a) Male RCA/phono plug. (b) Male DE-9/DB-9/D-sub connector.

Figure 4.4: Available interfaces on the ICB.

Sensor Data

Sensor data will be made available on pin 7 (AUX+) and pin 8 (AUX-) of the DE-9/DB-9/D-sub connector (Figure 4.4(b)) mounted in the ICB. The computer interface depends on the chosen communication standard.

Control Signals

Control signals generated on the computer are transmitted to the ICB with RS-232 communication. Only the RX, TX and GND signals from the standard are used, refer to (Catsoulis 2005, ch. 9) for further description.

Transmission commands and parameters are thoroughly described in Appendix A. The required communication parameters:

- Baud rate 9600 bps
- 8 data bits
- 1 stop bit
- No parity bits

Control signals are sent to the ICB's D-sub connector with the pin configuration shown in Figure 4.4(b) (pin 2, 3 and 5). Because both the ICB and computer have male connectors, a female-to-female straight through cable is needed (standard female-to-female cables have crossover wiring). Such a cable has been made for this purpose by student Knut Ove Stenhagen.

4.1.4 Computer

The computer should be a laptop to enable system portability. The computer must be able to retrieve the digitized video, sensor data and send control signals to the VideoRay system. The following requirements are set for the computer:

- "Normal" system performance specifications.
- Two available USB 2.0 or FireWire port.
- A available COM-port, or a COM-to-USB converter if the laptop is not equipped with a COM-port (which is the case for most new laptops).

4.2 Firmware

The MCU memory must be programmed with a group of instructions to perform the planned tasks. This instruction group is hereby referred to as the firmware and is responsible for gathering, processing and transmitting the sensor data. The firmware is stored in non-volatile memory where the instruction types, execution order and input parameters govern the overall behavior.

The size of the firmware is limited by the MCU's memory capacity.

4.2.1 Data Processing

Data gathering will depend on the different sensor interfaces and update rates. Sensors can indicate if data is ready by generating MCU interrupts or the MCU can poll the sensor to check if new data is available.

Different sensors have different data formats. Returned values depend on byte order, data length and the purpose of each bit. Some bits can indicate status information instead of being part of the actual measurement value. Data should be processed into a common format.

4.2.2 Data Transmission

Sensor measurements are combined into a single sample and transmitted at a specified rate. Sensor data must be packaged in a way that minimizes the number of transmissions per sample.

4.2.3 Remote Configuration

Remote firmware reprogramming is needed when the complete system is mounted and the embedded system is enclosed. Parts of the firmware must therefore enable remote MCU memory flashing.

Remote commands are necessary to enable online configuration of sensor properties, sample rates and to enable forced MCU resets. This will make the firmware dynamic in the sense that small changes in the system configuration are possible without having to reprogram the firmware.

4.3 Software

Measurements from the embedded system must be collected continuously to maintain system responsiveness. At each new data arrival the input is processed, stored and made available for state estimation. The estimator must be initialized correctly through a calibration routine.

To control the VideoRay ROV from a computer, a Graphical User Interface (GUI) should be made for easy interaction. This application should have the following properties:

- Interactive:
  - The path to be followed by the ROV must be user definable with x_t, y_t and z_t coordinates for each waypoint.
  - The user must be able to start and the stop the different system modules and enable calibration as needed.
  - Controller gains and filter parameters must be editable during run time.
- Informative:
  - Display the planned path and the ROV progress along it.
  - Display filtered and unfiltered states.
  - Display ROV attitude and depth.
  - Display ROV video feed.

Generated control inputs are based on estimated states and controller design. The application must deliver the input at a satisfying rate.

Computer - ICB communication is based on RS-232. There must be a serial communication module that complies with the communication protocol defined in Appendix A.

4.4 State Estimation

The state estimator/Kalman filter shall combine the different measurements and provide the best possible estimate of the actual system state. Gyroscope, accelerometer, compass and pressure values are inputs to the filter while attitude and position are its output. The estimated states must be sufficiently accurate during the length of a search pattern to obtain a consistent sweep.

4.5 Controller

The ROV has three actuators; a port, starboard and vertical thruster. With these actuators the controller should regulate the following states:

- Position: The ROV's position in the T reference frame.
- Depth: The ROV's depth in the T reference frame.
- Heading: The ROV's yaw angle in the T reference frame.

The ROV is underactuated in the sense that the sway y of the vehicle is not directly controllable. Error compensation will therefore be difficult when the ROV experiences disturbances in the port/starboard direction.

When the ROV is deployed the tether will affect the vehicle motion. Current-related cable pull and spring-like effects results in position, depth and heading disturbances.

Chapter 5

Design

5.1 Hardware

5.1.1 Sensors

The embedded system design depends primarily on the choice of sensors, communication protocol and MCU. Supply voltage, programming interfaces and PCB layout will follow as a result of the initial selection process.

IMU

Knut Ove Stenhagen's previous work with the ROV employed an Analog Devices IMU for inertial measurements, the ADIS16354AMLZ shown in Figure 2.4. This unit has measurement ranges that fit the stated requirements, a high bandwidth, internal filtering and calibration routines. Selected specifications:

- Gyroscope ranges: $\pm 75°/s$, $\pm 150°/s$, $\pm 300°/s$
- Accelerometer range: $\pm 1.7g$
- Bandwidth: maximum 350 Hz
- Sample rate: maximum 819 Hz
- Data resolution: 14-bit
- Operating temperature: $-20°C$ to $+85°C$

- Operating voltage: +4.75 V to +5.25 V DC
- Interface: SPI with high polarity and phase one

The IMU has an adjustable, internal sample rate where a lower rate will result in reduced power consumption. Maximum sample rate will however give the best noise performance according to the datasheet. Since there is no power limitation in this application, the maximum sampling rate is chosen.

With a 14 bit data resolution the IMU can achieve a sensitivity of 0.4625 $\frac{\text{mg}}{\text{LSB}}$ (Least Significant Bit) for the accelerometers and 0.07326 $\frac{°/\text{s}}{\text{LSB}}$, 0.03663 $\frac{°/\text{s}}{\text{LSB}}$, 0.01832 $\frac{°/\text{s}}{\text{LSB}}$ for the gyroscopes, depending on the chosen range.

Serial Peripheral Interface (SPI) is a synchronous protocol where a master is connected to one or more slaves and data exchange is achieved by bit shifting. Communication is full-duplex and based on four wires. The master selects the desired slave and generates a clock signal with a certain clock polarity and phase. SPI is further described in (Catsoulis 2005, ch. 7).

Error ratings:

- Gyroscope bias (in run bias stability, 1σ): $0.015°/s$
- Gyroscope output noise:
  - $0.60°/s$ rms ($\pm 300°/s$)
  - $0.35°/s$ rms ($\pm 150°/s$)
  - $0.17°/s$ rms ($\pm 75°/s$)
- Accelerometer bias (velocity random walk): 0.135 $\frac{\text{m/s}}{\sqrt{\text{hr}}}$
- Accelerometer output noise: 4.7 mg rms

Compass

The magnetic compass unit, OS4000-T from OceanServer Technology (Figure 5.1) was mainly chosen because of its small size (16 mm x 16 mm), but also because of its rich functionality, with three magnetometers and three accelerometers. This sensor combination enables the unit to automatically calculate its attitude and transform the magnetometer measurements to T. The compass heading can be read directly without having to transform the values at a later instance.

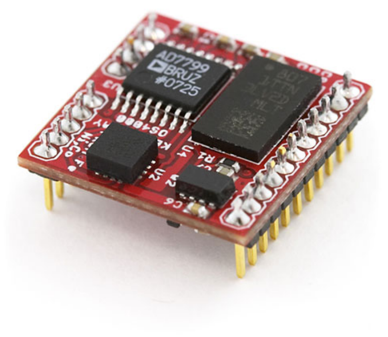


Figure 5.1: OceanServer Technology OS4000-T.

Compensation for hard- and soft iron effects are done with embedded calibration routines. Selected specifications:

- Compass accuracy: maximum 1.5° rms error when roll and pitch angles are $< \pm 60°$
- Bandwidth: maximum 40 Hz
- Output resolution: 0.1°
- Operating temperature: $-40°C$ to $+80°C$
- Operating voltage: +3.3 V to +18 V DC
- Interface: TTL-level UART
- Baud Rate: 4800 $^{\text{bit}}/_{\text{s}}$ to 115200 $^{\text{bit}}/_{\text{s}}$

Measurements are sent periodically from the compass as soon as power is applied. The sample rate is set to 10 Hz as this is the fastest rate allowed with a baud rate below 115200 $^{\text{bit}}/_{\text{s}}$ and is sufficient for the current application. The baud rate is set to 19200.

Universal Asynchronous Receiver Transmitter (UART) is a serial communication interface where receiver and transmitter communicate without a synchronizing clock signal. Both parts must agree on baud rate, the number of stop bits, data bits and data consistency checks before exchanging data. UART is further described in (Catsoulis 2005, ch. 9).

TTL-level (Transistor - Transistor Logic) refers to the logic levels used to represents ones and zeros. These levels are 0 V to 0.8 V for low voltages and 2 V to V_{cc} (supply voltage) for high voltage applications.

Pressure Transducer

Pressure measurements are performed with a Honeywell S&C 19C100PA4K shown in Figure 5.2. The transducer is used in (Skjaeveland 2009) and was shown to provide satisfactory accuracy when combined with proper signal processing.

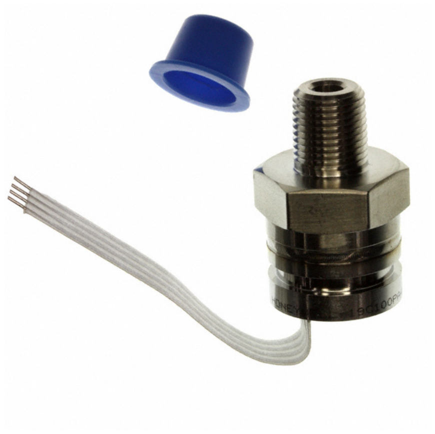


Figure 5.2: Honeywell S&C 19C100PA4K.

Selected specifications:

- Pressure range: absolute, 0 psi to 100 psi (300 psi and above will damage component)
- Output range: ratiometric, 0 V to $\frac{V_{cc}}{100}$
- Output non-linearity: maximum ±0.25% of output span
- Output resistance: 4.5 kΩ
- Operating temperature: −40°C to +125°C
- Operating voltage: maximum +15 V DC

Absolute pressure measurements are relative to vacuum. The transducer will therefore measure the pressure of one atmosphere when placed at sea level. Average sea-level pressure is defined as 14.696 psi = 101325 $^{N}/_{m^2}$ and leaves 85.304 psi of the initial range to measure water depth. 100 psi = 689400 $^{N}/_{m^2}$.

Equation (2.4) relates pressure to depth and is used to calculate the operating range of the sensor:

$$\begin{aligned} p_{100psi} &= p_a + \gamma z_n \\ z &= \frac{p_{100psi} - p_a}{\gamma} \\ &= \frac{689400 \text{ N/m}^2 - 101325 \text{ N/m}^2}{10050 \text{ N/m}^3} \\ &\approx 58.5 \text{ m} \end{aligned}$$

An operating depth of 58.5 m is less than the tether's 76 m limitation, but is still a satisfactory depth for search operations.

Ratiometric output indicates that the output is directly proportional to the input. For this unit the output is (as stated in the specification) proportional to the input by a factor of $\frac{1}{100}$. E.g. if the transducer is driven by a 10 V power source the output will be in the range 0 mV to 100 mV.

5.1.2 Tether Communication

In previous work with the ROV, Knut Ove Stenhagen experimented with the Controller Area Network (CAN) bus for communication. The standard was developed by Bosch during the 1980s and was intended to withstand the noise levels of an automotive electrical environment (Catsoulis 2005, p. 216).

The CAN bus uses two wires for communication, CANH (high) and CANL (low), allows multiple masters and is terminated at each end with a 120 Ω resistor. Termination is needed to avoid signal reflections.

Selected specifications:

- Maximum transmission speed: 1 Mbps over twisted-pair wiring with length $\leq$ 40 m
- Maximum bus length: 1000 m
- Data bytes per transmission frame: 0-8 bytes

The standard is suitable for this application because of inherent noise robustness, compatible transmission lengths/speeds and previous experience with the bus.

Two CAN specifications are described in (Bosch 1991) that are practically identical, except from their frame identifiers. CAN 2.0 A supports an 11-bit

ID while CAN 2.0 B supports a 29-bit ID. This system only has two nodes and a limited amount of message types, hence CAN 2.0 A is chosen. One message will therefore contain 44 bits of CAN specific information together with 0-8 data bytes.

The VideoRay system already has a CAN bus running through the tether, but this is only accessible from the embedded system and not the ICB. Modifications could have been done to the control unit, but the AUX+/- tether wires are unused and will successfully connect the embedded system with the ICB. This will give a two-node network that needs termination at both ends. Refer to (Catsoulis 2005, ch. 12) and (Bosch 1991) for a further description and specifications.

Two pins of a Phoenix Contact MKDSN1,5/4-5.08 is used to connect the AUX+/- wires to the PCB.

5.1.3 MCU

The chosen MCU is an Atmel AT90CAN32 with a built-in CAN controller, SPI, UART and a 10-bit Analog-to-Digital Converter (ADC). These options allow the MCU to interface all the chosen sensors and generate output compatible with the CAN bus.

Selected specifications:

- Maximum frequency: 8 MHz at 2.7 V and 16 MHz at 4.5 V
- Throughput: up to 16 MIPS at 16 MHz
- Memory:
  - 32K Bytes ISP Flash
  - 1K Bytes EEPROM
  - 2K Bytes SRAM
- Interfaces:
  - 2 x UART
  - 1 x SPI
  - 1 x CAN controller
- 8-channel, 10-bit resolution ADC

- General purpose I/O pins
- Two 8-bit and one 16-bit timer/counter
- Joint Test Action Group (JTAG) interface for programming and debugging
- Operating temperature: $-40°C$ to $+85°C$
- Operating voltage: $+2.7\,V$ to $+5.5\,V$ DC

Figure 5.3: Atmel TQFP-64 AT90CAN32.

The In-System Programmable (ISP) Flash allows the system memory to be reprogrammed in-system either by using an external serial interface or by an on-chip boot program running on the MCU. Program data will in the latter case be provided by an interface of choice (Atmel 2008).

16 Million Instructions Per Second (MIPS) throughput combined with 32K Bytes of ISP Flash memory is more than sufficient for the current application. To obtain this level of performance a 16 MHz external clock is used instead of the internal 8 MHz clock. Dedicated oscillators are more accurate and will reduce the number of communication-related errors (the AT90CAN32 datasheet states that CAN bus needs very accurate timing for high baud rates and advices to only use external crystals for CAN applications).

The 8-channel ADC allows for eight single-ended sources, i.e. signals relative to a common ground. The conversion range is governed by a voltage source serving as the reference value for all measurements. This source can be selected as AV_{cc}, an internal 2.56 V reference or an external AREF pin. The conversion rate is based on the system clock prescaled with a chosen value.

5.1.4 Power

A supply voltage of 5 V is chosen on the basis of the different component's ratings. The ROV and embedded system share the same 48 V power lines so noise and transient voltages must be handled by the DC-DC converter.

A Traco Power TEN 5-4811WI was used by Knut Ove Stenhagen and has desirable properties:

- Input range: +18 V to +75 V DC
- Output: +5 V DC/1000 mA
- Short-circuit and over voltage protection
- Built-in electromagnetic interference filter
- Galvanically isolated input/output
- Operating temperature: −40°C to +85°C

Figure 5.4: Traco Power TEN 5-4811WI (different unit, but same package).

The converter is enclosed and will limit some of the electromagnetic noise generated from the conversion. This is important since the converter and compass unit are placed on the same PCB in a confined space.

The remaining two pins of the Phoenix Contact MKDSN1,5/4-5.08 is used to connect the power wires to the PCB.

5.1.5 Programming- and Debugging Interfaces

Programming and run-time debugging of software/hardware can be done with a JTAG interface. The standard enables single-stepping through code, signal line toggling and access to registers/memory (Catsoulis 2005, p. 157).

JTAG is serial and uses four dedicated signals: Test Data In (TDI), Test Data Out (TDO), Test Mode Select (TMS) and Test ClocK (TCK). By adding the ground/supply voltages and MCU reset line, the embedded system can be connected to a Atmel AVR JTAGICE mkII which enables programming/debugging over USB. The reset line enables the system to be reset after a completed memory flash.

A reset button was added to the design to manually reset the circuit during development. Light-Emitting Diodes (LED) and an extra UART were added for simple status indication and textual debug information.

5.1.6 Signal Processing and Support Components

Pressure Measurements

The signal processing related to pressure measurements is based on the work done in (Skjaeveland 2009). The same components and processing steps are used to achieve the same level of performance.

The 19C100PA4K input terminals +IN/-IN are connected with the embedded system's supply voltage and ground signal. This gives an output range between 0 mV and 50 mV at the +OUT/-OUT terminals and must be amplified before digital conversion. A single-supply INA155UA instrumentation amplifier from Burr-Brown is used for this purpose. The unit has rail-to-rail output characteristics to achieve linear amplification close to 0 V. Because the transducer will sense the atmospheric pressure at sea-level the minimum amplified signal will be outside non-linear amplification areas.

Low-pass filtering is necessary to remove noise and signal components higher than the Nyquist frequency ($f_{ADC}/2$). This is done to smooth the signal and thereby avoid distortions from unpredictable signal convolutions (Atmel 2008). An RC-filter with a 10 Hz cutoff frequency is used for this purpose and will allow fast changes in depth while removing unwanted components. The resistor- (R) and capacitor (C) values control the cutoff frequency ($\omega_c = 2\pi f_c$) through the following relation (Nilsson & Riedel 2005, eq. 14.19):

$$f_c = \frac{1}{2\pi RC} \tag{5.1}$$

The ADC is configured to use the internal 2.56 V source as its reference voltage V_{ref}. By amplifying the transducer output with a gain value of 50 the INA155UA will deliver a voltage between 0 V and 2.5 V to the ADC input

V_{in} relative to ground. The conversion values are calculated with:

$$ADC = \frac{V_{\text{in}} \cdot 1023}{V_{\text{ref}}} \tag{5.2}$$

A gain value of 50 is accomplished by soldering a $0\,\Omega$ resistor between the pin 1 and 8 of the INA155UA as described in the datasheet.

The maximum value from the INA155UA is 2.5 V while the ADC reference is set to 2.56 V. This means that the number of discrete values is limited to $\frac{2.50\,\text{V} \cdot 1023}{2.56\,\text{V}} = 999{+}1$ levels and a resolution of $0.1\,{}^{\text{PSI}}/_{\text{bit}}$ or $6.86\,{}^{\text{cm change of depth}}/_{\text{bit}}$.

The complete signal processing chain is shown in Figure 5.5.

