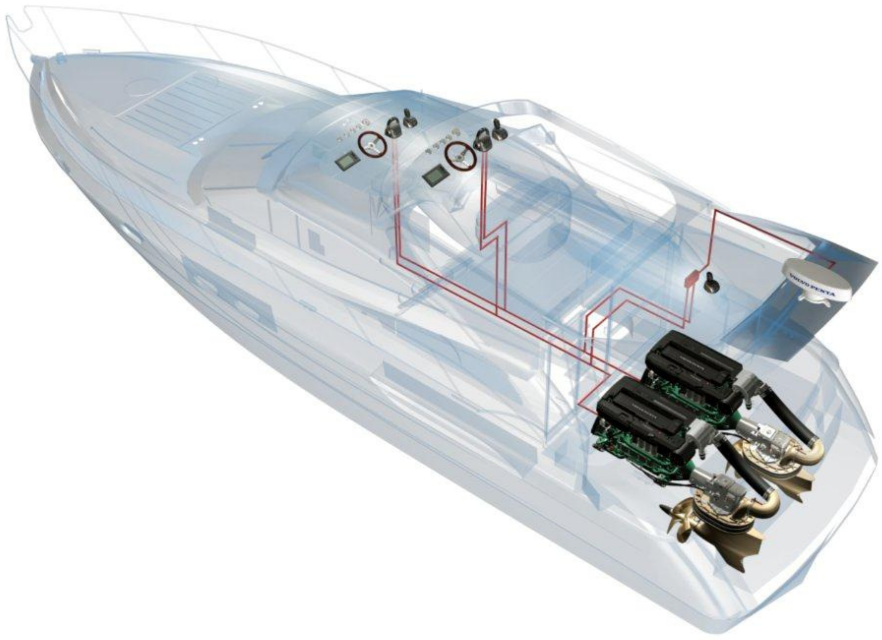
CHALMERS



A Multi-Purpose Gateway for Vessel Steer-By-Wire Systems

Embedded protocol converter, signal processor and controller

Master of Science Thesis in the Master's Programme Systems, Control and Mechatronics

MARTIN BYRHULT **VICTOR STENSSON**

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2011

A Multi-Purpose Gateway for Vessel Steer-By-Wire Systems

Embedded protocol converter, signal processor and controller

Master of Science Thesis in the Master's Programme Systems, Control and Mechatronics

MARTIN BYRHULT **VICTOR STENSSON**

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2011

Cover: The transparent boat shows a part of the steer-by-wire system and the electrical communication bus that are common among leisure boats. Source: Volvo Penta's Image Gallery.

Abstract

The marine leisure boat industry comprises many different systems, with different protocols, that need to exchange information. This is enabled by interface nodes called gateway that translates signals in between. From a number of protocol specifications, including NMEA 2000 and NMEA 0183, a list of requirements are compiled and a software platform is developed based upon them. The software design is based on layers similar to the well-known Open System Interconnection (OSI) model which communicate with each other over well-defined interfaces. Typically the lower level protocol layer receives and transmits information onto the network and parses signals to and from the signal layer which handles signal validity and signal scaling. The gateway layer maps signals between the protocols. A clear structure of the software platform makes it easy to define new protocols and change signal handling. The target processor is a 32-bit, ARM Cortex-M3 based microcontroller in ST Microelectronics' STM32 family with two CAN controller modules.

A gateway is also a proper placeholder for control and filtering functions since many signals passes through this device. The estimation of a boat's roll and pitch angles, used for trim control, is an example which is analyzed thoroughly. An accelerometer and a gyroscope are used for measuring absolute angle and angular rate respectively. By modeling the bias of the gyroscope with a constant and a proportional part in a Kalman filter together with the ability to turn off the accelerometer input during great acceleration, the overall system give an accurate estimation despite that the sensors are affected by horizontal accelerations from the boat.

Preface

This 30 hp master thesis is carried out at CPAC Systems AB located in Gothenburg, Sweden, the spring semester of 2011. It is a course in the master's programme Systems, Control and Mechatronics, given by the Department of Signals and Systems at Chalmers University of Technology.

CPAC systems AB, a company in the Volvo group, designs and manufactures steer-by-wire control systems for the marine and industrial vehicle industry. In particular, CPAC manufactures the control system that governs the Volvo Penta's well-known IPS system (Inboard Performance System), an innovative propulsion system bringing increased efficiency and maneuvering capabilities.

The authors would like to thank our supervisors at CPAC Systems AB, senior HW developer Michael Pettersson, for his technical advises and overall support, and R&D manager Marco Monzani, for his help with planning the project and great help with technical communication and writing. Many thanks also go to our supervisor at the Department of Signals and Systems, Associate Professor Jonas Fredriksson.

Keywords: gateway, CAN, NMEA 2000, trim system, sensor fusion, pitch control, kalman filter.