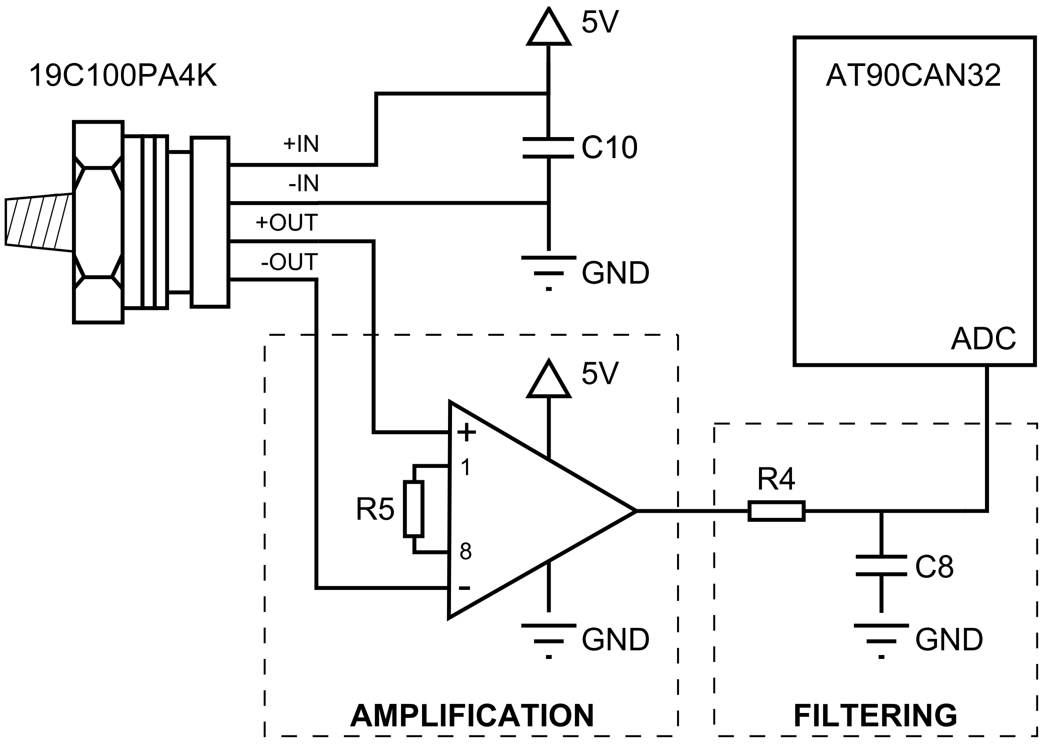


Figure 5.5: Pressure measurement processing chain, adapted from (Skjaeveland 2009)

CAN

A CAN transceiver is needed to convert transmitted and received CAN frames between the MCU and physical bus. The unit converts digital input to a differential output suitable for transmission (and vice versa), and protects circuitry from voltage peaks, Electro Magnetic Interference (EMI) and Electrostatic Discharge (ESD). The chosen transceiver is a Microchip MCP2551.

Because the embedded system functions as one of the two endpoints of the bus there must be a 120 Ω resistor between CANH and CANL. The slew-rate of the transceiver controls the rise and fall times of CANL/CANH. On the MCP2551 this is adjusted by a resistor connected between pin 8 and ground. By choosing a 22 Ω resistor the unit will allow the fast transmission speeds that are needed by the current application. This value is known to work from previous experience and will allow about 17 $^{V}/_{\mu s}$ according to Figure 5.6.

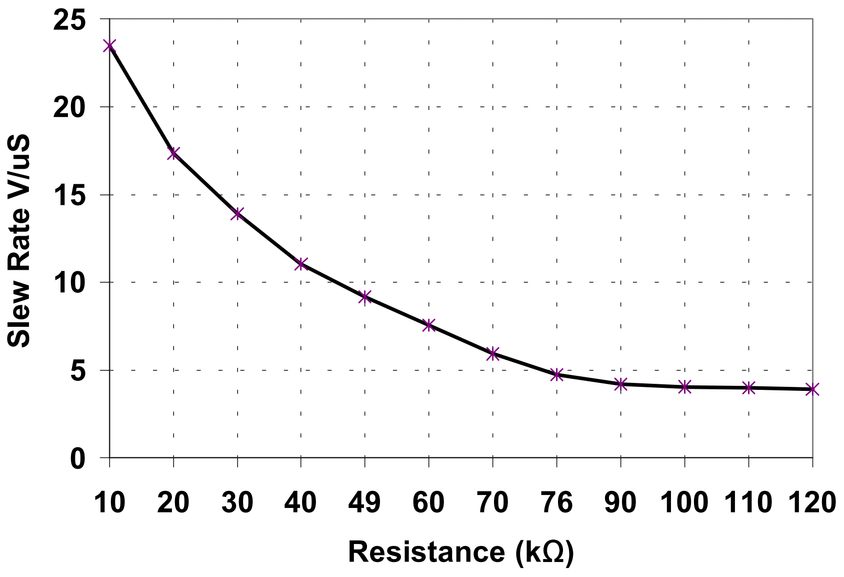


Figure 5.6: Slew-rate as a function of resistance for the MCP2551, ©Microchip Technology Inc.

5.1.7 Housing

The housing designed for the embedded system must be limited in size to reduce buoyancy and provide the required mounting options for PCB, SubConn cable and pressure transducer. Figure 5.7 shows the housing dimensions necessary to calculate the volume and size limitations. The housing is made up of two parts, the lid and the cylinder.

An estimate of the housing volume can be found with the formula for cylindrical volume: $V = r^2\pi h$, where r is radius and h is cylinder height. The

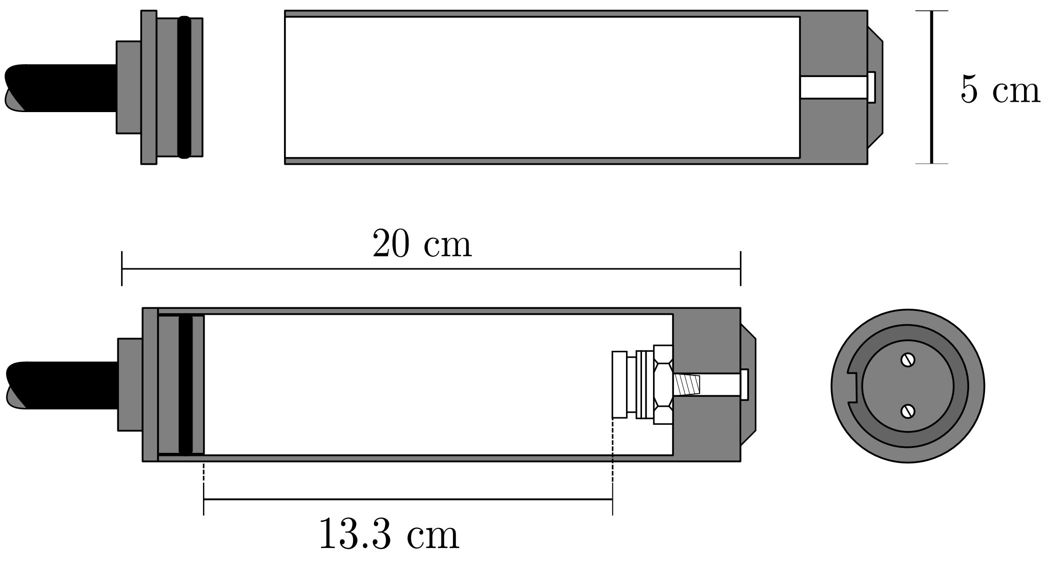


Figure 5.7: Open and closed housing with and without pressure transducer.

connected SubConn cable is 15 cm longer than the terminator that is usually connected to the ROV and has a diameter of 1 cm. When the system is mounted on the ROV, the total water displacement will increase by the following volume:

$$\begin{aligned} V &= V_{housing} + V_{cable} \\ &= 2.5^2\,\mathrm{cm}^2 \cdot \pi \cdot 20\,\mathrm{cm} + 0.5^2\,\mathrm{cm}^2 \cdot \pi \cdot 15\,\mathrm{cm} \\ &= 404.5\,\mathrm{cm}^3 \end{aligned}$$

If the volume is multiplied with the specific weight of seawater (previously defined to be $10050\,{}^{\mathrm{N}}/_{\mathrm{m}^3}$) the cable and housing will produce about 4.07 N buoyancy.

The total gravitational force acting on the housing, cable and embedded system was weighed to be 3.83 N, which is a 3.49 N increase compared to the terminator alone. By summing the gravitational- and buoyancy forces it can be shown that a 60 gram weight will compensate for the added buoyancy.

To install the 19C100PA4K a threaded hole is cut in the bottom of the cylinder with a diameter equal to the half-way diameter of the transducer's conic mounting screw. This will ensure that the system is watertight. A plate with a small duct is mounted at the bottom to reduce the effect of velocity induced pressure changes and to avoid foreign bodies getting caught in the hole.

The lid is where the SubConn cable enters and the PCB is mounted. There is an inlaid O-ring in the lid and isolating tape around the cable entrance to make a tight seal. The PCB mounting bracket is screwed into the lid and has two holes where the circuit board can be fastened.

The housing will alter the hydrodynamic properties of the ROV, but this is not considered in this thesis.

5.1.8 PCB

The design uses Surface Mount Devices (SMD) and two layers (top and bottom) to minimize the size of the PCB. According to Figure 5.7 it can be seen that the PCB length must be less than 13.3 cm. PCB width must be less than 5 cm because of the housing diameter, but the height of the IMU must also be considered.

Signal and power tracks should be as wide and short as possible. This reduces the track's DC resistance, inductance and capacitance. Track corners should have 45 ° angles and 90 ° angles should be avoided because of potential problems during etching (Jones 2004).

If the signal and return paths are close together, their magnetic fields will be mutually cancelled. Minimizing the distance between the wires and their lengths is called *minimizing the current-loop area.* By using a ground plane the loop area can be minimized by serving as a current return path for all loops in the circuit (Catsoulis 2005, p. 121). A ground plane is placed on both the top and bottom layers around signal and power tracks. Vias are placed at strategic places across the board to connect the planes and minimize ground paths.

Power lines are susceptible to noise which can cause unexpected behavior in the system components. To avoid this, the noise should have a path to ground locally for each component. This is achieved by connecting a decoupling capacitor between power and ground, as close to each device as possible (Catsoulis 2005). From prior experience the capacitor dimension is set to 100 nF.

To improve the results of the pressure measurements, techniques should be applied to cancel analog noise. (Atmel 2008) mentions some actions that can be taken:

- AV_{cc} and V_{cc} on the AT90CAN32 should be connected via an LC network with $L = 10\,\mu H$ and $C = 100\,nF$

- Analog signal paths should be as short as possible and kept away from high-speed switching digital tracks.

These measures were introduced with a ferrite bead instead of an inductor in the LC network.

The compass should be placed as far away from the DC-DC converter, power lines and other signal lines as possible. This will reduce the electromagnetic disturbances that affect the magnetometer measurements.

The schematics and board layout were made in CadSoft EAGLE v.5.0.0 and are shown in Figure 5.9 and 5.8 respectively.

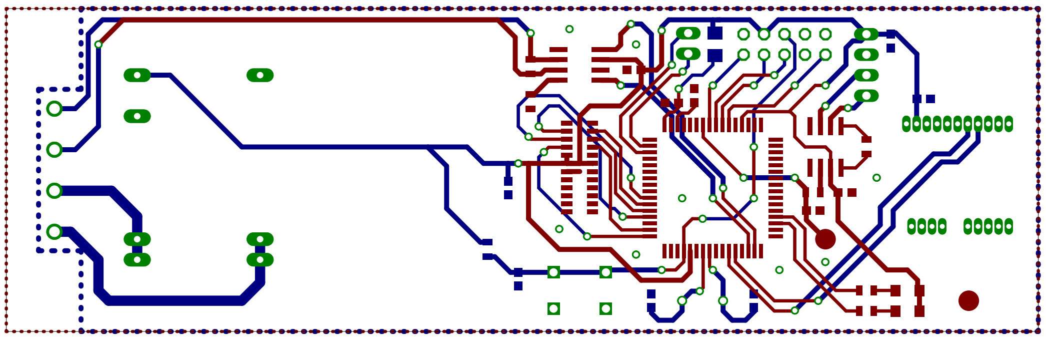


Figure 5.8: Board layout of the embedded system. Showing tracks, pads, vias and an outline of the ground planes.

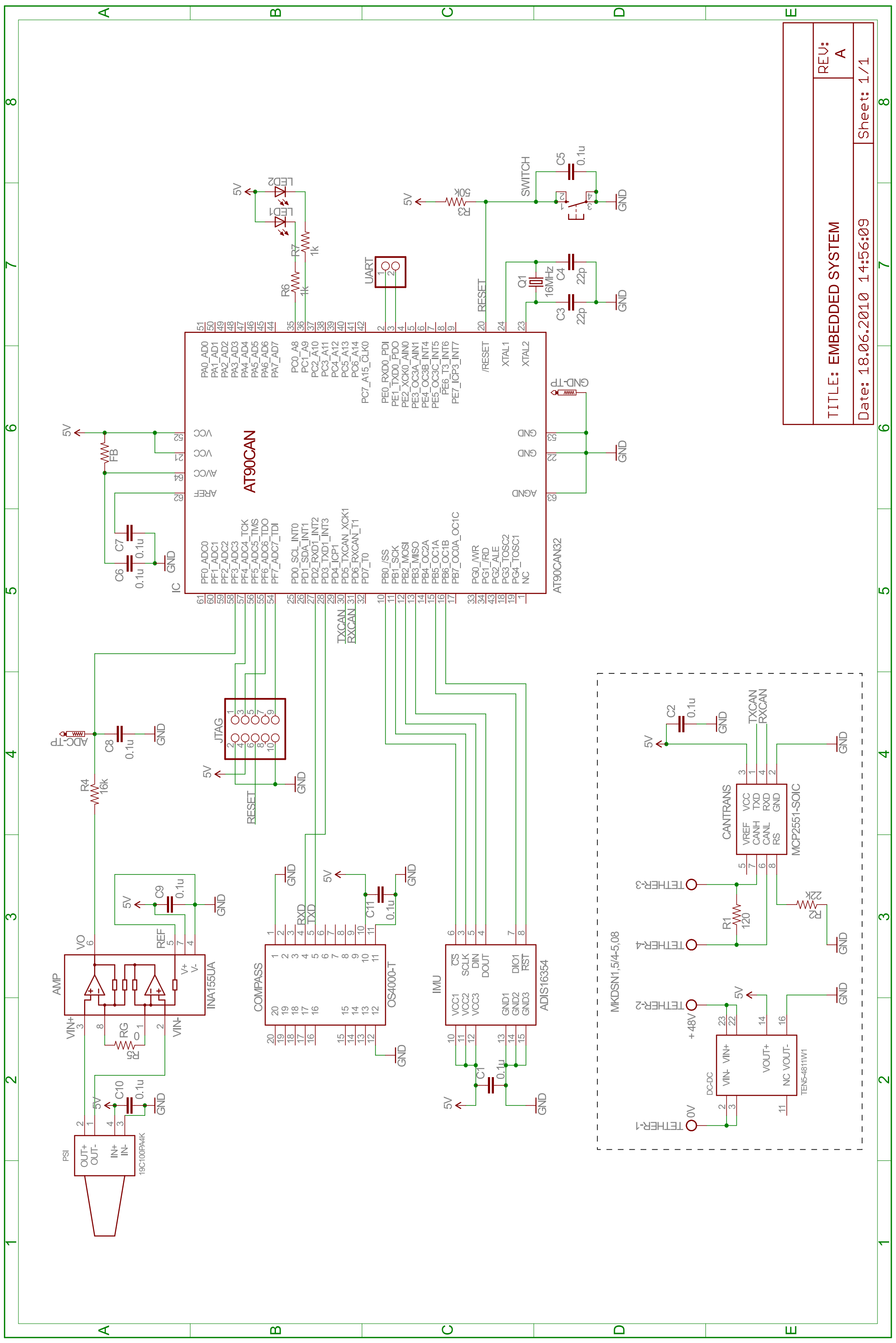


Figure 5.9: Schematic layout of the embedded system.

5.1.9 ICB Interface

To digitize the analog video stream a TerraTec Grabby frame grabber was chosen. The product is cheap, can interface RCA composite video and delivers output via USB. Selected specifications are given in Table 5.1.

Data Interface	USB 2.0
Analog Video Input	Composite- and S-Video
PAL Capture Resolution	Up to 720x576 pixels
PAL Capture Rate	Up to 25 fps at maximum resolution
Power Source	USB

Table 5.1: TerraTec Grabby specifications.

The CAN bus lines are interfaced with a Kvaser Leaf SemiPro HS to make the CAN frames available via USB. The product's interface to the CAN bus

Data Interface	USB 2.0
Galvanic Isolation	Yes
Supported Message Formats	2.0 A and 2.0 B
Bitrate	$5\,{}^{\text{kbit}}/_{\text{s}}$ to $1000\,{}^{\text{kbit}}/_{\text{s}}$
Maximum Message Rate	15000 Hz
Power Source	USB

Table 5.2: Kvaser Leaf SemiPro HS specifications.

is a male D-sub connector where pin 2 connects with CANL and pin 7 with CANH. This layout is identical to the PEAK PCAN-USB adapter. CANH and CANL are not internally terminated.

As mentioned in the specification a female-to-female straight through cable is needed to connect ICB RX/TX/GND correctly to the computer.

Both control signals from the computer and CAN bus lines use the same D-sub ICB interface for communication. To enable simultaneous connection, the signals from the ICB must be split into two D-sub connectors. One male interface for control signals and one female interface for the CAN adapter. The CANL/CANH lines are terminated inside the connector housing with a $120\,\Omega$ resistor. The wiring is presented in Figure 5.10.

The complete hardware setup is depicted in Figure 5.11.

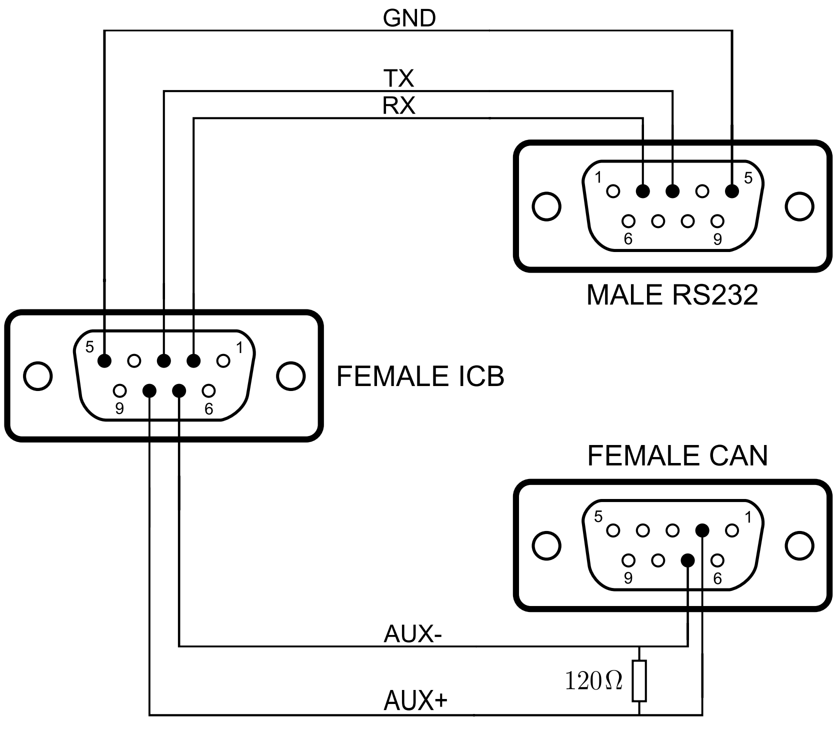


Figure 5.10: Splitter wiring scheme.