Table of Contents

Abstract i

Preface ii

Abbreviations v

1 Introduction 1

1.1 Background 1

1.2 Problem formulation 2

1.2.1 Limitations 3

1.3 Purpose 3

1.4 Prerequisites 3

1.5 Outline of thesis 4

2 Theory 5

2.1 Gyroscope 5

2.2 Accelerometer 5

2.3 Protocol 6

2.4 Controller Area Network 7

2.4.1 Arbitration field 7

2.4.2 Data field 7

2.4.3 CRC field 7

2.5 Multilink/INOC 7

2.5.1 Data frames 8

2.5.2 Address claim 8

2.5.3 Error detection 8

2.6 NMEA 2000 8

2.7 NMEA 0183 8

3 Generic gateway design 9

3.1 Requirements of the gateway 9

3.1.1 Instance handling 9

3.1.2 Time-out behavior 10

3.2 Software design 10

3.2.1 Software reuse 12

3.2.2 Describing a signal 12

3.2.3 Signal scaling 12

3.2.4 Signal validity 13

3.2.5 Optionals 13

3.2.6 Transmitter and receiver 13

3.3 Discussion 14

4 Trim controller case study 15

4.1 AutoTrim System 15
4.2 Problems with the existing ACU 15
4.3 Estimating roll and pitch: test results 16
4.3.1 Accelerometer 17
4.3.2 Accelerometer and gyroscope 19
4.3.3 Accelerometer, gyroscope and horizontal acceleration sensor 22
4.4 A new ACU 23
5 Conclusion 25
5.1 A multi-purpose gateway 25
5.2 Estimating roll and pitch: from sensors to Kalman filtering 25
5.3 Combining a gateway with filtering functionality 26
5.4 Future work 26
References 27
A Test rig
B MATLAB script

Abbreviations

8N1	8 data bits, No parity bit, 1 stop bit
ACU	AutoTrim Control Unit
ADC	Analog-to-Digital Converter
ATS	AutoTrim System
CAN	Controller Area Network
CRC	Cyclic Redundancy Check
FIR	Finite Impulse Response
GPS	Global Positioning System
HCU	Helm Control Unit
MEMS	Micro-Electro-Mechanical Systems
OSI	Open Systems Interconnection
PGN	Parameter Group Number
SPI	Serial Peripheral Interface

1 Introduction

The number and complexities of functions available on leisure boats has increased through the last decades. From being essentially mechanical, boats today include for example steer-by-wire systems, navigation equipment, autopilots and dynamic positioning systems (digital anchor). These embedded systems are available thanks to the great evolution of the computer. Like other industries, the development constantly tends to integrate more functions in smaller packages. This development gives road to new problems to arise but also great opportunities to realize new ideas and implement more advanced functionalities.

1.1 Background

A growing market and many manufacturers give rise to numerous products. Some of the products such as GPS (Global Positioning System) receivers, compasses and autopilots have a standardized communication protocol while other, more manufacture specific equipment, uses their own proprietary protocols. Customers have demands of being able to connect the external equipment to the boat main control system. This puts requirements on the manufacturers to have a system able to communicate with the standardized protocols, i.e. the different protocols should be able to exchange information. A typical setup is illustrated in Figure 1 where different protocols share information. The functionality that enables the protocol translation is an interface node called gateway.

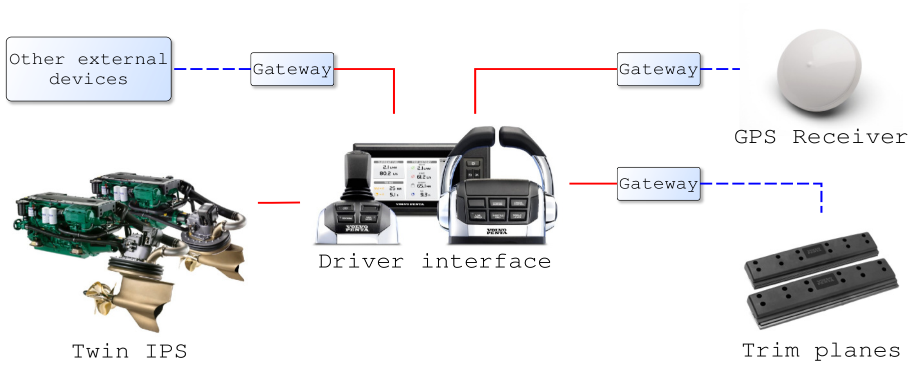


Figure 1 Typical applications of the gateway. Whole (red) lines represent CPAC's proprietary protocol while the dashed (blue) lines represent other protocols. Source for pictures in figure: Volvo Penta's Image Gallery.

As an example, an external control system should be installed on a boat and needs to be fed with information from the engine. Thus the system has to be connected to each other. It is however not possible since the external system and the engine uses different protocols. The solution is to add the functionality of a gateway to the external control system. The reverse could also apply: it is attractive to add control or filtering functions to a gateway since signals already are processed as they pass through the gateway. Such an external control system is for example one that controls the *trim* of a boat, i.e. the objective to minimize the angle deviation from an upright position.

The need to control the trim originates from that boats easily lean in some direction, due to wind, rough seas or rudder angles. It can also be affected by how the boat is loaded. Leaning resulting from wind and load, often results in stationary errors. These could be corrected manually in an open-loop manner by controlling actuators creating upward forces on each side of the stern, so called trim planes. However, if the weight distribution or the angle of the wind is changed, the trim plane setting must be changed in real-time to preserve the upright comfortable position of the boat, hence a closed-loop system is needed. The trim planes as well as the controlled angles roll and pitch, are shown in Figure 2.

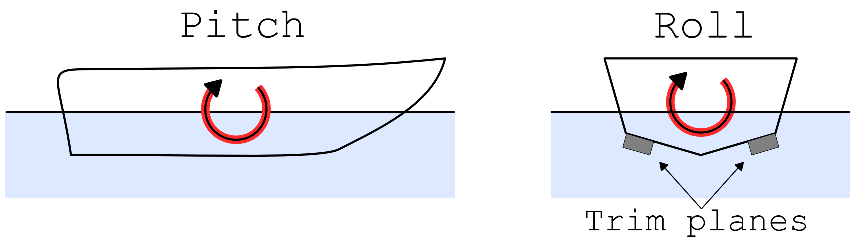


Figure 2 Defines the angles (roll and pitch) of a boat which are important for trim control. To the right the trim planes are outlined in the stern of a boat. The trim planes angle of contact can be adjusted individually.

1.2 Problem formulation

In order for different protocols to be able to exchange information, gateways are used as depicted in Figure 1. Besides that the protocols does not match, gateways are also needed to separate (electrically isolate) the protocols for safety reasons. The existing gateways are based on a rather outdated processor, which set severe limitations. In addition, there are numerous variants of the gateway in terms of both software and hardware, one for each pair of protocol.

Thus, the problem includes investigating the requirements of the existing gateways and developing the software platform for a *generic* gateway, see Figure 3, based on those investigations. The generic gateway will replace the existing variants and thus has to support multiple protocols. A software design description should be written to document the functions and structure of the gateway software.

As mentioned in the background, the gateway can be combined with control or filtering functions since many signals are processed in the gateway anyway. Filtering functions can be used to adapt the dynamics of an external signal to the receiving protocol or sensor fusion functions can be used to improve estimates of process variables. The capability of integrating filters in a gateway is investigated with the trim controller as a use case. Furthermore the trim controller system itself is investigated in order to discuss and propose improvements for the existing system's shortcomings. More specific, the problem of determining the process output (measuring the roll and pitch angles) is investigated.

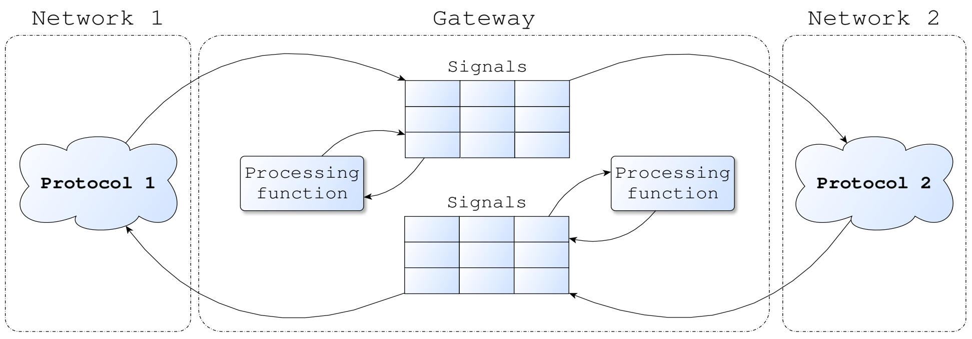


Figure 3 A gateways' task is to transfer signals between a set of two different protocols. Often some processing is needed. A generic gateway has to support various combinations of protocols.

1.2.1 Limitations

The development of a generic gateway in this thesis excludes hardware. Not all protocols in existing gateway applications are implemented, the goal is to create a generic software platform that makes such an implementation easy.

Regarding the trim controller, the work does not include investigating the controller or its design. The works aims on investigating the estimation of the process output.

1.3 Purpose

The purpose of developing a generic gateway is to make an up to date interface node that can be configured and adapted to numerous protocols without having to carry out a new product developing and testing process for every new pair of protocol. A generic gateway will simplify the maintenance of the gateway firmware and the new target processor can also cope with heavier calculations thus allowing control functions and filters to be implemented.

The purpose of investigating the trim controller system, particularly the roll and pitch estimation, is to improve the overall performance by proposing possible solutions to problems identified with the existing system.

1.4 Prerequisites

As target processor for the generic gateway a microcontroller in ST Microelectronics' STM32 family is used, since it is widely used in CPAC's products. Thus the hardware used during the development and verification is a development board including the STM32 microcontroller and two CAN channels. Two CAN channels are required since a common application for the gateway involves NMEA 2000 and Multilink/INOC which both are CAN based. The board is called IAR-STM32F107VC-SK[1].

[1] More information can be found online at http://www.iar.com/ [2011].

Multilink/INOC (one of CPAC's proprietary protocols) is often used in CPAC's devices and thus a software platform is available for the STM32 microcontroller which implements CAN and Multilink/INOC functionality. It is written for IAR's[2] C/C++ compiler and will serve as a basis for the generic gateway. For this reason the software development in this project employ IAR's integrated development environment and its supplied standard peripheral library.

1.5 Outline of thesis

The required theory about sensors and relevant protocols are described in chapter 2. In chapter 3 the requirements of the gateway are first listed and then the related solution is outlined. In the end of the chapter the gateway solution is summarized in a discussion. The trim controller, as a possible application to be combined with the gateway, is analyzed in chapter 4. Problems with the existing solution are identified and the estimation of roll and pitch are studied through test results from a test rig. In the end of chapter 4 the new solution to the trim controller is presented and discussed. Finally conclusions are stated in chapter 5.

[2] IAR Systems is a provider of software tools for embedded systems.

2 Theory

The theory presented in this section is required to understand the other parts of the report. If the reader already is familiar with the subjects, this section can be skipped.