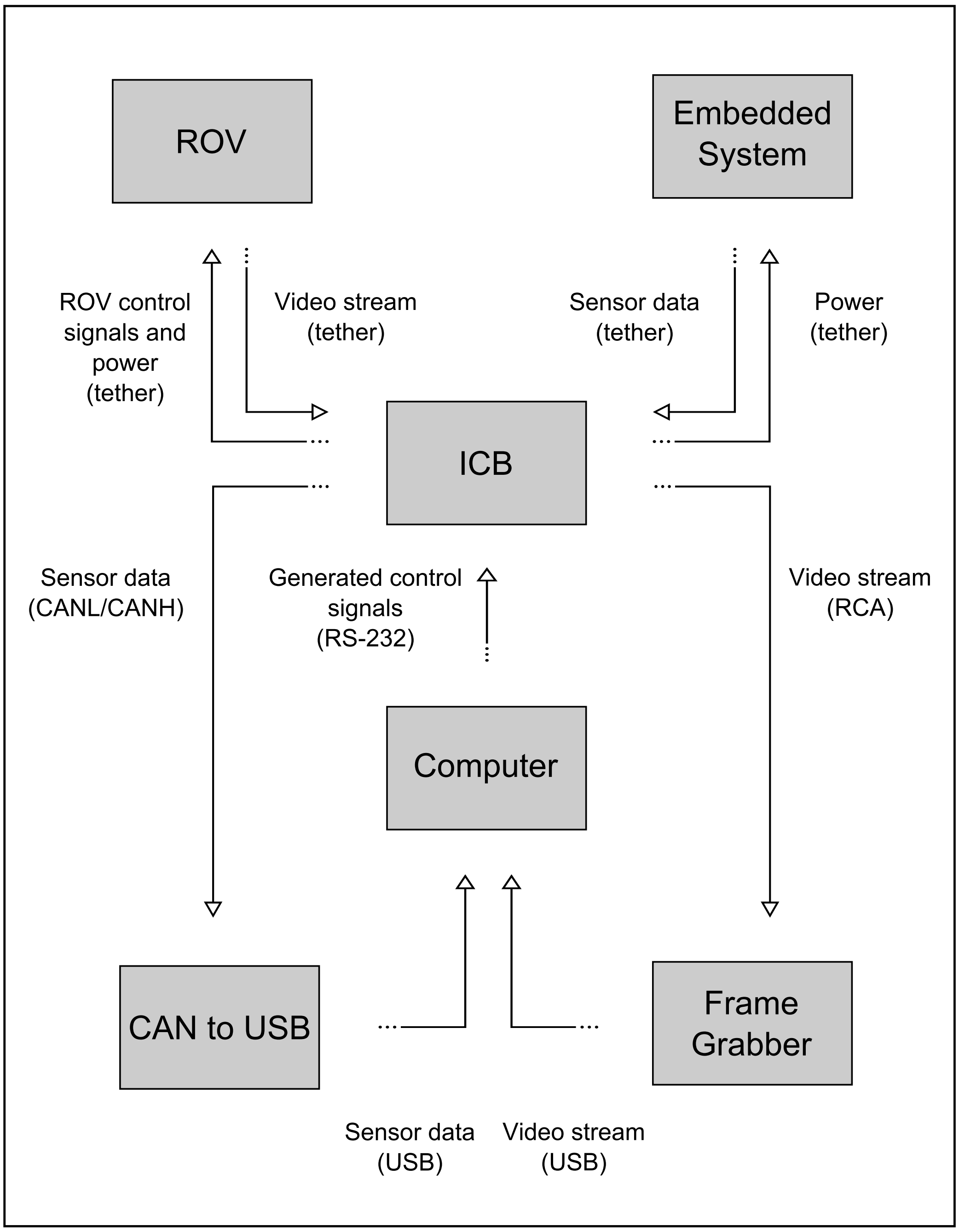


Figure 5.11: Extended overview of hardware setup.

5.2 Firmware

The firmware on the AT90CAN32 is stored in ISP Flash memory which is divided into a Bootloader Flash Section (BFS) and an Application Flash Section (AFS). This segmentation allows the on-chip boot program stored in the BSF to reprogram the ASF without affecting bootloader code. Hence providing true Read-While-Write operation.

The MCU is programmed to always start in the bootloader section by setting the BOOTRST fuse to zero (Atmel 2008, sec. 24.5). If the MCU starts after a power-on reset, or the MCU just came from the bootloader, the control flow is directed to the application section. If none of the conditions are met, the bootloader enters programming mode and resets the MCU when finished. To check if the MCU just came from the bootloader, a flag stored in non-volatile EEPROM is used.

When the control flow enters the application section, the EEPROM flag is changed. If memory programming is desired the user will send a reset command via CAN and the MCU will reset. A chart describing the control flow is shown in Figure 5.12.

5.2.1 Bootloader Flash Section

The bootloader's role is to reprogram the application section when a user wishes to update functionality. Several interfaces can be used to deliver memory data during programming, but the choice that fits the current setup is data delivery via CAN. When the bootloader starts after an application reset, it will configure the necessary parameters and start listening for memory data.

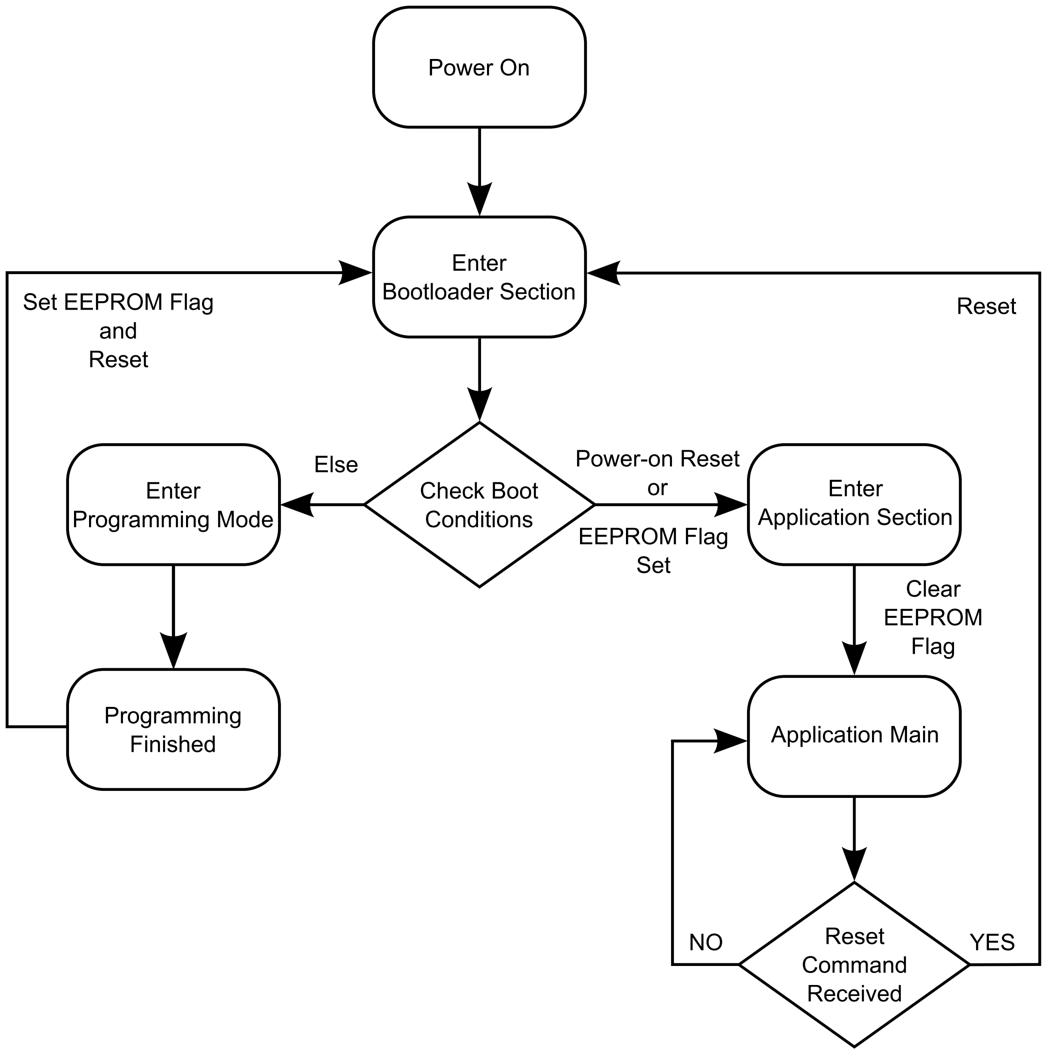


Figure 5.12: Firmware control flow after the embedded system is powered on.

5.2.2 Application Flash Section

The application is structured to use timers and interrupts to sample, receive and transmit data from the sensors. Two interrupt routines and the main routine control application behavior:

- Compass Interrupt Routine: Parse and store heading- and temperature values when UART data is received from the compass.
- Master Interrupt Routine: Sample ADC- and IMU data, assemble all measurements into three packages and transmit via CAN. Periodically run the procedure at a specified rate.
- Main Routine: Initialize all subsystems and execute commands received via CAN. Sleep when idle.

Compass Interrupt Routine

Data from the compass can be delivered in several sentence formats and can contain a number of different measurements. The chosen standard is referred to as “$C” format and the chosen data fields are heading- and temperature measurements. A valid “$C” sentence could look like “$C120.5T25.4*20”:

$	Start of the sentence
C	Succeeding characters will be the compass reading
T	Succeeding character will be the temperature reading
*xx	End of sentence, where xx indicates the 8 bit XOR-sum of all character values between $ and *

Each time a character is received via UART1 an UART receive interrupt is generated. Each interrupt calls a routine that parses the character and controls the XOR-sum if a sentence is completed. If a sentence passes the test the heading- and temperature values are stored and made available to the master interrupt routine.

Heading output is within the interval 0.0 to 359.9 while temperature values will be above zero. Both data fields are presented with decimal precision which is undesirable in an MCU context. To avoid floating point values the decimation symbol is ignored during parsing such that the measurements can be stored as 16-bit integers. Data reconstruction is done at a later point by dividing the values with the factor ten.

Master Interrupt Routine

The rate at which the embedded system gathers and transmits data is controlled by Timer0 on the AT90CAN32. Interrupts are generated at each timeout and the corresponding interrupt routine is called, performing the following tasks:

1. Sample IMU.
2. Assemble x_s, y_s and z_s accelerations from the accelerometers and transmit as first CAN message.
3. Assemble x_s, y_s and z_s rotation rates from the gyroscopes and transmit as second CAN message.
4. Sample ADC.
5. Assemble compass heading, temperature and ADC data and transmit as third CAN message.

Each measurement is stored as a 16-bit integer and divided into high/low byte when stored in a CAN message. Each message will then contain six bytes of data together with the 44 bits of CAN 2.0 A specific information. An overview of the different CAN message data fields is shown in Figure 5.13.

IMU When sampling the IMU, SPI is used to request gyroscope and accelerometer data in the x_s, y_s and z_s direction. One request will retrieve data from one sensor and one axis, so to access all six measurements, six requests must be made. The IMU takes 16-bit input as shown in Figure 5.14 where the 8-bit register address refers to a particular axis on accelerometer/gyroscope and data bits are “don’t care” for the read requests. SPI transmits/receives one byte per cycle which results in two SPI cycles per IMU request.

For each 16-bit request frame, a 16-bit data frame is simultaneously returned as a result of the previous request. This is because of the SPI’s full duplex capabilities that make sequential sampling efficient. Returned data is big-endian which means that the byte with the most significant bits (high byte) is returned first, then the byte with the least significant bits (low byte). Two example request cycles are shown in Figure 5.15.

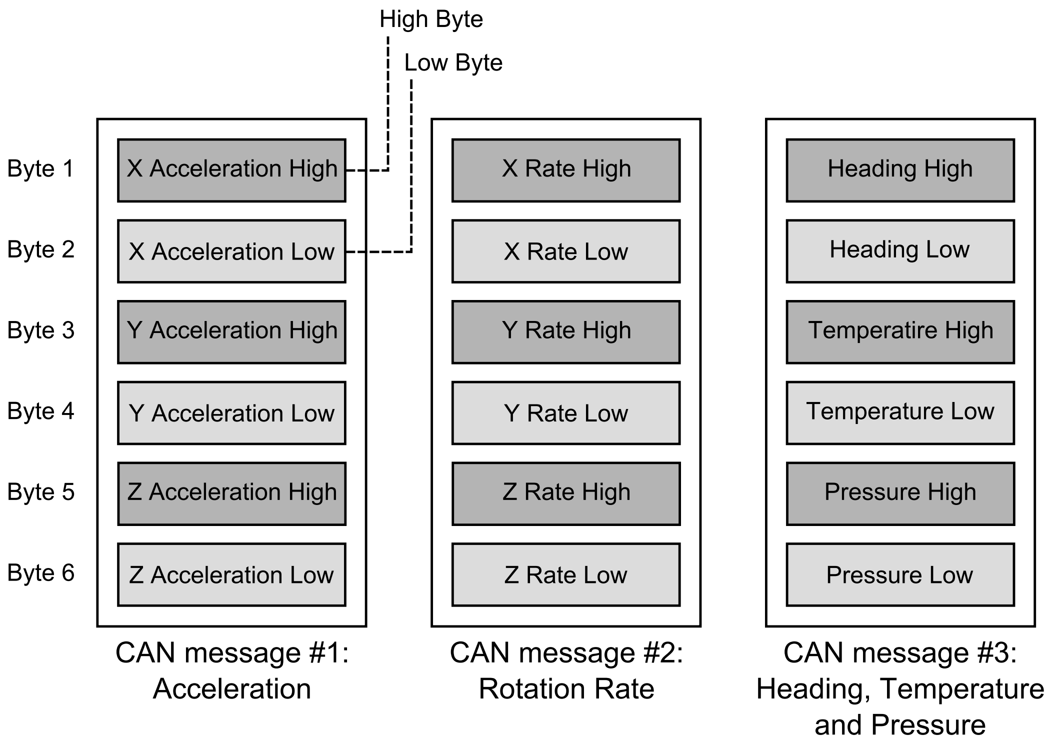


Figure 5.13: The three CAN messages sent at each execution of the master interrupt routine.

The 16-bit return data consists of a 14-bit measurement and two status bits: New Data (ND, bit 16) which equals 1 for new data, and Error/Alarm (EA, bit 15) which equals 1 if an internal error or a user specified alarm occurred (further error description is found by checking the internal STATUS register). The 14-bit measurement is an integer value that must be scaled in a later instance to obtain the correct floating point value. To produce both positive or negative values the data bits are formatted as two's complement. Care must therefore be taken when the 14-bit value is extended to a 16-value to maintain the two's complement property. This is done by copying the value of bit 14 to bit 15 and 16.

ADC To drive the conversion process the ADC needs a clock signal between 50 kHz and 200 kHz to obtain the maximum 10 bit resolution. The clock signal is based on the system clock running at 16 MHz and must be scaled to fit the specified range. By setting the prescale factor to 128 the resulting clock will run at 125 kHz.

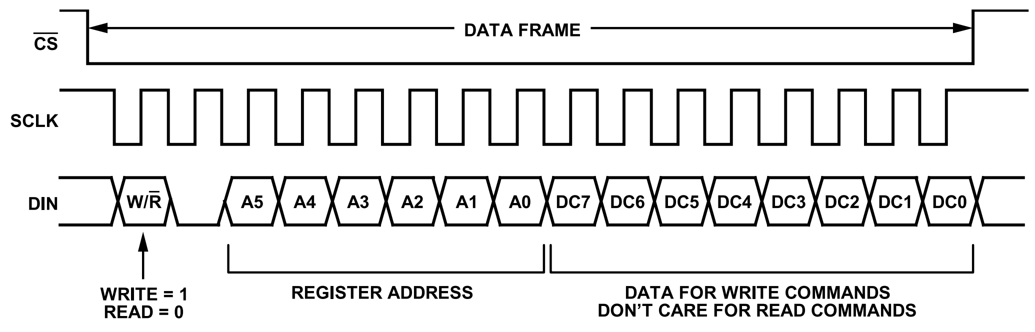


Figure 5.14: IMU request data frame ©Analog Devices, Inc.

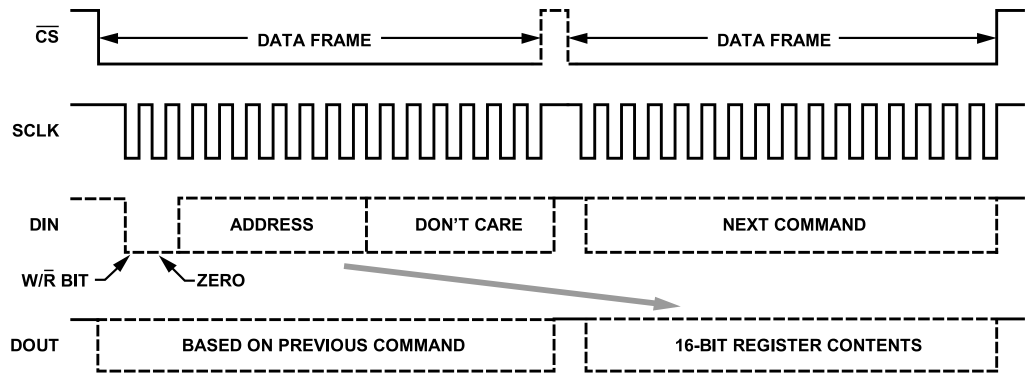


Figure 5.15: SPI sequence for read command ©Analog Devices, Inc.

The ADC is by default in Single Conversion Mode such that each conversion must be started by the master interrupt routine. When a conversion result is ready the high byte is read first and then the low byte. This is because the MCU locks the ADC output registers when the high byte is read and unlocks when the low byte read is completed.

Main Routine

The main routine sets up the different firmware subsystems before entering the system's `while()` loop. For each run of the loop the routine checks for newly received CAN messages, performs corresponding tasks and finally enters sleep mode. When an interrupt is generated the routine awakes and start at the top of the loop.

When CAN messages are received they perform one of the following commands:

- MSG_RESET: Reset the MCU by using a watchdog timer.
- MSG_CONFIG_COMPASS: Configure different compass features, run calibration routines, etc. Desired action is stored in the CAN message.
- MSG_CONFIG_IMU: Configure different IMU features, run calibration routines, etc. Desired action is stored in the CAN message.
- MSG_CONFIG_SAMPLING: Change the rate of the master interrupt routine. New sample rate is stored in the CAN message.

5.3 Software

The use of the Nokia/Trolltech Qt framework has greatly influenced the design of the software. This framework was chosen because of its inherent platform portability (Mac OS X, Windows, Linux/X11, Symbian, etc.), rich set of features and simple development environment.

Central to the use of Qt is the concept of "signals" and "slots". These are mechanisms provided by the base class QObject and is a Qt hallmark providing loosely coupled components. This makes it easier to stay true to the desired design principle "loose coupling strong cohesion".

Signals are emitted when certain events occur, these events can be user defined or related to component activities (e.g. mouse events). Slots are the signal's counterpart and are defined as functions that can take arguments passed by the signal. These two mechanisms can be defined for a subclass of QObject and used to connect subclass instances together. Each instance has its own received-signal queue. Communication is therefore asynchronous and offers great cross-component flexibility.

The software design is based on two worker threads that gather and process data from the embedded system. One thread is responsible for receiving CAN messages, reassembling the data content and sending notifications for new arrivals. The other thread processes the new values into valid measurements and combines them into an estimate of the ROV state.

Each complete sample from the embedded system consists of a package triplet. For each package type, a corresponding processing routine is invoked:

- Package one: Accelerometer data is scaled and transformed to BODY-frame. Roll/pitch angles are estimated
- Package two: Gyroscope data is scaled and transformed to BODY-frame.
- Package three: Depth is calculated from pressure and compass heading is converted from $[0.0, 359.9]$ to $\langle -\infty, +\infty \rangle$. Temperature is scaled.

When a triplet is completed the state estimator is run, converting measurements from BODY-frame to T-frame, running the Kalman filter and emitting the results.

To obtain user defined search paths a navigation interface (NavigationScene) is designed to accept waypoints relative to a virtual ROV. A user will select where waypoints should be placed and then prompted to insert desired depth.

When the estimated state changes the navigation interface is updated to reflect the true ROV state. By treating the virtual ROV state as the true state, setpoints are generated based on the estimated position and transmitted to the Controller.

Several actions are triggered by the arrival of new data. As shown in Figure 5.16, a cascade of signal/slot interactions starts after a CAN message is received. The sequence diagram depicts the following set of events:

- CANbus thread receives the last message of a triplet sent from the embedded system and extracts the data. Filter is notified that the third message is received and that the triplet is complete.
- Filter preprocesses and transforms the received data and starts the state estimator, emitting the results to the Controller.
- NavigationScene registers that a new waypoint is hit and generates a new setpoint for the Controller.
- Controller receives current state and setpoint to calculate control inputs based on the control criteria.

The QObject subclass QTimer is used to emit periodic signals at user defined intervals. Two such timers are used in the software design. One timer to control the rate at which new frames are captured from the video source and one timer to control the rate at which control inputs are sent to the ROV (control timer).