2.1 Gyroscope

A gyroscope is an element capable of sensing angular rate ([degrees/second]) through the measurement of different physical phenomena. There are basically three types of gyroscopes [1]:

- The spinning mass gyroscope involves a mass spinning at high speed and can detect angular momentum caused by Coriolis force.
- The optical gyroscope such as the ring laser gyroscope uses the Sagnac effect which implies that the time of travel for a light beam in an object undergoing rotation is a function of the angular rate. It is therefore the relative phase shift between two beams traveling in opposite directions which results in changes in the interference pattern, a phenomena which is detected and converted into angular rate.
- Vibrating (resonator) gyroscopes measures the deflection in the vibrating element affected by Coriolis force. The vibrating mass is excited electrostatically and the deflections are sensed capacitively. The widespread MEMS (Micro-Electro-Mechanical Systems) sensors are examples of this type of gyroscope.

The first and second types are more accurate and considerably more expensive compared to the vibrating gyroscope. The vibrating gyroscope technique is attractive because of cost and size. A typical MEMS gyroscope fits in a surface mounted 10x10 mm package. The drawback is however that it is quite much affected by temperature changes and that the zero angular rate output level also changes with time. Since the gyroscope output often is integrated to get the angle, the issue with changing zero rate output level becomes a noticeable problem since the error will tend to grow over time.

2.2 Accelerometer

An accelerometer senses acceleration ([g] or [m/s^2]) and exists in many designs but it is only the MEMS type which is applicable to inclination measuring in a boat. MEMS accelerometers can be either piezoresistive, capacitive or resonant [2].

The MEMS accelerometer is a cheap and attractive alternative for the measurement of inclination. Considering that all acceleration is zero with respect to an inertial frame and the only force present is the gravitational force, the angle relative the horizontal can be calculated as $\alpha = \sin^{-1}\left(\frac{a}{g}\right)$ considering the setup in Figure 4.

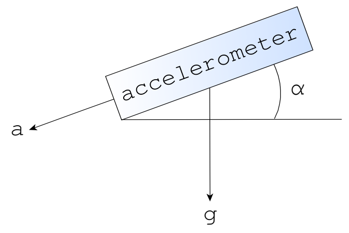


Figure 4 Illustrates a tilted (α) accelerometer sensing an acceleration a when affected by gravity g.

However if a horizontal acceleration (perpendicular to g) also affects the sensor, the given equation would give a false estimation of the angle. Typically this is a problem if the accelerometer is to be used in an accelerating vehicle.

2.3 Protocol

In order to transfer information from sender to receiver, a common physical transmission protocol must be used. The protocol defines the rules for the transmission, [3]. Since the complexity of a protocol varies from simple protocols as RS-232, defining only electrical specifications, to protocols like IP (Internet Protocol), defining rules for sending information over vast ranges, a model has been defined by the International Standards Organization.

The OSI (Open Systems Interconnection) model, Figure 5, divides a protocol in different layer protocols. The physical layer, being the only layer having a physical interconnection between sender and receiver, is responsible for sending raw bits and define for example bit rate. The next layer is called the data-link layer and makes bits into for example data bytes, check bytes and address bytes. The next five layers, network, transport, session, presentation and application layer continues to higher the abstraction level of information interpretation.

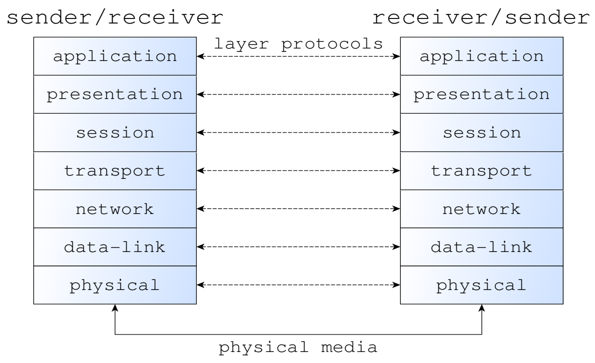


Figure 5 The OSI model divides a protocol in seven layers, numbered 1-7 starting from the bottom. The physical layer is the only layer having a physical connection, the other connections are virtual ones.

The virtual protocols (layer 2-7) communicates with each other, from sender to receiver, by sending the information through the lower layers, over the physical media and, at the receiver, from the physical protocol and further up again.

2.4 Controller Area Network

CAN is one of the most used protocols within vehicle industry in general, and also in the more specific leisure boat segment. Both Multilink/INOC and NMEA 2000 uses CAN as the physical and transport layer (see section 2.3 for the layers in the OSI-model) to transfer their own specific data frames. CAN is electrically built around an open collector (wired OR) bus with multiple nodes and is a multimaster protocol. That gives the advantages that any of the nodes can start to transmit and all the other nodes can listen to the information. The information in this section originates from Bosch CAN specification, [4].

The core element in the CAN protocol is defined by the data frame as illustrated in Figure 6. It consists of seven fields.

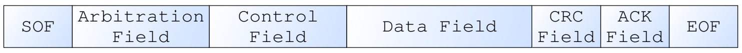


Figure 6 A CAN data frame consist of seven fields. The data field holds the useful information while the other fields only are used during communication.

2.4.1 Arbitration field

Part of the arbitration field is the identifier which can have two formats: standard (11-bit) or extended (29-bit). A typical identifier can hold information about both the information content of the data frame and for example the source or destination address, i.e. which node that sent the message and which node that should receive it.

A great advantage that is included in the CAN protocol is the arbitration mechanism, i.e. if more than one node starts sending at the same time, the conflict of which node that should be allowed to send their massage is avoided. This is solved through arbitration where the sending nodes can detect the conflict and resolve it without having to resend any information. A node sends information, bit by bit, and continuously reads the bus and as soon it detects a *dominant* bit level, though sending a *recessive* bit, the node has lost and must withdraw.

Arbitration is carried out on the identifier and thus they are often carefully selected so that a data frame with short deadline has a low identifier, thus important frames are guaranteed to be sent before less important frames.

2.4.2 Data field

The effective load of the data frame is contained in the data field which is between 8 and 64 bits wide. The structure and interpretation of this field are determined by the higher layer protocols such as Multilink/INOC or NMEA 2000.

2.4.3 CRC field

The CAN protocol has a built in CRC (Cyclic Redundancy Check) that is able to detect if there are any errors in the transfer. Other higher layer protocols may define additional error detection mechanisms.

2.5 Multilink/INOC

Multilink/INOC is a bus protocol CPAC has developed and used in the Volvo Penta Electric Vessel Control system. It is used as a synchronization bus between the HCU:s (Helm Control Unit) but also as an interface to various displays and sensors. It is built upon the J1939 CAN based interface standard, but with own data frames defined. This information in this section comes from CPAC’s Multilink CAN specification, [5].

2.5.1 Data frames

The Multilink/INOC standard specifies a variety of *parameter groups* which basically are CAN frames which hold a number of related *signals*. The 64-bit data contained in each frame usually represent five to ten signals which for example could be pressures, voltages, speed, position or temperatures.

Every parameter group has a unique *PGN* (Parameter Group Number) which together with the priority bits and source address build up the parameter groups CAN identifier. Furthermore each parameter group has a specified update rate. This implies that a parameter group is sent periodically. If a node needs the signals in a certain parameter group at some other time it can send a *request* with the PGN as an argument. The transmitting node should then answer with the requested parameter group.

2.5.2 Address claim

Besides request, another parameter group worth notice is the address claim. It is sent upon startup of a node to make sure that it has a unique source address. If the addressed claimed is already taken the holder of that source address evaluates if it has a higher or lower priority than the newly started node. If the answer is higher, an address claim is sent telling the newly started node that it has to try with a new address to claim. As a result every node continuously monitors the bus for address claims.

2.5.3 Error detection

Multilink/INOC has besides the built in fault detection in CAN (CRC field), two own error detections. Sequence counter is one of them, it is an 8-bit value that increments every transmission. In this way a receiver of the data frame can detect if it has missed any update of the parameter group. The second error detection is a CRC, which in contrast to the one defined by the CAN specification is contained in the data field. Both the sequence counter and the CRC is only part of more safety critical parameter groups.

2.6 NMEA 2000

NMEA 2000 is a standard describing a CAN based instrument network used to interconnect marine electronic devices. The standard developed by National Marine Electronics Association is designed to be able to insure that mission critical data is transmitted timely and insure that possible faulty data is detected.

NMEA 2000 utilizes data frames and address claim in the same way as Multilink/INOC, described in section 2.5.1 and 2.5.2. Regarding the data frames, NMEA 2000 has its own set of parameter groups.

2.7 NMEA 0183

NMEA 0183 interface standard is the predecessor to NMEA 2000 but is, however, still used in marine applications today, especially for professional applications on ships. It is a one-way serial data bus which carries information in *sentences* consisting of ASCII characters. A bus can have one *talker* and one to several *listeners*. The baud rate is typically 4800 bit/s (but it can be higher) and data is sent in 8N1 format.

A sentence can be at maximum 80 characters long and starts with a '$'. After the start, a five character identifier follows together with the data which is separated by commas. The sentence ends with an optional check sum followed by carriage return and line feed ('CR', 'LF'). In the most typical case, each sentence is sent once per second. Standard sentences are defined which for example transfer information about vessel's position, speed, water depth or engine RPM. Manufacturer-specific sentences may be defined.

3 Generic gateway design

As mentioned in the introduction the generic gateway should be able to gateway signals across a number of protocols, two at a time, and with as few changes as possible should be reconfigured to support another set of protocols. The software should also have enough flexibility to be expanded and include more protocols. This implies among all that the software must to be structured over well-defined layers, similar to the OSI model, and interfaces to make it easy to implement new future protocols. Common to all gateway variants is that Multilink/INOC is always one of the two protocols.

3.1 Requirements of the gateway

The requirements on the gateway listed below are compiled from the current gateway specifications, [5], [6] and [7]. The related solutions to those requirements are explained in detail in section 3.2.

1. The software should have a structure based on well-defined and separated layers representing the structure of a general protocol. It should be easy to make changes concerning what information that should be handled. The software should also be constructed in such a way that new protocols would be easy to implement and use.
2. Frames
    - 2.1. The NMEA 2000 protocol should be able to handle multiframes (i.e. frames that contains more than 64 bits data).
    - 2.2. Instances between Multilink/INOC and NMEA 2000 should be handled as described in section 3.1.1.
    - 2.3. A frame should only be sent if it is populated by any signal that is not timed out (see point 3.2). See also description in section 3.1.2.
    - 2.4. Parts of a data field that is not defined by a gatewayed signal should be able to be chosen arbitrary.
3. Timing
    - 3.1. Frames should have an individual given update rate.
    - 3.2. A signal should be considered as timed out if it isn’t received within ten times its update period. A timed out signal is assigned with an error code.
    - 3.3. The maximum latency for the Multilink/INOC-NMEA 2000 gateway shall be 50 ms. Defined as the time from when a signal is received to when it is ready to be sent.
4. Gateway
    - 4.1. The information for the gateway should have a structure such that it is easy to maintain.
    - 4.2. The gateway should translate error codes between the protocols.
    - 4.3. The values of the signals should be checked before and after conversion to be within the given minimum and maximum values for that signal.