Figure 5.17 shows a smaller control cascade that is triggered by the control timer. At each timeout the latest control input is sent to the ROVCOM class where it is formatted and transmitted serially.

By looking at Figure 5.18 the class relationships can be observed. All classes are in some fashion related to the MainWindow which is the class responsible for the GUI. Every class made for this application is subclassed from a Qt class. The external classes are open, made by a third-party and designed for use with the Qt framework. Qwt (Qt Widgets for Technical Applications) and QextSerialPort provide GUI components (e.g. plots, dials) and serial communication, respectively.

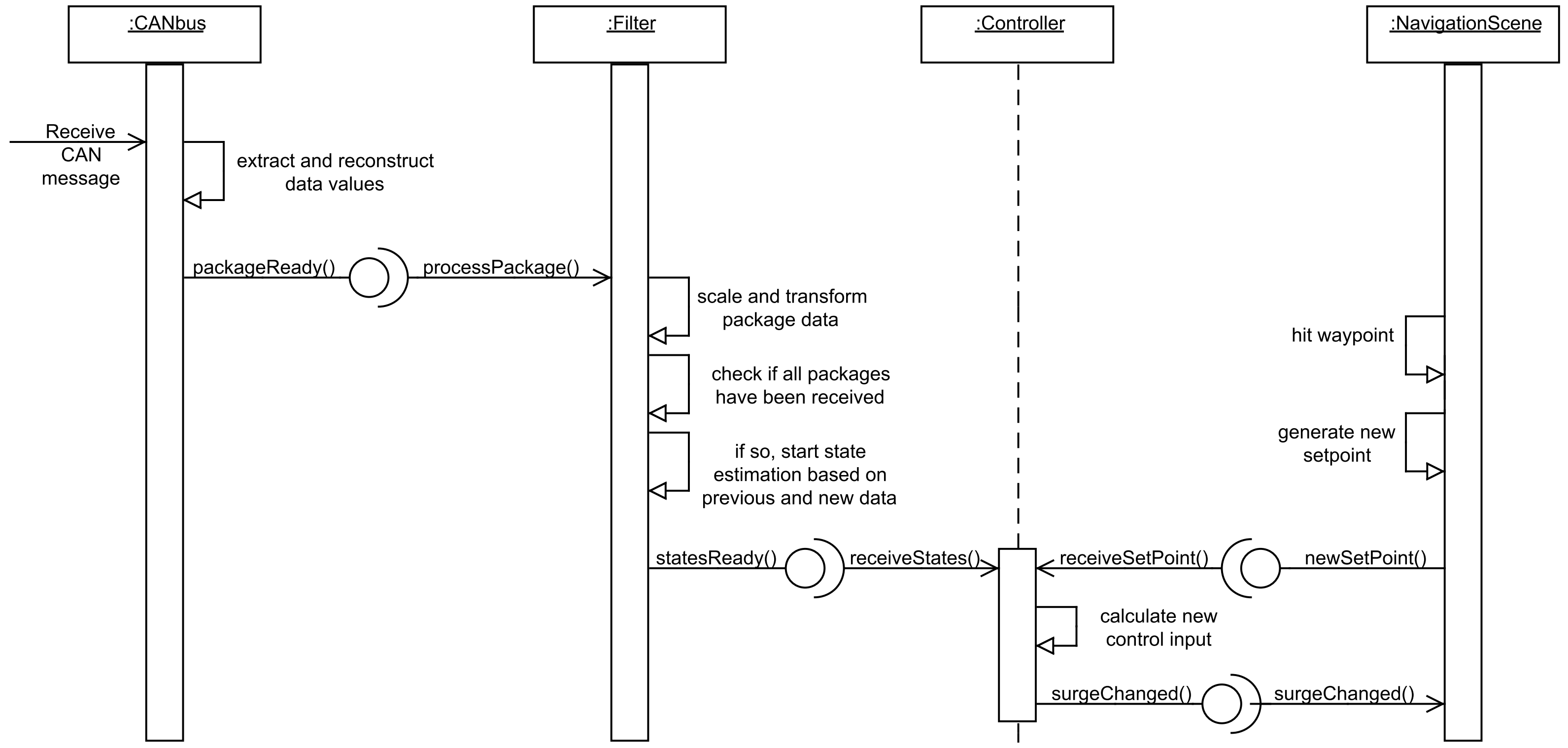


Figure 5.16: UML sequence diagram: All class instances are already initialized and a CAN message is received in the CANbus thread. Showing the resulting cascade from CANbus to Controller. “Socket” indicates signal/slot interface.

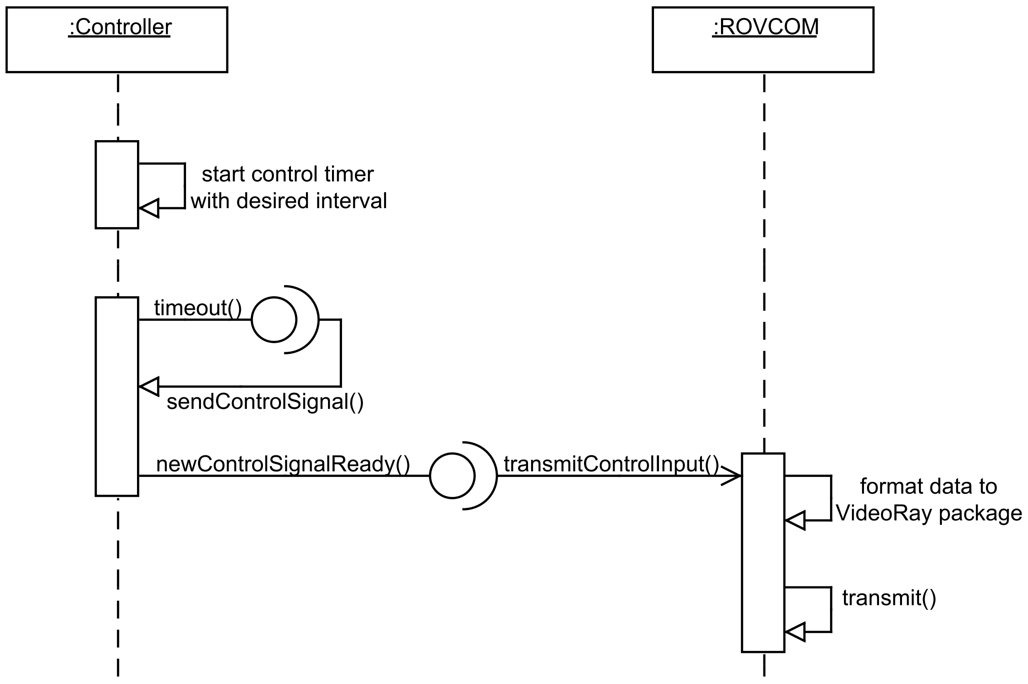


Figure 5.17: UML sequence diagram: All class instances are already initialized and the control timer is started. Showing the first timeout cascade from the Controller. “Socket” indicates signal/slot interface.

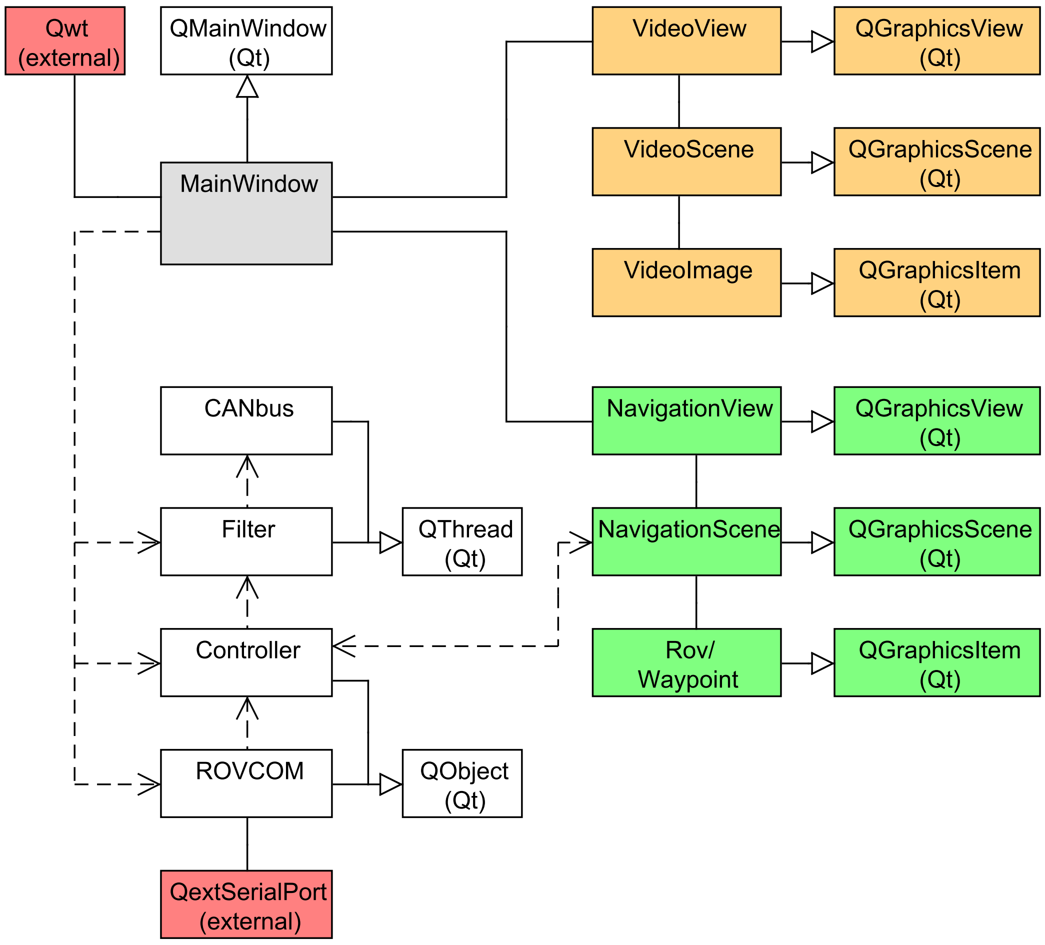


Figure 5.18: UML class diagram: Showing the relationship between the designed classes, Qt classes and the external classes (red). Solid lines with triangular arrow indicates superclass. Solid lines show association. Dashed lines show dependency in the direction of the arrow (signal/slot dependency). Video interface is colored orange, navigation interface green.

5.4 State Estimator

A Kalman filter is used to estimate the attitude of the ROV. Horizontal position (x_t,y_t) is calculated from the estimated yaw and the assumption of constant and unbiased forward speed. Depth (z_t) is based on the pressure measurements.

The model used in conjunction with the filter is simpler than the INS error model described in Section 2.5. Instead of estimating the errors of each state, the states are estimated directly, also known from (Maybeck 1979) as *direct* integration. The IMU error model is also made simple by modelling the sensor error with a single bias term and a time constant. The filter model for a single gyroscope can be expressed as:

$$\begin{aligned}\dot{\theta} &= \omega_{\text{imu}} + b + w_1\\ \dot{b} &= -\frac{1}{T}b + w_2\\ y &= \theta + v\end{aligned}$$

In filter context the rotation rate ω_{imu} is regarded as the input and y as the measurement. These values will be combined by the filter's optimal blend factor and result in an estimate of the angle state θ. The measurement y is the angle based on accelerometer data which is determined in the same manner as in INS alignment Equation (2.6). All IMU readings are transformed from S to BODY before use.

The complete continuous time state model is derived for roll, pitch and yaw:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu} + \mathbf{w}\\ \mathbf{y} &= \mathbf{Hx} + \mathbf{v}\end{aligned}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{b}_1\\ \dot{\theta}\\ \dot{b}_2\\ \dot{\psi}\\ \dot{b}_3\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0\\ 0 & -\frac{1}{T_1} & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & -\frac{1}{T_2} & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{T_3}\end{bmatrix}\begin{bmatrix}\phi\\ b_1\\ \theta\\ b_2\\ \psi\\ b_3\end{bmatrix} + \begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\\ 0 & 0 & 1\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\omega_x^*\\ \omega_y^*\\ \omega_z^*\end{bmatrix} + \begin{bmatrix}w_1\\ w_2\\ w_3\\ w_4\\ w_5\\ w_6\end{bmatrix}$$

$$\begin{bmatrix}y_1\\ y_2\\ y_3\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\phi\\ b_1\\ \theta\\ b_2\\ \psi\\ b_3\end{bmatrix} + \begin{bmatrix}v_1\\ v_2\\ v_3\end{bmatrix}$$

where ω_x^*, ω_y^* and ω_z^* are the gyroscope measurements transformed to T by $\mathbf{R}_b^t$.

To make the model compatible with the recursive Kalman filter it must be discretized by employing Equation (2.9) and (2.11). The resulting discrete matrices:

$$\boldsymbol{\phi}_k = e^{\mathbf{A}\Delta T} = \begin{bmatrix} 1 & T_1\left(1-\mathrm{e}^{-\frac{\Delta T}{T_1}}\right) & 0 & 0 & 0 & 0 \\ 0 & \mathrm{e}^{-\frac{\Delta T}{T_1}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T_2\left(1-\mathrm{e}^{-\frac{\Delta T}{T_2}}\right) & 0 & 0 \\ 0 & 0 & 0 & \mathrm{e}^{-\frac{\Delta T}{T_2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T_3\left(1-\mathrm{e}^{-\frac{\Delta T}{T_3}}\right) \\ 0 & 0 & 0 & 0 & 0 & \mathrm{e}^{-\frac{\Delta T}{T_3}} \end{bmatrix}$$

$$\boldsymbol{\Delta}_k = \left(\int_0^{\Delta T} e^{\mathbf{A}\tau}\, d\tau\right)\mathbf{B} = \begin{bmatrix} \Delta T & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \Delta T & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Delta T \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{H}_k = H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The sample time ΔT will depend on the user specified update rate.

The process- and gyroscope covariance can be obtained by performing noise analysis on the IMU output. An accurate determination of these factors will improve the filter's ability to weight the measurements correctly. In this design process the factors were roughly adjusted through experiments, providing acceptable results. The diagonal of $\mathbf{Q}$ was set to 0.015 while $\mathbf{R}_k = \mathrm{diag}(0.135, 0.135, 0.5)$, $T_i = 5.0$.

Horizontal position measurements are only available through double integration of the accelerometers. Even though this was tested with trapezoidal integration the results were far from acceptable. Experimental data is displayed and discussed in Chapter 7.

An alternative method is developed that is based on the estimated yaw and controller information. The surge control input is multiplied with the sample time and a scaling factor to obtain a measure of traveled distance. This value is then propagated in the yaw direction of the ROV. To produce correct estimates this procedure requires a constant forward-speed, a good yaw estimate and no transverse disturbances.

Depth is determined by converting the ADC value to its corresponding pressure measurement and applying Equation (2.4):

$V_{\text{in}} = \dfrac{\text{ADC} \cdot V_{\text{ref}}}{1023}$	Based on Equation (5.2), $V_{\text{ref}} = 2.56\,\text{V}$.
$V_{\text{pressue}} = \dfrac{V_{\text{in}}}{50}$	Voltage from the pressure transducer is amplified by a factor of 50.
$p_{\text{psi}} = \dfrac{V_{\text{pressue}} \cdot 100\,\text{PSI}}{50\,\text{mV}}$	Transducer maximum pressure and output voltage.
$p_{\text{d}} = p_{\text{psi}} \cdot 6894\,\text{Pa}$	$1\,\text{PSI} = 6894\,\text{Pa}$
$z_t = \dfrac{p_{\text{d}} - p_{\text{atm}}}{\gamma}$	Based on Equation (2.4), p_{atm} is the pressure measured at surface level, $\gamma = 10050\,{}^{\text{N}}\!/_{\text{m}^3}$

$$z_t = \frac{\text{ADC} \cdot 2.56\,\text{V} \cdot 100\,\text{PSI} \cdot 6894\,\text{Pa}}{1023 \cdot 50 \cdot 50\,\text{mV} \cdot 10050\,{}^{\text{N}}\!/_{\text{m}^3}} - \frac{p_{\text{atm}}}{10050\,{}^{\text{N}}\!/_{\text{m}^3}} \tag{5.3}$$

An overview of the different estimation steps is displayed in Figure 5.19.

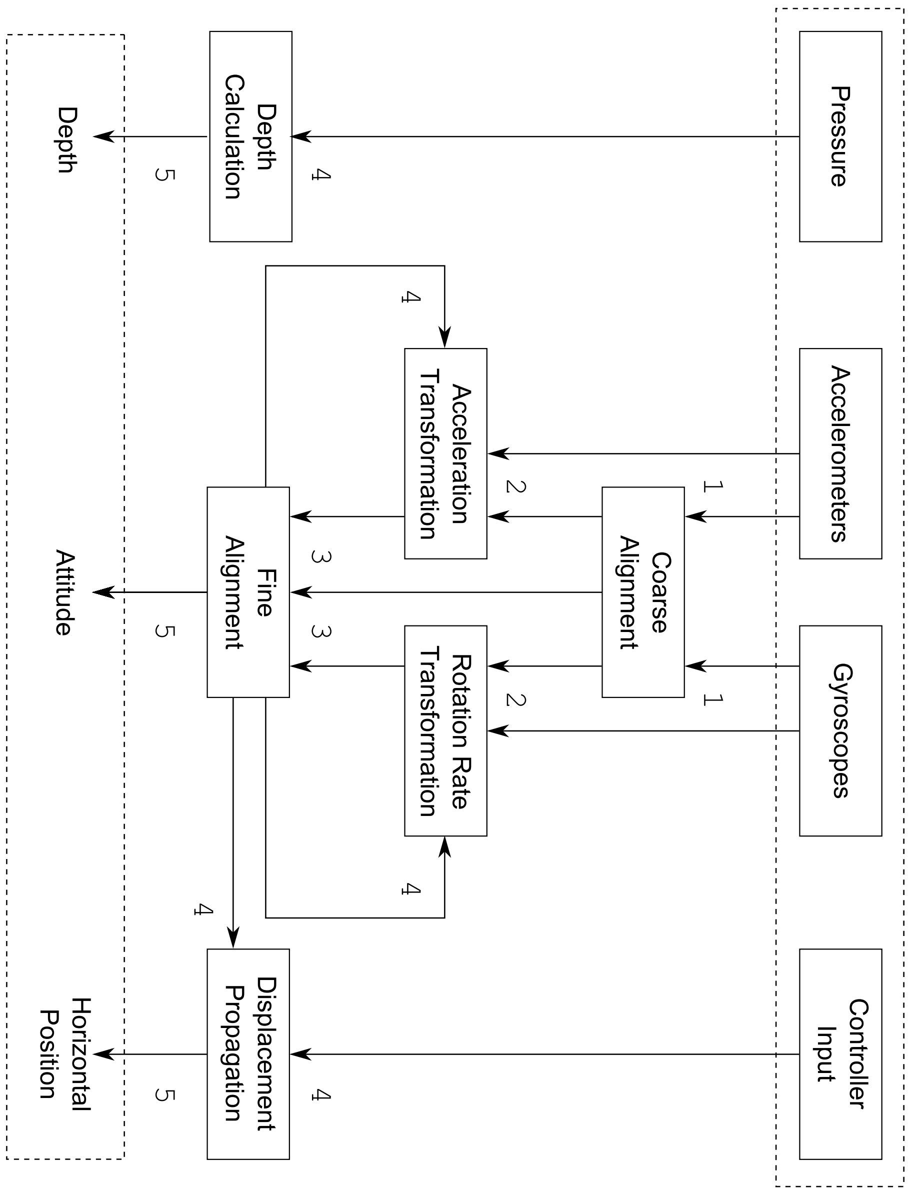


Figure 5.19: The different steps in estimating the system states. The numbering indicates the order of execution.

5.5 Controller

The control objective of the ROV is to follow a user specified path decomposed in T, independently of time. This is referred to as path-following in (Fossen 2002) and is closely related to trajectory tracking, where positions are dependent on time.