3.1.1 Instance handling

Both Multilink/INOC and NMEA 2000 have some messages that have several instances. That is, a message defined by a unique PGN can be used by up to four engines and thus those messages need to be distinguished by an instance number.

There is however a difference concerning the position in the data frame where the instance number is stored. On Multilink/INOC instance information is the last 8-bits in the identifier and on NMEA 2000 the instance is in the data field as the first 8-bits.

In frames where both transmitting and receiving side have instances defined they should be mapped according to Table 1. When a NMEA 2000 PGN, without any instances defined, should get data from a Multilink/INOC PGN, that has instances defined, data should only be sourced from one instance on Multilink/INOC, according to the following priority (highest priority first) 0x00, 0x01, 0xEF, 0xF0, lowest source address.

Table 1 Specifies the relation of Multilink/INOC network address (identifier) and NMEA 2000 instance (data field) in different engine installations.

	Multilink/INOC address - NMEA 2000 instance			
Engine	*Single engine installation*	*Twin engine installation*	*Triple engine installation*	*Quad engine installation*
Port	0x00 - instance 0	0x00 - instance 0	0x00 - instance 0	0x00 - instance 0
Centre, Port centre	N/A	N/A	0xEF - instance 1	0xEF - instance 1
Starboard centre	N/A	N/A	N/A	0xF0 - instance 2
Starboard	N/A	0x01 - instance 1	0x01 - instance 2	0x01 - instance 3

3.1.2 Time-out behavior

Data frames are received periodically from the networks. Every frame has its own defined update rate and if the signal has not been received for a time equal to ten times the update rate it is timed out and is considered to be old. It shall then not be sent through the gateway to the other protocol. However, an outgoing frame may contain both up-to-date signals as well as timed out signals. The essence of point 2.3 is that the frame should though be sent as long as at least one signal is up-to-date.

3.2 Software design

Solutions in this section make cross-references to specific requirements, i.e. the requirements listed in section 3.1. A detailed description of the software is contained in the document [8].

The Gateway basically consists of three different layers as illustrated in Figure 7. The protocol layer (in this case CAN) handles the reception and extraction of signals from raw CAN data as well as packaging of signals into a CAN frame and the transmission of it. The protocol layer corresponds to the physical layer and the data-link layer in the OSI model. The signal layers main purpose is to continuously update the validity (i.e. the usefulness) of signals and to contain interface functions which can output the signal and its current validity state to neighboring layers. The signal layer is mainly an interface between the protocol layer and the gateway layer and has no direct representation in the OSI model. The gateway layer would represent the application layer in the OSI model.

The core functionality of the software resides in the gateway layer which contains databases that map the content from receiver to transmitter. This signal map database holds detailed information of every signals properties, which are instructions to the gateway on how to process the signal.

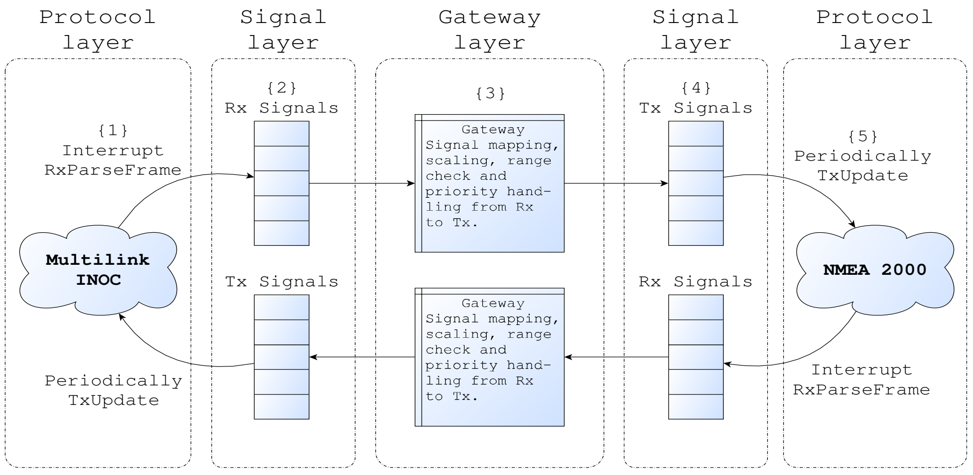

Protocol layer
Signal layer
Gateway layer
Signal layer
Protocol layer
{1} Interrupt RxParseFrame
{2} Rx Signals
{3}
Gateway Signal mapping, scaling, range check and priority hand-ling from Rx to Tx.
{4} Tx Signals
{5} Periodically TxUpdate
Multilink INOC
NMEA 2000
Tx Signals
Rx Signals
Gateway Signal mapping, scaling, range check and priority hand-ling from Rx to Tx.
Periodically TxUpdate
Interrupt RxParseFrame

Figure 7 General structure of the software for the generic gateway. Here exemplified with Multilink/INOC and NMEA 2000.

To further emphasize the overall structure of the software, an explanation of the path a signal travels on its way from receiver to transmitter follows. Consider, as an example, the transmitter being on the Multilink/INOC side and the receiver on the NMEA 2000 side, as in Figure 7.

Upon reception of a CAN frame, an interrupt {1} is triggered and a table is looked up, which lists the identifier of all frames that are relevant to the receiver, with the purpose of finding a match. If a match occurs, the signals specified for the current frame are extracted and copied to a specific location in the receiver signal table {2}.

The task of the gateway layer is to check the receiver signals every millisecond and as soon as new data is available in the receiver signal table {2} it is processed by the gateway {3}. This fulfills the requirement 3.3 in section 3.1, to be able to send the signal within 50 ms from reception.

Incoming data may be copied from the receiver signal table {2} to the transmitter signal table {4} or discarded, this depends on if the signal pass certain checks. The checks can be both mandatory and conditional. A typical mandatory check is the range check applied both before and after signal scaling. If the signal is out of range, it is discarded.

If conditional checks are defined, then the signal may be discarded or further processed before being copied to the transmitter signal table {4}. For example, conditional checks apply to different signals that describe the same property. The signal that is preferred over the other is always copied. However, the low priority signal is only copied if the high priority signal does not exist. In this way, two signals on the receiver can be mapped to one signal on the transmitter and the best signal is always chosen based on availability.

The content of the transmitter signal table {4} is used to build output frames. The transmission procedure {5} monitors which signals are available and valid, builds and sends all frames which can be derived from those signals. A signal is valid for a certain amount of time after reception. After that, the signal is considered timed out and not further transmitted.

3.2.1 Software reuse

The software platform mentioned in section 1.4 offers a convenient interface for handling CAN frames and their associated signals. The procedure focuses on the signals rather than on the frames: signals can be individually monitored independently on which frame they belong to. This principle is one of the architectural key elements.

Another feature of this software is that it keeps track of the number of engines that is broadcasting. With this knowledge the generic gateway can assign the (static) instance signals with the right value. Thus requirement 2.2 (instance handling) is fulfilled. Static signals are further described in section 3.2.5.

3.2.2 Describing a signal

There is no one to one mapping between frames in two CAN protocols (Multilink/INOC and NMEA 2000) and certainly not between a CAN protocol and a serial protocol, such as NMEA 0183. The signal-oriented way of handling data frames is preferred and applied for the development of the generic gateway. Above this signal management method, another layer (the gateway layer) is needed to generate output signals according to a protocol from input signals formatted according to another protocol. In order to accomplish this, a generic description of a signal and its properties is needed for the gateway to be able to match the incoming and outgoing signals.

For every incoming signal, a row in a table is created, which lists a number of properties, one for each column. The *source* is defined as an incoming signal that should be copied to the *destination* which is an outgoing signal. The following properties apply:

- Scale of source signal
- Minimum and maximum value of source signal
- Multiplier, divider and offset for rescaling of signal
- Message options (optional)
- Pointer to function (optional)
- Pointer to signal destination
- Scale of destination signal
- Minimum and maximum value of destination signal
- Message options data (optional)
- Timeout

Having all the information collected within one single row in a table, makes it easy to verify which signals are handled by the gateway and also easy to add/delete signals in connection with software updates (see requirement 1 and 4.1). That is part of why this structure is selected. The properties describing a signal connection are further explained below.

3.2.3 Signal scaling

The scaling of a signal (at both source and destination) is an important property which describes the bit length and sign of the signal and also indicates which error codes the signal should be compared to. The property is used to check whether or not the signal has a value corresponding to an error code. It is also used in conjunction with the minimum and maximum property (both source and destination) to set the correct error code to a signal if it turns out to be out of range in the min/max check. The min/max check is performed both before and after the scaling by multiplier, divider and offset since the source and destination protocols may have defined different limits for the corresponding signals. The scaling itself is executed since the source and destination often have different resolution for the signals. In summary, these properties are used by the gateway layer to ensure that requirements 4.2 and 4.3 are fulfilled.

3.2.4 Signal validity

The timeout property is used by the signal layer as a reference to compare against when outputting the signal’s state. The signal is either old or fresh depending on the time since it was last updated, ten times the timeout property is considered old, see requirement 3.2. As a consequence of the scale of a signal, the output from the signal layer could also be that the signal value is *not available* or *out of range*. This information (the signal itself, the validity state and the timer) is copied by the gateway function between the source and destination, thus the transmitter can evaluate if a frame should be sent satisfying requirement 2.3, i.e. a frame should only be sent if it contains at least one signal that is fresh.

3.2.5 Optionals

The optional points in the property list above are used for signals that only shall be copied if certain conditions are fulfilled. For example, a signal should not be copied if a specific signal exists on the destination or if a higher priority signal exists on the source. Optionally a pointer to a function could be added in order to support the rarest signal handling routines or to call a controller function for the signal.

3.2.6 Transmitter and receiver

A transmitter frame consists of the following parts, each frame having a defined position in a transmitter frame table:

- Message identifier
- Message options
- Signal table
- Update period and timer

The transmitter periodically searches the outgoing frame table for frames with an expired timer and upon match it resets the timer to the specified update period, according to requirement 3.1 (frames have an individual given update rate), and tries to send the frame if it has at least one fresh signal as mentioned before. Signals that are old in that frame are sent with the error code: not available.