The specification of such a path is the responsibility of a guidance system. The chosen method of guidance in this application is based on the use of waypoints. These are positions in the navigation frame that are assigned three-dimensional coordinates and a radius of intersection. A complete path can then be generated by calculating sub-paths between waypoint based on a set of geometric rules. Commonly used rules include:

- Piece-wise continuos interpolation: Straight lines between each consecutive waypoint.
- Circles and straight lines: Straight lines between each consecutive waypoint and a user specified turning circle close to the waypoint.
- Continuous interpolation with splines: A piecewise polynomial curve between the defined waypoints.

Samples of the three types are displayed in Figure 5.20. The piece-wise continuos interpolation is used in the current application.

The intersection radius is used as a proximity threshold which govern the current control objective. When the radius is intersected by the ROV the subsequent path-segment is chosen as the new control objective. During such an event the forward speed is reduced to zero until the yaw error is less than 45°.

Setpoints are based on the estimated states and the current control objective. An accurate solution would be to tightly follow the path by controlling the heading towards intermediate points on a path segment. One such technique is known as Line-of-Sight (LOS) path-following and is described in (Fossen 2002). The chosen solution is simpler and provides acceptable results.

The heading setpoint ψ_d is generated from the current ROV position relative to the waypoint at the end of the current path segment. x and y coordinates of the ROV and waypoint are decomposed in T and used in conjunction with a trigonometric function:

$$\psi_d = \text{atan2}(y_{\text{wp}} - y_{\text{rov}}, x_{\text{wp}} - x_{\text{rov}}); \tag{5.4}$$

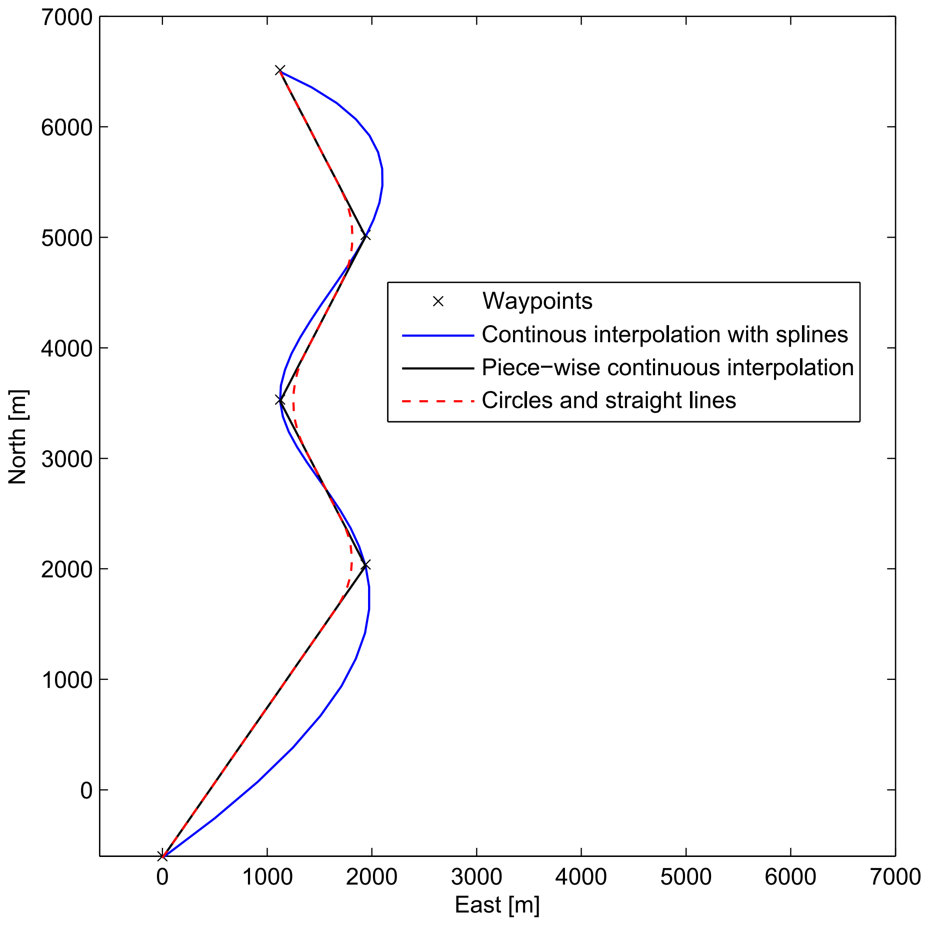


Figure 5.20: Waypoint-based path types.

Yaw error can then be defined as the difference between the setpoint and estimates yaw angle:

$$e_\psi = \psi_d - \hat{\psi} \tag{5.5}$$

The depth setpoint is specified by the user and compared with the estimated depth:

$$e_z = z_d - \hat{z} \tag{5.6}$$

Two independent controllers are used to compensate for these errors while a third is designed to generate a constant forward speed:

- The vertical thruster is controlled to compensate for e_z:

$$u_{vertical}(t) = -\left[K_{pz}e_z(t) + K_{iz}\int_0^t e_z(\tau)\,d\tau\right] \tag{5.7}$$

$u_{vertical}(t)$	Vertical thruster control input.
K_{pz}, K_{iz}	Proportional- and integral gain in heave direction.

- Port and starboard thrusters can be actuated symmetrically to compensate for e_ψ. The propellers are mounted in opposite directions and a positive control input will make the thrusters turn in the same direction. If two identical control inputs are fed to the port and starboard actuators the vehicle will change its yaw.

$$u_{starboard}(t) = -\left[K_{p\psi}e_h(t) + K_{i\psi}\int_0^t e_h(\tau)\,d\tau\right] \tag{5.8}$$

$$u_{port}(t) = -\left[K_{p\psi}e_h(t) + K_{i\psi}\int_0^t e_h(\tau)\,d\tau\right] \tag{5.9}$$

$u_{startboard}(t)$	Starboard thruster control input.
$u_{port}(t)$	Port thruster control input.
$K_{p\psi}$, $K_{i\psi}$	Proportional- and integral yaw gain.

- To generate a constant surge speed a constant term can be added to both port and starboard thruster. By considering the propeller configuration the term must be added to one of the thrusters and subtracted from the other:

$$\begin{aligned} u_{starboard}(t) &= K_{px} \\ u_{port}(t) &= -K_{px} \end{aligned}$$

K_{px}	Proportional surge gain.

When considering all the terms the port and starboard thrusters can be controlled with the following PI-controllers:

$$u_{starboard}(t) = -\left[K_{p\psi}e_h(t) + K_{i\psi}\int_0^t e_h(\tau)\,d\tau\right] + K_{px} \tag{5.10}$$

$$u_{port}(t) = -\left[K_{p\psi}e_h(t) + K_{i\psi}\int_0^t e_h(\tau)\,d\tau\right] - K_{px} \tag{5.11}$$

Integral terms of the two controllers will compensate for the constant error that can arise from tether pulling/springing.

Chapter 6

Implementation

6.1 Hardware

Components were mainly bought over the Internet and some were available from the institute's stock. Table 6.1 shows a list of the system components.

Component Type	**Product**	**Source**
CAN Adapter	Kvaser Leaf SemiPro HS	ELFA.se
CAN Transceiver	Microchip MCP2551-I/SN	Farnell.com
Compass	OceanServer Technology OS4000-T	SparkFun.com
DC-DC Converter	Traco Power TEN 5-4811WI	ELFA.se
Ferrite Bead	Murata BLM18KG601SN1D	Farnell.com
Frame Grabber	TerraTec Grabby	Pixmania.com
IMU	Analog Devices ADIS16354AMLZ	Newark.com
IMU Connector	Samtec CLM-120-02-L-D	Farnell.com
Instrumentation Amplifier	Burr-Brown INA155UA	Farnell.com
MCU	Atmel AT90CAN32	In stock
Oscillator	Generic 16MHz	In stock
Pressure Transducer	Honeywell S&C 19C100PA4K	DigiKey.com
P.T. Connector	Tyco Electronics AMPMODU MOD II	ELFA.se
Tether Connector	Phoenix Contact MKDSN1,5/4-5.08	In stock

Table 6.1: Alphabetically sorted list over used components.

When ordering the PCB, two service providers were considered. BatchPCB (`www.batchpcb.com`) has low prices, basic production options and an automated service that verifies your PCB design. Because of a 3-4 week waiting period an alternative provider was chosen. MakePCB (`www.makepcb.com`) is slightly more expensive, but offers shorter delivery times and more advanced production options (e.g. 10-layers). The PCB design was exported from CadSoft Eagle as Gerber files and roughly controlled with BatchPCB's automated verification process. Figure 6.1 shows the finished PCB.

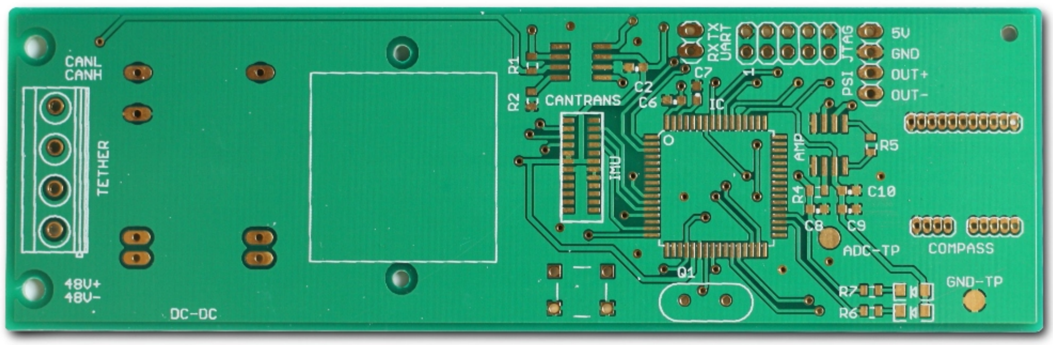


Figure 6.1: PCB card from MakePCB.

The embedded system housing was made out of PVC at the institute's workshop. A brass bracket was made to fasten the PCB to the PVC lid and the whole unit was pressure tested before attaching any system components. Lid, bracket and PCB are shown in Figure 6.2. The assembled navigation module is shown in Figure 6.3.

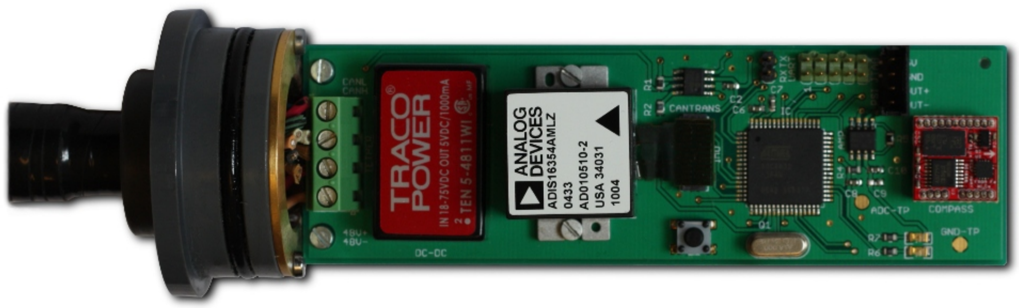


Figure 6.2: PCB card with components and mounted on the housing lid.

Figure 6.4 shows the navigation module connected on the ROV. Nylon cable ties were used for mounting.

The ICB - computer adapters are shown in Figure 6.5.

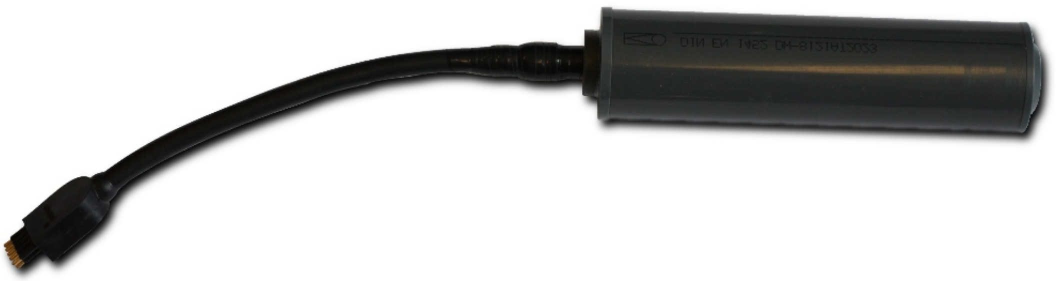


Figure 6.3: Completed navigation module.

The largest predefined CAN transmission speed that was compatible with the tether length was found experimentally to be 250 kbit/s. By choosing one of the predefined transmission speeds (1 Mbit/s, 500 kbit/s, 250 kbit/s, 125 kbit/s, etc.) the process of finding custom timing parameters was avoided. If the speed is set to 250000 bit/s, messages have a 92 bit length and each complete sample consist of three messages the theoretical sample rate will be 905 samples/s.

For a complete specification of the navigation module's connector, signals, CAN messages and operating ranges refer to Appendix B.

6.2 Firmware

The firmware was developed in AVR Studio 4.18 together with GNU GCC compilers and libraries included with WinAVR 20100110. Initial programming and debugging made use of the embedded system's JTAG interface by connecting an Atmel AVR JTAGICE mkII. This product is compatible with AVR Studio and is powered and interfaced through USB.

When the initial testing and development was completed, memory programming was set up to be performed over CAN bus. Atmel has already made a bootloader for the AT90CAN32 that uses CAN for data transport and is a part of their AT90CAN128/64/32 Software Library (available at `www.atmel.com`). This code was used with some minor adjustments:

- `config.h`: Change oscillator speed to 16 MHz and define EEPROM flag address/data.
- `main_can_bootloader.c`: Configure bootloader to check startup conditions.

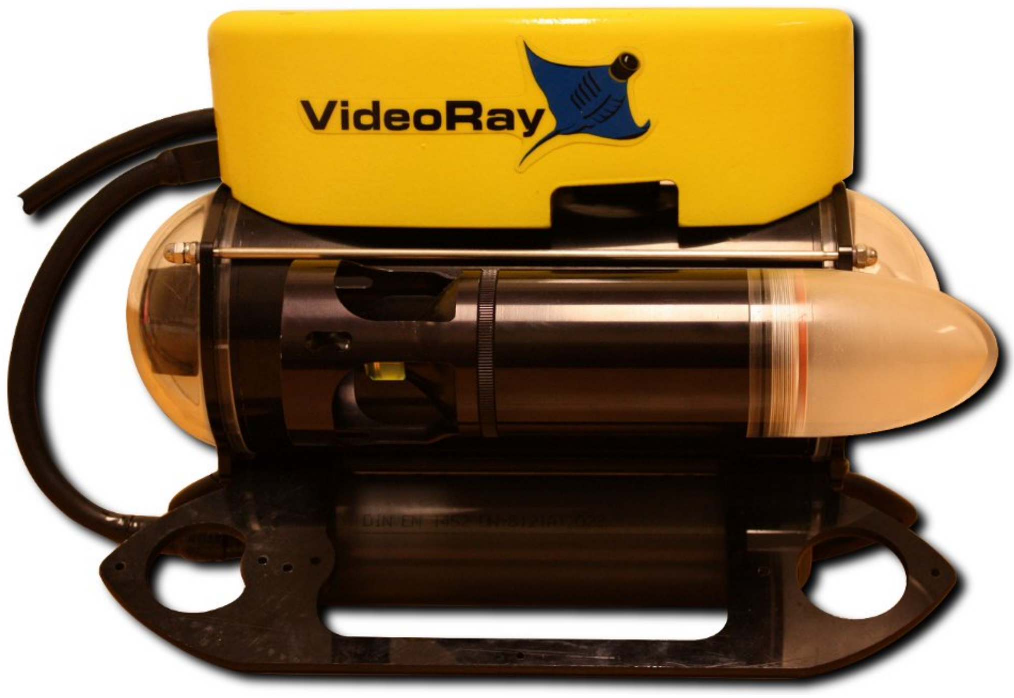


Figure 6.4: Navigation module connected and mounted to the ROV.

- `can_isp_protocol.c`: Configure bootloader to change flag data and reset after a successful reprogramming.

AVR Studio was setup according to the documentation and the bootloader was programmed to the BFS. Further programming of the AFS was done with Atmel's FLIP 3.4.1 or batchisp that are both included in the FLIP installer. The software is designed to work with different nodes by referring to their unique address defined in each node's bootloader. Because there is only one programmable node on the bus the default addresses were used: Address = FF, CRIS = 00. These parameters are further described in the AT90CAN128/64/32 Software Library documentation.

The current version of FLIP has built-in support for several CAN-USB adapters, but not from Kvaser. A PEAK PCAN-USB was therefore used to program the embedded system. Both adapters have the same pinout so both are compatible with the ICB signal splitter.

To update firmware functionality the ordinary firmware was compiled in a separate project and the generated hex-file uploaded with FLIP or batchisp.

The CAN driver used on the AT90CAN32 is developed by Infidigm. The

Figure 6.5: TerraTec Grabby on the left and Kvaser Leaf SemiPro HS on the right.

driver is interrupt based and provides a simple interface to the CAN controller. Source code is found on `www.infidigm.net/projects/avrdrivers/`.

6.3 Software

The software implementation was developed in Qt Creator 1.3.1. Several libraries were used in conjunction with the Qt libraries:

- Qt 4.6.2: The main application framework.
- Qwt 5.2.1: A range of plots, dials and other widgets made for Qt (3rd party).
- QextSerialPort 1.2a: Cross-platform serial port class made for Qt (3rd party).
- OpenCV 2.1.0: Video capturing, Kalman filtering and matrix operations.
- Kvaser CANlib SDK 4.2: Interface to the Kvaser CAN adapter.

GUI

The final application has the GUI shown in Figure 6.6. Each red square is given an index and explained below:

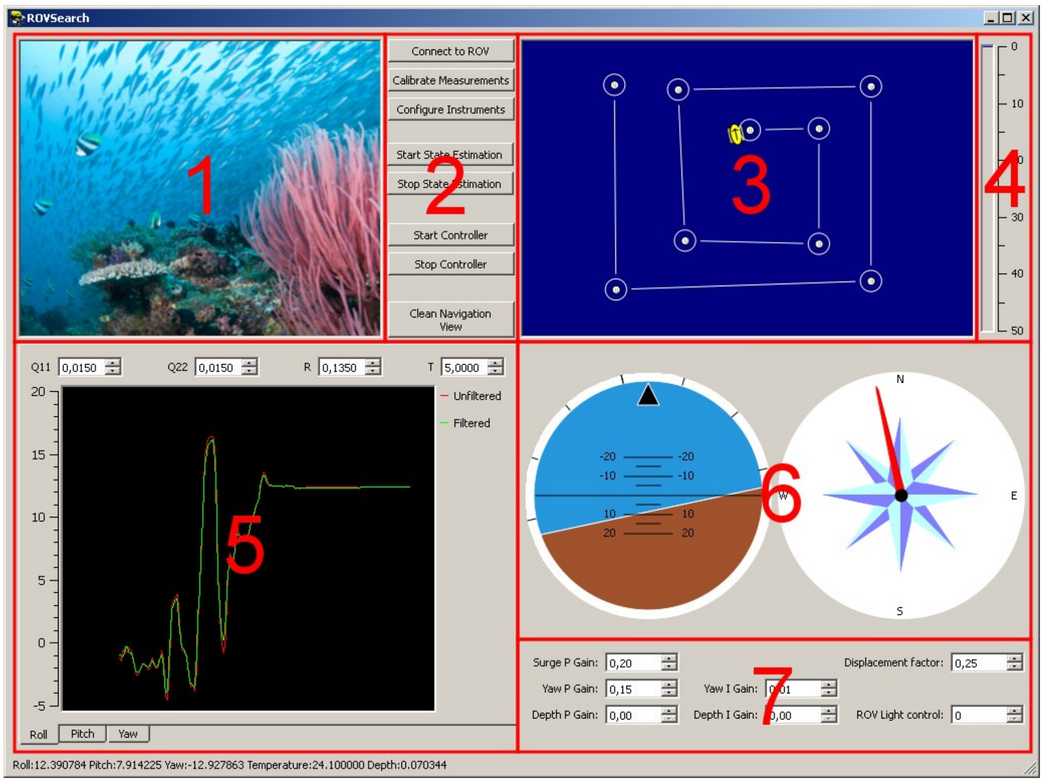
1. Video Interface: Display camera feed from ROV.
2. Software Controls: Buttons for connecting to the ROV, calibrate measurements/instruments, start/stop state estimator and controller, reset navigation module.
3. Navigation Interface: Display estimated position and yaw angle. Receive user specified path.
4. Depth Gauge: Display ROV depth.
5. Filtering Interface: Display the filtered/unfiltered roll-, pitch- and yaw angles. Kalman parameters $\mathbf{Q_k}$, $\mathbf{R_k}$ and T_i can be adjusted in the top controls.
6. Attitude Indicators: Indicate the estimated ROV attitude. Roll and pitch to the left and yaw on the right.
7. Controller Parameters: Set controller gains and surge displacement factor.

Navigation Interface

The navigation interface receives user input by registering mouse events. For each click of the left mouse button the pointer coordinates are stored and the user is prompted for the desired depth. The values are then stored as a waypoint with x, y and z coordinates.

New waypoints are stored in a list together with the previously generated waypoints. Inter-waypoint line segments are added subsequently. A sample path is displayed in Figure 6.7.

The ROV and waypoint models are implemented as subclasses of Qt's QGraphicsItem. This enables built-in collision detection and can be used to detect ROV-waypoint intersections. Since the virtual ROV position is updated with state estimates the behavior observed in the navigation interface is used to determine when a waypoint is reached. When a collision event occurs the controller is notified and the yaw setpoint updated.



Q11 0,0150 Q22 0,0150 R 0,1350 T 5,0000

Surge P Gain: 0,20
Yaw P Gain: 0,15
Depth P Gain: 0,00
Yaw I Gain: 0,01
Depth I Gain: 0,00
Displacement factor: 0,25
ROV Light control: 0

Roll:12.390784 Pitch:7.914225 Yaw:-12.927863 Temperature:24.100000 Depth:0.070344

Figure 6.6: Screenshot of GUI with indices that indicate the different modules.

6.4 State Estimator

OpenCV (Open Computer Vision Library) is used for matrix operations and Kalman filtering. The library is distributed as open source under the BSD-license and designed to be cross-platform. It was originally developed by Intel and therefore utilizes the optimized routines provided by Intel's Integrated Performance Primitives. These routines are supported by a range of processors/platforms and offers increased performance. The OpenCV library v2.1.0 (released 04/2010) was used in this project.

The state estimator is implemented as a subclass of QThread. This enables the estimator routines to run independently of data capturing or GUI processing.

OpenCV's Kalman filter is divided into three parts. One structure that contains all the filter data (struct CvKalman) and two procedures that are called alternately:

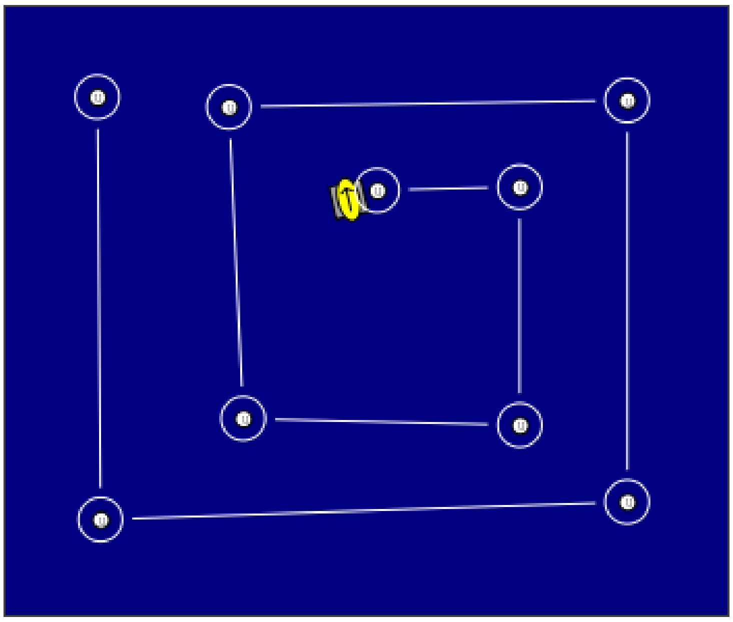


Figure 6.7: Navigation interface with sample path. The arrow on the yellow marker indicates the ROV heading. White markers with orbit indicate waypoints with corresponding intersection radius.

- `cvKalmanPredict(`u_k`)`: Propagate system model and include new control input. Left box in Figure 2.6.
- `cvKalmanCorrect(`z_k`)`: Correct system model with new measurement. Top, right and bottom boxes in Figure 2.6.

Reference-frame transformations and gravity compensation can be performed with the OpenCV function `cvGEMM()` where GEMM stand for Generalized Matrix Multiplication.

6.5 Controller

The control input update rate is governed by a timer as shown Figure 5.17. The implementation can be found in the Controller class.

Chapter 7

Experiments

7.1 Experimental Setup

Several tests were performed to measure the combined properties of the navigation system and ROV. Initial testing was carried out on a subsystem basis before mapping the combined system performance. Accelerometer and gyroscope measurements were taken on land (dry subsystem tests) while depth measurements and controllers were tested in a water tank at Tyholt in Trondheim (wet subsystem tests). To efficiently benchmark the system's ability to perform search patterns a camera was used to capture the ROV motion and estimate real world positions (search pattern tests). Paths, states and control inputs were logged during each run.

Dry Subsystem Tests

Accelerometers and gyroscopes were tested independently without filtering or external influences (except from earth rotation and gravitational forces) by placing the ROV on a steady surface. Gravitational acceleration was subtracted from the accelerometers measurements and the result was integrated twice to obtain a measure of displacement along the three axes. Gyroscope measurements were integrated once to obtain ROV attitude displacement.

Accelerometer and gyroscope data are combined in a Kalman filter to test the estimated roll- and pitch angles. In the first test the ROV was stationary, in the second test the ROV was exposed to periodic linear- and rotational motion.

The compass is calibrated for hard iron effects and the 360° range is compared to an ordinary liquid compass.

Wet Subsystem Tests

Stationary depth measurements were taken in the approximately 1 m deep water tank. Depth measurements were also performed during fast movements at a fixed depth.

The yaw controller was tested with several configurations of the proportional- and integration gains. Setpoints were added so that the ROV experienced a 90° error in yaw attitude.

Search Pattern Tests

Search patterns are limited to a plane defined by the water surface. This choice makes ROV video tracking possible and simplifies the calibration/ transformation procedures. True and estimated positions were captured during two types of maneuvers:

- Follow a straight line and make a sharp turn to travel the same distance in opposite direction.
- Follow a simple, geometric path with approximately 90° turns in yaw.

Paths are defined by waypoint data entered in the navigation interface.

A webcam of the type Logitech QuickCam Messenger was used to capture the ROV motion. The webcam was not mounted directly above the test area, but at an angle, facing along the length of the water tank. This was done because of limitations in cable length and the camera's angle of view. Visual perspective skews the true motion of the ROV and must be compensated for.

A camera calibration routine (Appendix C) was initially run to map the relationship between camera- and real world coordinates. Four markers were placed on the water tank walls as show in Figure 7.1 to establish a real world geometrical figure. Each side of the square is approximately 2.5 m and coplanar.

The real world coordinate system is locally referenced to the water tank and not related to the T-frame. Orientation is therefore not relative to north, but in the negative direction of the X-axis. The coordinate system is right-handed, has its origin at the lower left marker, X-axis along the tank and

Y-axis across the tank. The camera coordinate system is also right-handed with its origin in the top-left image corner, u-axis along its horizontal edge and v-axis along the vertical edge. Both coordinates systems are shown in Figure 7.1.

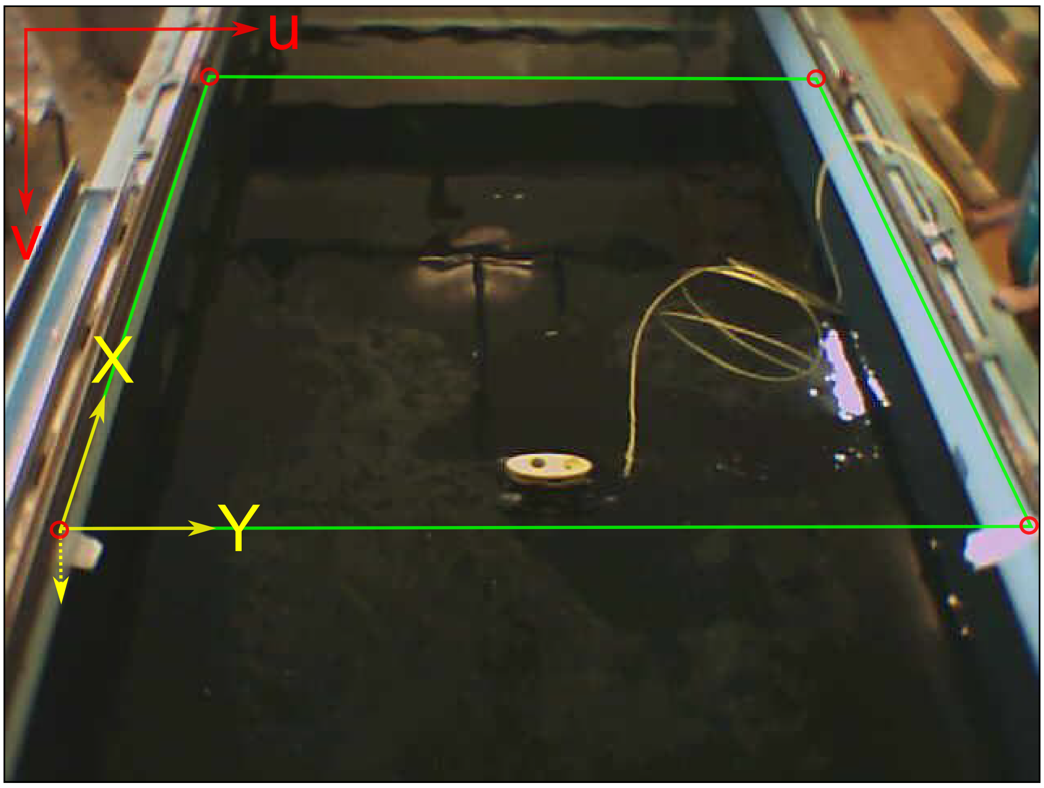


Figure 7.1: The red circles indicate the four markers used in the coordinate mapping procedure. The green square indicates the resulting plane.

Each VideoRay maneuver was recorded after the camera had been calibrated. One feature that was distinct during the setup at Tyholt was the ROV's strong color intensity when compared to the surrounding water. By converting each video frame to the HSV (Hue, Saturation, Value) color space (described in (Joblove & Greenberg 1978)) the image intensity was available through the V channel. Intensity information was then thresholded to a level that made the ROV stand out as shown in Figure 7.2. These operations are available through the OpenCV functions `cvCvtColor()` (convert to HSV) and `cvThreshold` (threshold V channel) both described in (Bradski & Kaehler 2008).

To further track the ROV a mean-shift tracking algorithm was used. In short the algorithm will converge a search window to the body's center of

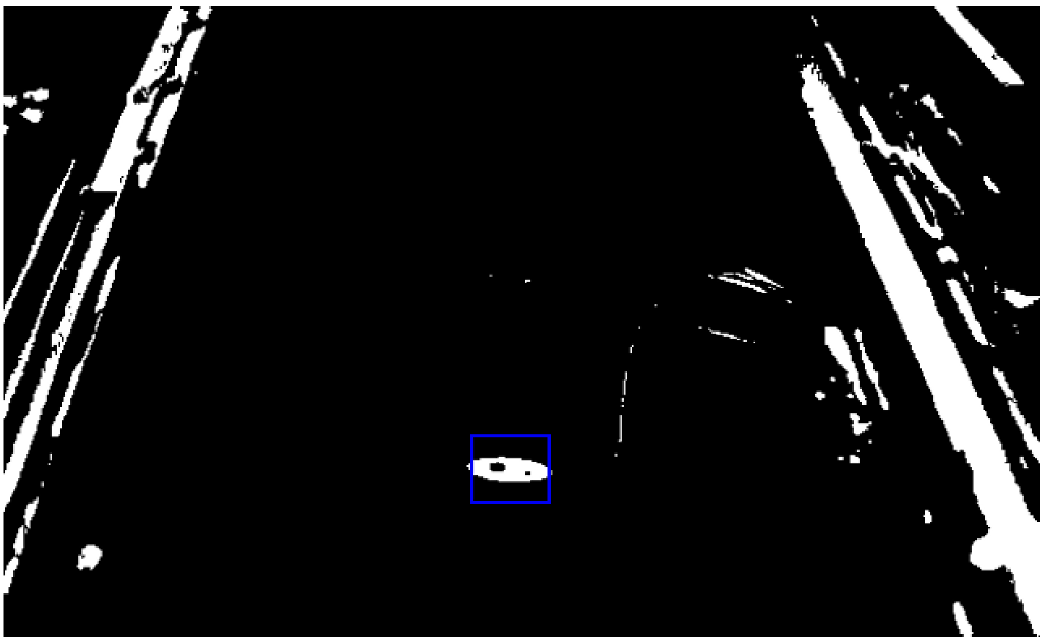


Figure 7.2: V-channel of sample frame. The blue square indicates the mean-shift search window.

mass for each new frame (described in (Bradski & Kaehler 2008)). The center coordinates of this search window will then act as a measure of the ROV's position. This operation is performed with the function `cvMeanShift()`. A sample search window is shown in Figure 7.2 represented by a blue square.

The coordinates for each frame can be transformed to real world coordinates by using the calibration data and Equation (C.9).

7.2 Results

Unfiltered IMU Tests

Unfiltered accelerometer- and gyroscope measurements are sampled at 10 Hz while the ROV and navigation module is stationary. Acceleration data is integrated twice to produce corresponding velocity- and displacement data as shown in Figure 7.3. Gyroscope data is integrated once to obtain corresponding attitude data as shown in Figure 7.4. Numerical integration is performed with the trapezoidal rule.

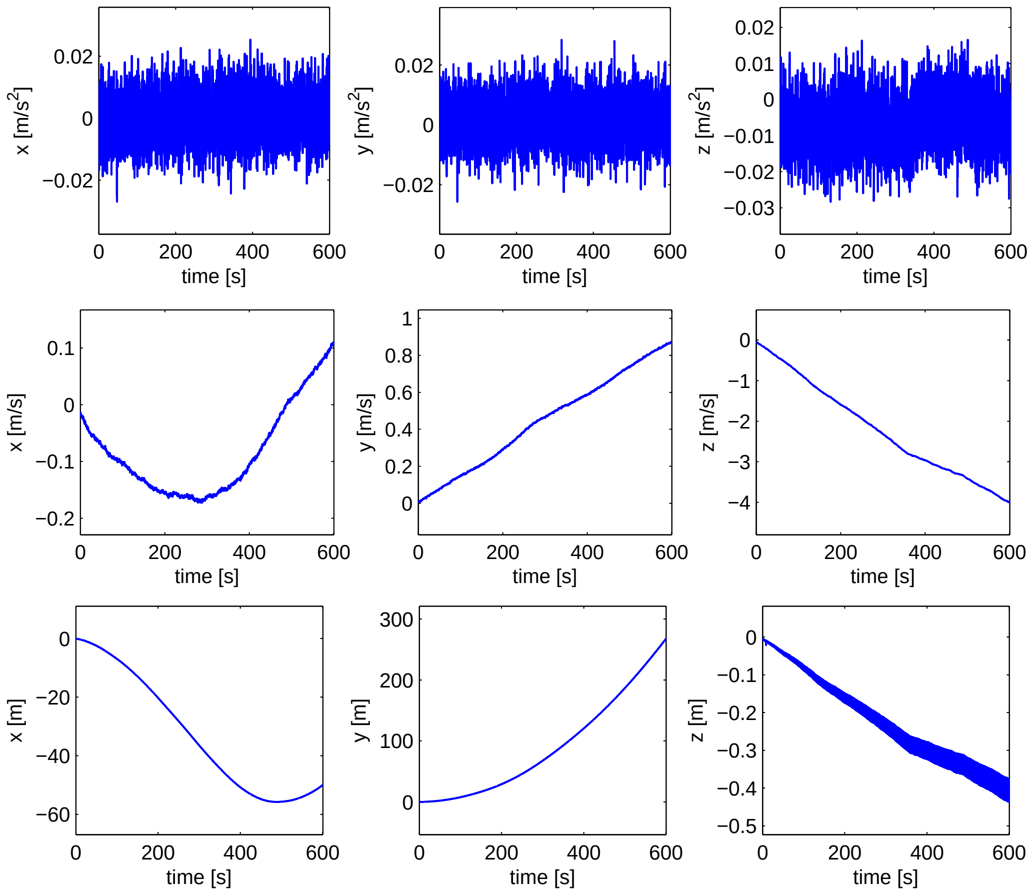


Figure 7.3: Unfiltered, stationary IMU test. First row: Unfiltered acceleration measurements where the gravitational acceleration is subtracted from each axis. Second row: Integrated velocities based on data from the first row. Third row: Integrated displacement based on data from the second row.

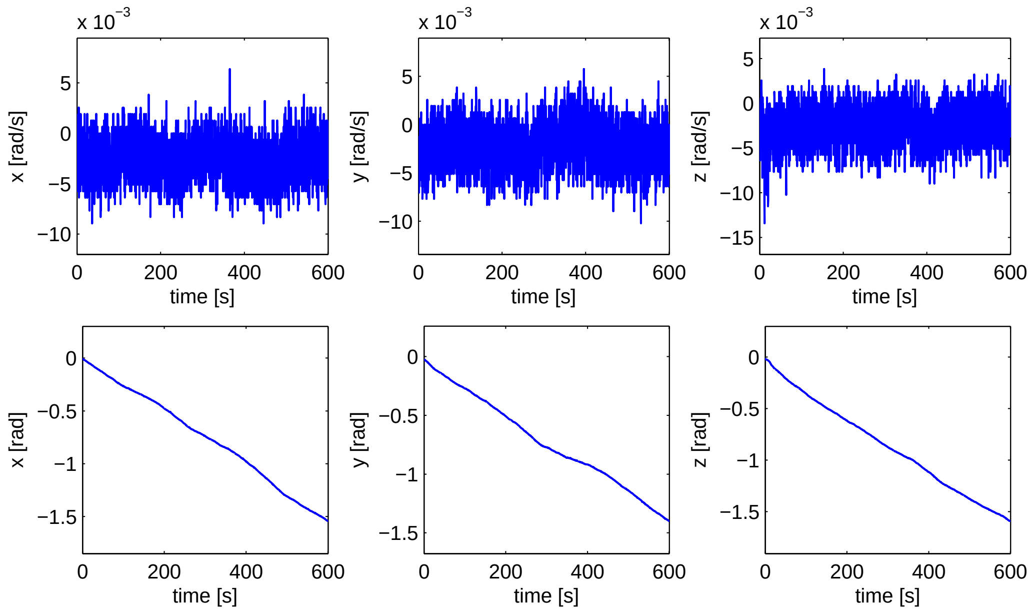


Figure 7.4: Unfiltered, stationary IMU test. First row: Unfiltered angular rate measurements for each axis. Second row: Integrated attitude based on data from the first row.

Filtered IMU Tests

Accelerometer and gyroscope data are combined in a Kalman filter to produce non-drifting roll and pitch angles. During the stationary test the filtered angles were recorded in parallel. These are shown in Figure 7.5.

The filtered yaw angle was also tested while stationary as shown in Figure 7.6.

To ROV was exposed to linear and rotational motion to observe the filter performance. Roll and pitch angles based on accelerometer data are compared with integrated gyroscope data and the filtered angles. A comparison is shown in Figure 7.7.

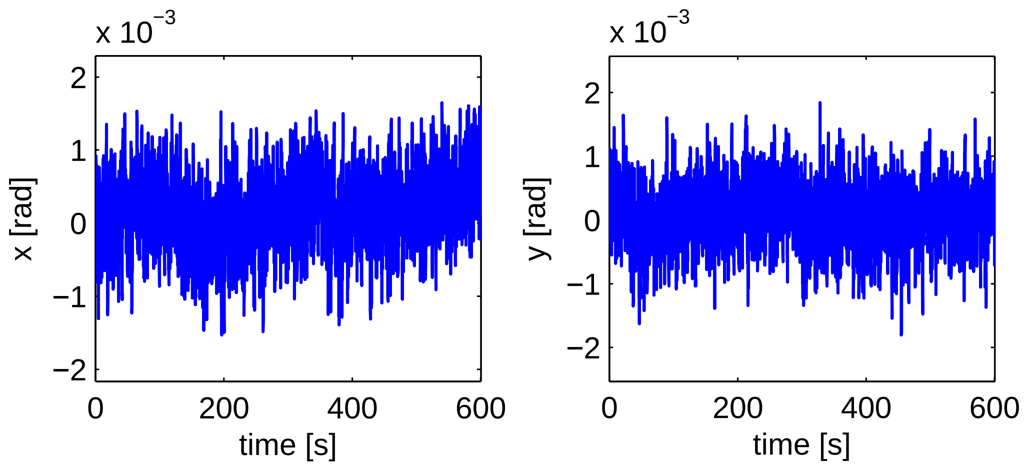


Figure 7.5: Stationary filter test. Filtered roll- and pitch angles centered to zero.

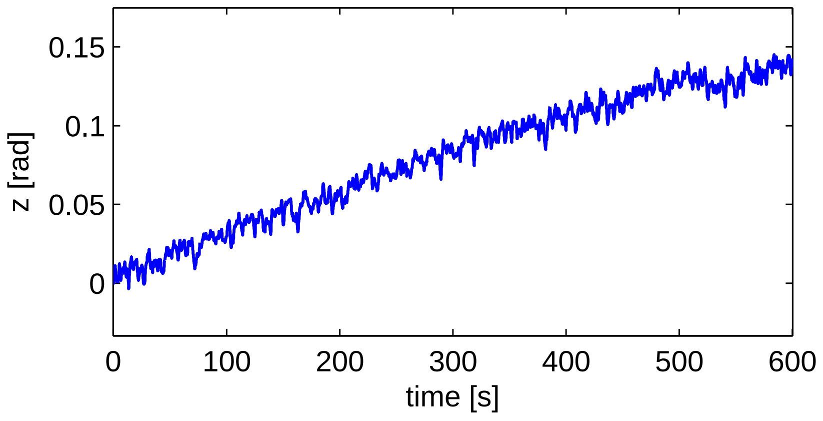


Figure 7.6: Stationary filter test. Filtered yaw angle centered to zero.

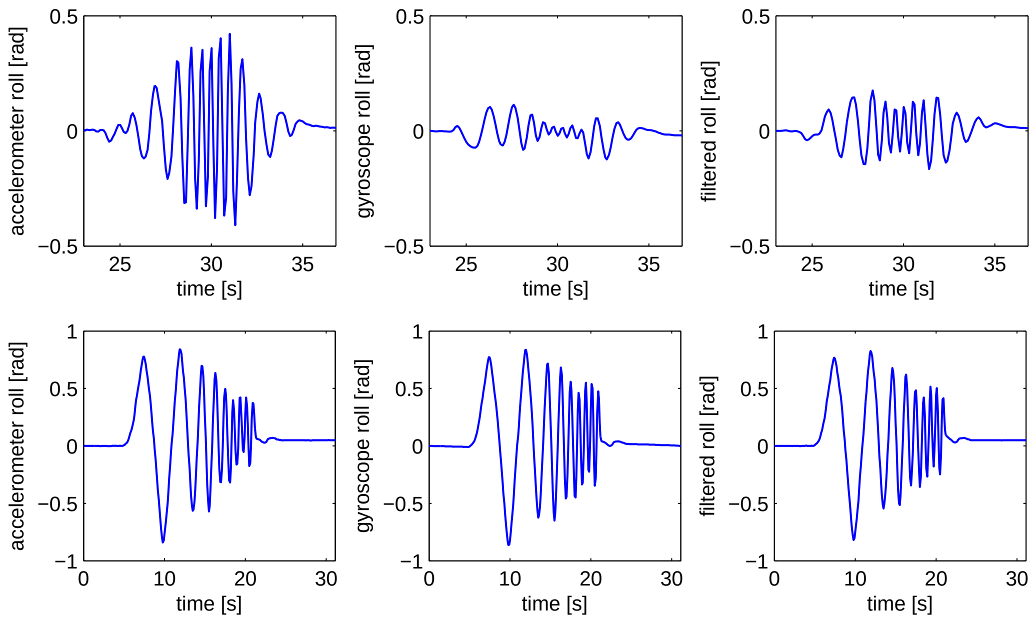


Figure 7.7: Non-stationary test of filtered roll angle. First row: linear oscillations along y_b producing roll excitations in accelerometer, gyroscope and filter calculations. Second row: rotational oscillations along x_b producing excitations in accelerometer, gyroscope and filter calculations.

Compass Test

The test was performed indoors on a stable, rotatable surface. After a successful calibration the digital compass output was consistent with that of the handheld liquid compass. However, when the module was removed from its place of calibration the readings became unreliable. Certain parts of the range were consistent, but other parts were sporadic.

Depth Measurements

The navigation module was lowered to four known depths and the corresponding sensor readings were recorded. Output:

Depth	**Measurement**
0.20 m	0.21 m
0.50 m	0.49 m
0.70 m	0.70 m
0.90 m	0.92 m

When the unit was rapidly moved through the water the depth measurements increased by maximum 0.14 m.

Yaw Controller

The proportional gain was set to $K_p = 0.30$ after several experiments. Integral gain was gradually increased from zero to produce different control behavior. These results are displayed in Figure 7.8.

Camera Calibration

Camera calibration was based on four coordinate pairs from the 640x480 video frame shown in Figure 7.1:

(X, Y)		(u, v)
$(0.0, 0.0)$	$\longleftrightarrow$	$(35, 322)$
$(0.0, 2.5)$	$\longleftrightarrow$	$(633, 321)$
$(2.5, 2.5)$	$\longleftrightarrow$	$(501, 45)$
$(2.5, 0.0)$	$\longleftrightarrow$	$(127, 44)$

These points resulted in the following mapping between camera- and real world coordinates (Equation (C.2)):

$$\mathbf{C} = \begin{bmatrix} 67.2257 & 239.3318 & 35.0000 \\ -100.6588 & 0.8091 & 322.0000 \\ 0.2396 & 0.0038 & 1.0000 \end{bmatrix} \tag{7.1}$$

where $s = 0.8434$.

(7.1) was used to remap the coordinates recorded in the tracking procedure. A sample mapping from the first search path is shown in Figure 7.9.

Search Pattern Tests

Two square search paths and one linear was tested at Tyholt. A comparison of the true- and virtual search paths are shown in Figure 7.10.

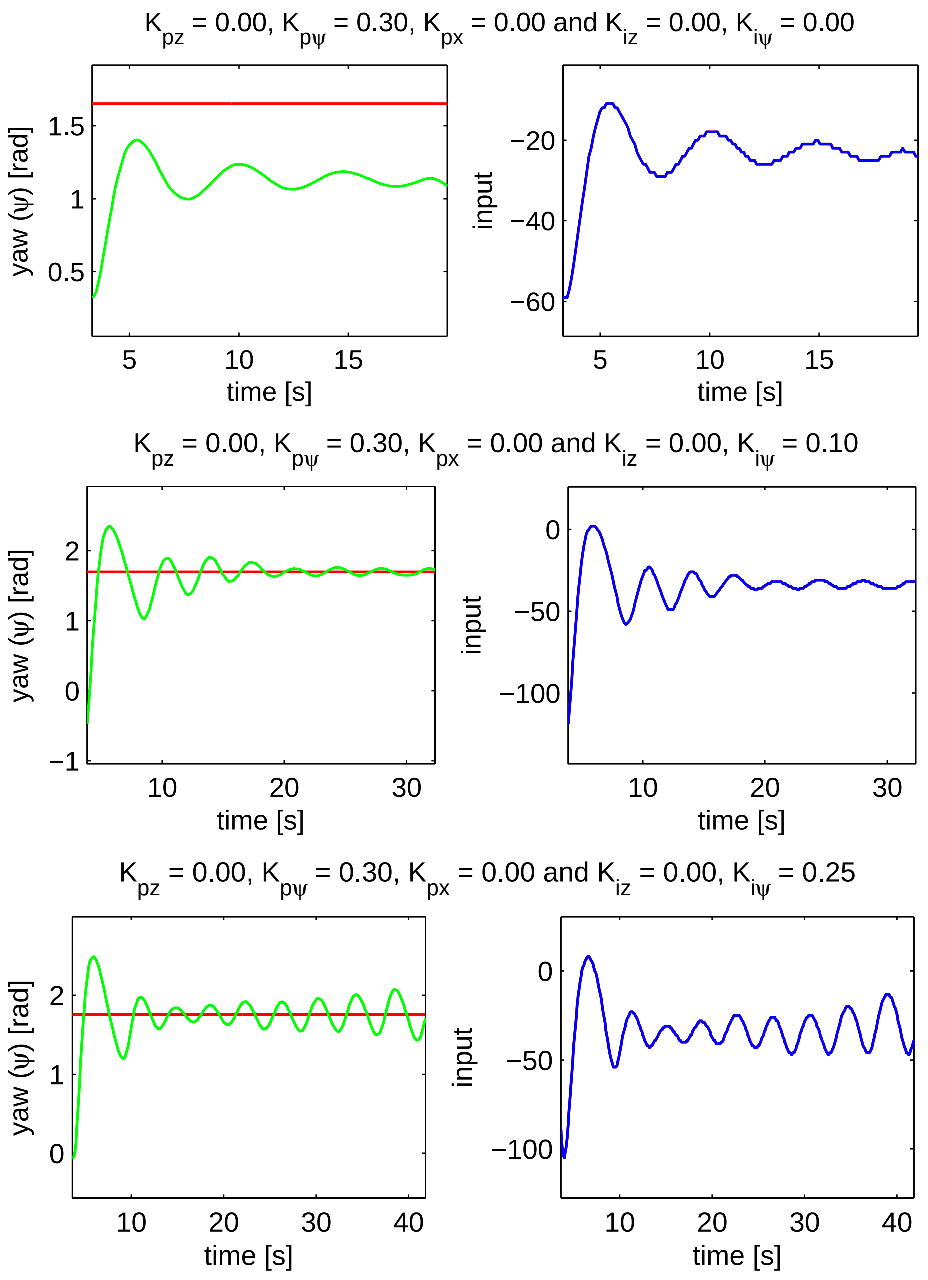


Figure 7.8: Independent yaw control. Left column: Green indicates measured yaw angle. Red indicates yaw setpoint. Right column: Blue line indicates port and starboard control inputs.

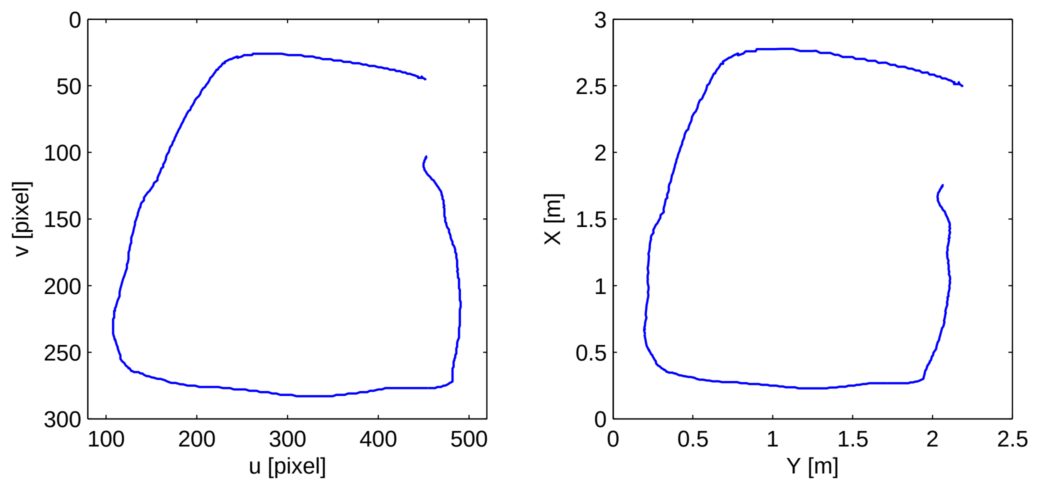


Figure 7.9: Sample remapping of first search path test. Left: pixel coordinates obtained with the mean-shift algorithm. Right: real-world coordinates mapped from pixel coordinates.

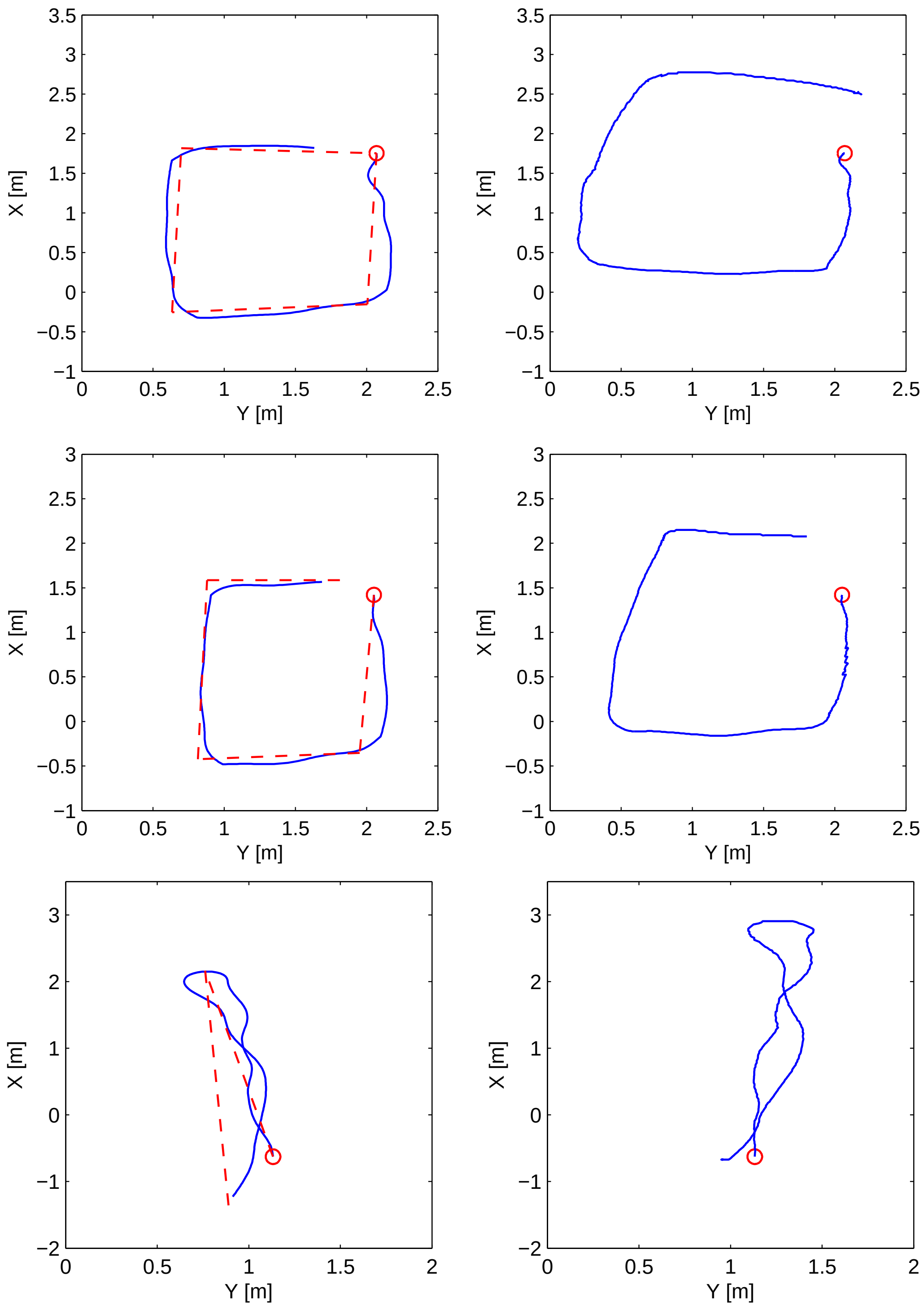


Figure 7.10: A comparison between virtual- and estimated search paths. Left column: Blue indicates the virtual ROV coordinates that result from the path-following objective. The red, stippled line indicates the search path based on user specified waypoints. The red circle indicates the ROV's start point. Right column: Blue indicates an estimate of the ROV's true coordinates based on video tracking.

7.3 Discussion

Hardware and Software

The navigation module was operable in the water tank and did not take in water when exposed to a depth of 1 m. This was expected as the housing was pressure tested at a much greater rating in the workshop.

The navigation software worked under Windows XP, but the video module failed sporadically under Windows 7 and was therefore removed during the tests.

Unfiltered IMU Tests

Acceleration data alone is in this case not sufficient for calculating position. After a period of 10 minutes the integrated velocity and displacement had grown significantly despite of being stationary. Professional inertial navigation systems do not solely rely on MEMS-technology and will have access to more accurate inertial measurements. By adding more sensors with different error characteristics the overall performance of a navigation system could increase. If used in conjunction with a more accurate error model the gyro drift and acceleration bias could be greatly reduced (Barshan & Durrant-Whyte 1993).

Each sensor's output will slightly depend on its internal temperature which changes during the course of operation. The sensor bias will therefore evolve and eventually stabilize at a noise level. The raw sensor measurements are modelled with a white noise bias which will generate a random walk when integrated. In the seconds rows of Figure 7.3 and Figure 7.4 the random walks can be observed.

Two potential error sources during general operation are the gravitational acceleration compensation and the external sampling rate. The accelerometer data was used to calculate the roll- and pitch angles of the G-vector relative to the sensor axes. Small errors in these angles will result in components of the G-vector on all three axes. As a concrete example consider a tilt error of 0.05°. This error will cause a component with magnitude 0.0086 m/s2 to be projected onto the horizontal axes. This residual bias causes an error in the horizontal position which grows quadratically to 7.7 m after 30 seconds (Woodman 2007).

The sample rate during the stationary tests was set to 10 Hz which is significantly smaller that the IMU's 819 Hz maximum internal sample rate. Such a low rate will not allow the integrators to capture all the dynamics of the ROV during maneuvers. When setting the rate to 60 Hz (which is the largest rate that has been tested successfully) no improvements were observed.

Filtered IMU Tests

As shown in Figure 7.5 the filtered roll- and pitch angles have a much more desirable behavior than those obtained from the integrated gyroscopes alone. The drift shown in Figure 7.4 is not present in the filtered values which only vary within a range of $\pm 2.0\mathrm{e}^{-3}$. However, the yaw angle experiences a large drift of 0.15 rad in 10 minutes. This is 10% of the unfiltered drift which is acceptable for short missions.

The non-stationary tests in Figure 7.6 are more interesting because they show filter performance during ROV motion. Angles calculated from the G-vector are sensitive to the Coriolis effect and centripetal forces. Movement along a circular path will for instance generate forces along y_s which in turn will affect the estimated roll angle. Gyroscope measurements will not be affected as heavily, but will on the other hand drift over time.

The linear oscillation results in Figure 7.7 shows how the accelerometer-based roll increases greatly. The integrated gyroscope rate registers this motion also, but most likely because of the slight rotation of the ROV during handling. Filtered results are a combination of the mentioned values, but is weighted more against gyroscope output. This behavior is desirable since the acceleration data's main purpose in attitude determination is drift compensation.

Rotational oscillation values are almost identical across the different measurement types. The estimated roll- and pitch angles are proficient estimates of the ROV attitude.

Compass Test

A calibration is necessary to remove disturbances associated with the compass mounting location. Motors, ROV housing and internal components of the navigation module all generate unwanted magnetic fields. By compensating for these disturbances the compass range can be made valid, but if this is performed indoors several irrelevant sources will be taken into account.

Armature, reinforced concrete and the surface that the ROV is placed on will all contribute to the disturbance sensed by the compass. Indoor calibration will therefore not produce a valid set of data for the navigation module.

Outdoor calibration was not performed, but will produce better calibration data. This is important when using the system in the field where search patterns are related to the magnetic- or geographic north. A valid measurement combined with the z_s gyroscope data will complete the ROV attitude.

Depth Measurements

The atmospheric pressure was measured right above the water surface and stored for depth calculations. Measurements were accurate within the limited test range, but the module should be tested at greater depths to verify the total theoretical operating range of 58.8 m.

The small error that appeared when the ROV was thrusted through the water is of no concern to the current application. A precision of ± 0.14 m is more than sufficient. The depth controller was not tested because of the water tank's limited depth.

Yaw Controller

A step in yaw reference and no integral term results in a stable response, but with a constant error of about $32.4\,^\circ$ and a stationary port/starboard control input at -20. This error was introduced by the cable.

A new run with the same cable configuration, but with an increased integral gain $K_{i\psi} = 0.10$ removes the constant error, but results in a marginally stable behavior.

Increasing the integral gain to $K_{i\psi} = 0.25$ results in gradually increasing oscillations in error and control input.

Results are displayed in Figure 7.8.

Camera Calibration

The coordinate remapping is very useful when recovering the ROV behavior during a path-following maneuver. Not only does it reverse the skew related to visual perspective, but the output is in a more usable format. The left plot in Figure 7.9 has a trapezoid shape while the right plot show a less skewed shape.

Search Pattern Tests

In the first test, the ROV was supposed to track a square path and return to its point of origin. As shown in the first row of Figure 7.10 the controller produces virtual ROV positions that are consistent with the path. True positions are however less consistent. The length which the ROV travels along each edge is either too short or too long when compared to the path. This is a result of wrong scaling between actual and virtual surge displacement for each time step. Cable pull will give rise to problems that are not fixed by correct scaling. Some maneuvers/positions are affected more by the cable than others and this leads to differences in traveled lengths. By observing the left, vertical side it can be observed how the ROV has the correct orientation, but is being pulled to the right. The mentioned problems are related to the lack of proper position measurements.

In the second run, the virtual and true behavior of the ROV was very similar to those of the first. The path was slightly different, but the cable and scaling factors resulted in the same type of path mismatch.

In the third run a different type of path was tested. The path-following controller did not perform as well as in the previous tests, but the coarse characteristics of ROV behavior reflected those of the virtual.

Chapter 8

Conclusion

The simple design of the navigation module fits the desired specifications, but further development is needed to improve sampling rate and maybe extend the system with velocity measurements. The CAN bus successfully delivers data over the 76 m long tether at i high bit rate and enables remote MCU reprogramming and configuration. All in all, the navigation module fulfills its purpose.

The attitude of the ROV can be estimated with sufficient accuracy when combining gyroscope-, accelerometer- and compass data, even though a simple error model has been used. Depth based on pressure measurements have shown to be very accurate within the test range.

The guidance system and yaw controller are simple, but enables the ROV to roughly follow the planned path. Despite differences in the virtual and true behavior of the ROV, the overall performance is acceptable. The shapes of the paths are similar and a user would not notice the deviation during short missions. However, for missions that take more than a few minutes (which is normal) the difference between the true- and virtual positions would increase without bounds. Error increases with time and with the in-water length of the tether, consequently producing more cable pull and more deviation.

To combat the errors associated with cable pull, a position measurement is necessary. IMU integration alone was shown to be highly unreliable in the current setup. Accelerometer data should be combined with other measurements and a more accurate error model. This will enable a filter to estimate displacement with a higher precision, potentially adequate for real world search and rescue operations.

Further Work

Hardware

- Velocity measurements should be added to improve the position estimate. Acoustic Doppler measurements can provide velocity relative to the seabed in all three axes. Flow measurements based on pressure or turbines can also be used, but will not distinguish between currents and ROV velocity.
- The problems associated with cable pull can be removed by converting the ROV to an Autonomous Underwater Vehicle (AUV). This has been done in (Stipanov et al. 2007) by making control signals and power available from the attached module.
- A dedicated solution for mounting the navigation module is desired. The housing can slip during operation and misalign the sensor axes post calibration. It would also be preferable if the weights used to compensate for ROV buoyancy were integrated or made attachable to the mounting mechanism.

State Estimation

- By implementing a more accurate error model the filter will produce more precise bias estimates and consequently more accurate measurements. In addition, if the error model is combined with the ROV model the state propagation will reflect the kinetics of the ROV. Such a model contains non-linear terms and is compatible with an Extended Kalman filter as described in e.g. (Brown & Hwang 1997). If the model is implemented with unit quaternions, accuracy will be increased, singularities avoided and computational time reduced (if compared to an Euler angle implementation). Such a solution is developed for a low cost IMU in (Kong 2000).
- The IMU's noise properties should be properly investigated to improve Kalman filter performance.
- A complementary filter can be used instead of a Kalman filter. The filter is simpler, but offers some advantages as described in (Brown & Hwang 1997).

- A greater sample rate is desirable because this will capture more of the ROV's dynamics and result in more accurate values from the integrators. At this point the sampling rate limitation is not known, but by moving the estimation process over to the MCU, data transmission is reduced and less processing is necessary in software. The 16 MIPS MCU is capable of handling such a workload, especially if a steady-state Kalman filter is applied.

Controller

- More advanced control schemes can be simulated with the ROV model presented in Chapter 3.
- The guidance system can be improved by using one of the other rules for path generation. Path-following would also be improved by the use of LOS as described in (Borhaug & Pettersen 2005).
- Instead of manually allocating the different thruster inputs, a thrust allocation matrix can be developed as described in (Fossen 2002).

Bibliography

Alcocer, A., Oliveira, P. & Pascoal, A. (2007), 'Study and implementation of an ekf gib-based underwater positioning system', *Control Engineering Practice* **15**, 689–701.

Atmel (2008), *AT90CAN32 Datasheet.*

Ballard, D. H. & Brown, C. M. (1982), *Computer Vision*, Prentice Hall.

Barshan, B. & Durrant-Whyte, H. F. (1993), 'An inertial navigation system for a mobile robot', *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems* **3**, 2243–2248.

Borhaug, E. & Pettersen, K. Y. (2005), 'Adaptive way-point tracking control for underactuated autonomous vehicles', *Proceedings of the 44th IEEE Conference on Decision and Control* .

Bosch (1991), *CAN Specification Version 2.0*, Robert Bosch GmbH.

Bradski, G. & Kaehler, A. (2008), *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media.

Brown, R. G. & Hwang, P. Y. C. (1997), *Introduction to Random Signals and Applied Kalman Filtering*, 3 edn, Wiley.

Catsoulis, J. (2005), *Designing Embedded Hardware*, 2 edn, O'Reilly Media.

Chen, C.-T. (1999), *Linear System Theory and Design*, 3 edn, Oxford University Press.

Farrell, J. A. & Barth, M. (1998), *The Global Positioning System and Inertial Navigation*, McGraw-Hill.

Fossen, T. I. (2002), *Guidance and Control of Marine Craft*, Marine Cybernetics.

Jalving, B., Gade, K., Hagen, O. K. & Vestgard, K. (2003), 'A toolbox of aiding techniques for the hugin auv integrated inertial navigation system', *OCEANS 2003. Proceedings* **2**, 1146–1153.

Joblove, G. H. & Greenberg, D. (1978), 'Color spaces for computer graphics', *Computer Graphics (SIGGRAPH '78 Proceedings)* **12**(3), 20–25.

Jones, D. L. (2004), *PCB Design Tutorial*, `http://alternatezone.com/electronics/pcbdesign.htm`.

Karras, G. C. & Kyriakopoulos, K. J. (2007), 'Localization of an underwater vehicle using an imu and a laser-based vision system', *Proceedings of Mediterranean Conference on Control and Automation* .

Kong, X. (2000), Inertial Navigation System Algorithms for Low Cost IMU, PhD thesis, University of Sydney.

Lee, P., Jeon, B., Kim, S., Choi, H., Lee, C., Aoki, T. & Hyakudome, T. (2004), 'An integrated navigation system for autonomous underwater vehicles with two range sonars, inertial sensors and doppler velocity log', **3**, 1586–1593.

Leondes, C. T. (1963), *Guidance and control of aerospace vehicles*, McGraw-Hill.

Maybeck, P. S. (1979), *Stochastic models, estimation, and control*, Vol. 141 of *Mathematics in Science and Engineering*, Academic Press.

Miskovic, N., Vukic, Z. & Barisic, M. (2007), 'Identification of coupled mathematical models for underwater vehicles', *Proceedings of OCEANS07* .

Nilsson, J. W. & Riedel, S. A. (2005), *Electric Circuits*, 7 edn, Prentice Hall.

Skjaeveland, J. (2009), Utvikling av akustisk målemetode for oppdrettsmerders form, Master's thesis, Norwegian University of Science and Technology.

Stipanov, M., Miskovic, N., Vukic, Z. & Barisic, M. (2007), 'Rov autonomization - yaw identification and automarine module architecture', *Proceedings of the 2007 IFAC World Conference* .

VideoRay (2010), 'Technical data', `http://www.videoray.com`.

Vik, B. (2009), *Integrated Satellite and Inertial Navigation Systems*, Department of Engineering Cybernetics, NTNU.

Whitcomb, L. L., Yoerger, D. R. & Singh, H. (1999), 'Combined doppler/lbl based navigation of underwater vehicles', *Proceedings of the 11th International Symposium on UUST* .

White, F. M. (1998), *Fluid Mechanics*, 4 edn, McGraw-Hill.

Woodman, O. J. (2007), *An Introduction to Inertial Navigation*, University of Cambridge, Computer Laboratory.

Zhang, Z. (1999), 'Flexible camera calibration by viewing a plane from unknown orientations', *Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999* **1**, 666–673.

Appendix A

VideoRay Communication Protocol

The Communication Protocol for VideoRay Pro III and Desktop Computer Revised 11/04/06 by Marcus Kolb

Physical media: RS232, baud rate 9600, 8 bit, 1 stop, no parity.

- **Enabling computer control**

  When the VideoRay is powered on it waits 5 seconds for a byte to be received on the RS232 port. If it receives anything, it enters into computer control mode. Otherwise the vehicle will be directly controlled by the control panel.

- **Normal Communications Between VideoRay and PC**

  VideoRay waits for 8 bytes containing information for running the vehicle. Then, the VideoRay sends out 7 bytes containing a 3 byte identifier, compass and pressure data.

  The PC sends 8 control bytes and waits for 7 data bytes coming from VideoRay and if it receives them, it sends out the next 8 control bytes immediately. VideoRay will keep waiting for the entire 8 bytes until it receives all of them (VideoRay Pro works in polling mode).

  The total time used for exchanging information between VideoRay Pro and PC is about 15.6ms ((8+7)/(9600/(1+8+1)). This will not affect the control characteristics of the vehicle provided the PC does not make the VideoRay Pro keep waiting for too long.

- **Information of the 7 Bytes VideoRay Sends** The first 3 bytes of the 7 bytes contain an identifier then compass low byte, compass high byte, pressure low byte and pressure high byte.

  1. 0x40 (hex) (All VideoRay models)
  2. 0x31 (hex) (All VideoRay Pro III)
  3. 0x02 (hex) (data type for future use)
  4. Low byte of Orientation
  5. High byte of Orientation
  6. Low byte of Depth
  7. High byte of Depth

  The relation between low byte, high byte and the real value is:

  Real value = Low byte + 256 x High Byte, for instance:

  Orientation* = Low byte of Orientation + 256 x High Byte of Orientation (0-359)

  Depth = Low byte of Depth + 256 x High Byte of Depth (0-1023)

  *When Orientation is calculated, the following conversion is needed for the real orientation:

  Real Orientation = 360 - Orientation; // mirror the image of the orientation

  if (Real Orientation < 90) Real Orientation = 270 + Real Orientation; // shift 90 degrees counterclockwise

  else Real Orientation = Real Orientation - 90;

- **Information in the 8 Bytes PC Sends**

  1. 0x23 (hex) (for all VideoRay models)
  2. 0x31 (hex) (for Pro III)
  3. Current for the port thruster, minimum 0, maximum 220
  4. Current for the starboard thruster, minimum 0, maximum 220
  5. Current for the vertical thruster, minimum 0, maximum 220
  6. Current for the lights, minimum 0, maximum 200

7. Bit level Controls for the manipulators and auto depth:

   D7 D6 D5 D4 D3 D2 D1 D0

   - D0 = 0, Manipulator 1 close; D0=1, Manipulator 1 open
   - D1 = 0, Manipulator 1 disable; D1=1, Manipulator 1 enable
   - D2 Reserved
   - D3 Reserved
   - D4 Reserved
   - D5 Reserved
   - D6 = 0, Front light / camera; D6=1, Rear light / camera
   - D7 Reserved

8. Bit level controls of camera tilt and focus, the direction of the thrusters:

   D7 D6 D5 D4 D3 D2 D1 D0

   - D0 = 0, Tilt up; D0 = 1, Tilt down
   - D1 = 0, Tilt disable; D1 = 1, Tilt enable
   - D2 = 0, Focus near; D2 = 1, Focus far
   - D3 = 0, Focus disable; D3 = 1, Focus enable
   - D4 = 0, Port backward; D4 = 1, Port forward
   - D5 = 0, STBD backward; D5 = 1, STBD forward
   - D6 = 0, Vertical up; D6 = 1, Vertical down
   - D7 is reserved

Appendix B

Navigation Module Specifications

By assessing the different component ratings the navigation module will have the following specifications:

- Operating voltage: +18 V to +75 V DC
- Operating temperature: −20°C to +80°C
- Maximum depth with valid output: 58.5 m (100 psi)
- CAN transmission speed: 250 kbit/s
- Maximum observed sample rate: 60 Hz

The navigation module's connector pinout displayed in Figure B.1. Pins 3/5 are connected to the power source, pins 4/6 are the CAN bus lines and pins 1,2,7,8,9 are not connected.

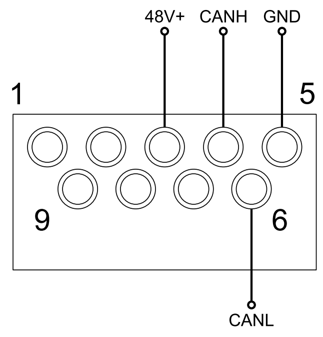


Figure B.1: Navigation module connector pinout.

The transmitted and accepted CAN messages are shown in Table B.1.

Direction	**Message Type**	**ID**	**Data Fields**
TX	Data package one	0x01	Byte 1-6: Accelerometer data (Figure 5.13)
TX	Data package two	0x02	Byte 1-6: Gyroscope data (Figure 5.13)
TX	Data package three	0x03	Byte 1-6: Compass-, temperature- and pressure data (Figure 5.13)
TX	IMU error	0x12	Byte 1-2: High- and low byte of IMU status respectively.
RX	Reset MCU	0x000	N/A.
RX	Configure compass	0x100	Byte 1: Compass specific command.
			Byte 2: Compass specific command has argument (0x00 = false, 0x01 = true).
			Byte 3: Compass specific command argument high byte.
			Byte 4: Compass specific command argument low byte.
RX	Configure IMU	0x200	Byte 1: Command
			0x00 = Sample rate.
			0x01 = Range and filter taps.
			0x02 = Run calibration.
			Byte 2: Argument 1
			0x01 = Set sample rate to 100 Hz.
			0x02 = Set sample rate to 200 Hz.
			0x03 = Set sample rate to maximum.
			0x04 = Set range to 75°/s.
			0x05 = Set range to 150°/s.
			0x06 = Set range to 300°/s.
			Byte 3: Argument 2 (used in conjunction with command 0x01)
			0x07 = Set filter tap number to 2.
			0x08 = Set filter tap number to 4.
			0x09 = Set filter tap number to 8.
			0x0A = Set filter tap number to 16.
			0x0B = Set filter tap number to 32.
			0x0C = Set filter tap number to 64.
			0x0D = Set filter tap number to 128.
RX	Configure sample rate	0x300	Byte 1: Sample rate in Hz (unsigned char).

Table B.1: CAN messages that are transmitted and accepted by the navigation module.

Appendix C

Camera Calibration and Coordinate Recovery

A camera calibration routine is used to map the relationship between the image pixels and real world coordinates. This mapping depends on several properties that are referred to as intrinsic- and extrinsic camera parameters. Intrinsic parameters are related between the camera coordinate system and the image sensor, e.g. focal length and scale factors. Extrinsic parameters involve the rotation and transformation between camera- and real world coordinates.

The mapping between coordinate systems can be represented by the following Equation (Zhang 1999):

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{C.1}$$

s	Arbitrary scale factor related to homogenous coordinates
u, v	Camera coordinates
$\mathbf{A}$	3-by-3 intrinsic camera transformation
$\mathbf{r}_1, \mathbf{r}_2$	Rotation column vectors of extrinsic camera transformation
$\mathbf{t}$	Translation column vector of extrinsic camera transformation
X, Y	Real world coordinates

Equation (C.1) is a simplification of the general case where real world coordinates have an additional Z-value. Because the ROV motion is performed in a plane Z is set to zero.

By multiplying the intrinsic and extrinsic matrices the mapping can be expressed as:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathbf{C} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{C.2}$$

u and v can be expressed as:

$$u = \frac{su}{s} = \frac{C_{11}X + C_{12}Y + C_{13}}{C_{31}X + C_{32}Y + C_{33}} \tag{C.3}$$

$$v = \frac{sv}{s} = \frac{C_{21}X + C_{22}Y + C_{23}}{C_{31}X + C_{32}Y + C_{33}} \tag{C.4}$$

The overall scaling of $\mathbf{C}$ is irrelevant, thanks to the homogenous formulation, so C_{33} may be arbitrarily set to 1 (Ballard & Brown 1982). By rearranging Equation (C.4) the following relationship is obtained between each set of camera- and world coordinates:

$$u = C_{11}X + C_{12}Y + C_{13} - uC_{31}X - uC_{32}Y \tag{C.5}$$

$$v = C_{21}X + C_{22}Y + C_{23} - vC_{31}X - vC_{32}Y \tag{C.6}$$

Eight of the nine elements in $\mathbf{C}$ are unknown and minimum eight equations are needed to determine their values. By choosing four camera coordinates and measuring their real world positions, the eight necessary constraints are achieved:

$$\mathbf{A}\mathbf{b} = \mathbf{c} \tag{C.7}$$

$$\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -v_1X_1 & -v_1Y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -u_4X_4 & -u_4Y_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -v_4X_4 & -v_4Y_4 \end{bmatrix} \begin{bmatrix} C_{11} \\ \vdots \\ C_{32} \end{bmatrix} = \begin{bmatrix} u_1 \\ v_2 \\ \vdots \\ u_4 \\ v_4 \end{bmatrix} \tag{C.8}$$

Equation (C.8) is a non-homogenous set of equations that can be solved by inverting the 8-by-8 $\mathbf{A}$ matrix: $\mathbf{b} = \mathbf{A}^{-1}\mathbf{c}$. The matrix will be non-singular as long each marker is selected only once.

When the $\mathbf{C}$ matrix has been computed the mapping between camera- and world coordinates can be found by rearranging Equation (C.2):

$$\frac{1}{s}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathbf{C}^{-1}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{C.9}$$

World coordinates are found by first recovering the scaling factor s from the matrix product and then scaling X and Y with s.