It is common that the whole 64 bits of the data field in a frame is not defined by incoming signals. However, it is still important that those gaps have a defined value and according to [5] this value needs to be able to be chosen arbitrary. The signal type that fills the gaps can neither be fresh nor old since this would conflict with requirement 2.3, i.e. a frame should only be sent if it contains at least one signal that is fresh.

To be able to choose undefined parts (the gaps) of a data field arbitrary (see requirement 2.4) another signal state is implemented, which is called *static*. A static signal can be initiated to an arbitrary value and is considered by the gateway layer as a signal that is neither fresh nor old. In practice this means that a frame that is to be sent can be defined by both fresh incoming signals, static signals and possibly old signals assigned to an error code. However, a frame that contains no fresh signals, but possibly both old and static, will not be sent even if the data field is defined by some static signals.

In order to satisfy requirement 2.1 (regarding multiframes) a frame can be set with a certain multiframe option. In that case the transmitter function knows that the signal table contains signals that sum up to more than 64 bits of data and the transmitter will send as many single frames as required for all signals to be sent. This way of handling the multiframes is referred to as fast-packet multiframes in the NMEA 2000 standard.

A receiver frame is defined similar to the transmitter frame except that the update period is not used. Instead another field is added defining an optional function pointer that, if it exists, can be called upon reception of a frame to handle very specific frames. For example the address claim frame (see section 2.5.2) is treated in this way.

3.3 Discussion

A generic gateway software platform has been developed in C for the new 32-bit target processor family from ST Microelectronics. The code has been developed and written in such a manner that it should be easy to read and understand. This type of code is often not the most optimal to execute and store in the memory but a clear and easy structure have many other advantages. One of the advantages is that further development time and effort is minimized as a new person easier understands the code and its purpose. This strategy is enabled by the vast executing power and memory of modern microcontrollers. Another important condition to use this strategy is the low volume of units produced and hence per unit cost is allowed to be higher if developing cost could be minimized.

During the development of the software a developing board with a STM32F107VC processor has been used. This processor is one of the more advanced 32-bit processors from ST Microelectronics. This processor has many internal functions, there among two CAN channels and Ethernet. The gateway in its current application does not need Ethernet and hence the STM32F105xx processor could be used. This processor is almost identical to the STM32F107xx but without the Ethernet. The STM32F105xx comes in two different pin numbers 64-pin or 100-pin and each of them has three memory sizes: 64kb, 128kb and 256kb. Depending on code size and the number of I/O-pins needed one of the six variants is chosen.

4 Trim controller case study

As mentioned in the background (section 1.1), a motor boat shows a tendency to roll and pitch, in consequence of the combined driving force from the engines with the hydrodynamic pressure on the hull. The tendency to roll/pitch can be compensated for by trim planes mounted astern. Trim planes affect the roll/pitch angles of the boat by exerting hydrodynamic pressure, which can be controlled by modifying the trim plane angle. One may describe the operation of the trim planes by recalling how a rudder works: the trim planes may be seen as rudders pivoted on the horizontal axis rather than on the vertical one.

For optimal tilt compensation, the angle of the trim planes has to be adjusted in function of a number of quickly variable parameters, boat speed and wind before all, an operation that most boat users find not at all intuitive. A function that performs such adjustments automatically is therefore extremely appreciated. This is the function implemented by CPAC System's ATS (AutoTrim System).

4.1 AutoTrim System

The core element of the ATS is the ACU (AutoTrim Control Unit) and its purpose is to control the roll and pitch. This unit contains the electronics, control functions and the communication to a control panel and a pair of trim planes over a serial protocol called J1587. It is also connected by NMEA 0183 to a dedicated GPS receiver, which feeds the ACU with speed information. The speed is very important for the controller since the control signals to the trim planes depend on speed through water, and it is also used by the controller to determine when to activate the function. For example the pitch controller is only activated for speeds greater than six knots – in fact, the trim planes do not have any effect on the boat angle unless the boat is moving.

The ATS is a fully independent system and do not have any connections to other buses on the boat. The driver's interface to the system is through the control panel where either manual or automatic (closed-loop) control can be selected.

To determine the roll and pitch angles a two axis inclinometer build upon accelerometers is connected to the microcontroller inside the ACU. The signal from the accelerometers is filtered through a first order low-pass filter with a cut-off frequency of 0.5 Hz. Since accelerometers are used, which are disturbed by horizontal accelerations (accelerations perpendicular to the gravitational force) from the boat, acceleration data from the GPS is used to compensate for this. From the information acquired from the sensors, the microcontroller calculates the control signals to the trim planes.

4.2 Problems with the existing ACU

From [9] it is known that the existing ATS needs a more accurate and less delayed estimate of the roll and pitch angles. The GPS receiver gives a too delayed acceleration signal and thus cannot compensate for the horizontal boat acceleration correctly. This becomes notable when the boat accelerates and is about to go up on plane. The preferred functionality is that the trim planes are in their lowest position until the boat is on plane and then fast risen, then the ACU is switched over to closed-loop control. This only works for boat sizes close to the size that the function is optimized for. For larger boats the system is too slow. For smaller boats that get up on plane easy, the system is not able to detect the acceleration fast enough and keeps the trim planes in the lowest position too long. This result in the stern of the boat is lifted and the bow is pushed down which makes the boat to bow steer, a dangerous situation where the boat yaw becomes unstable.

Another problem is that the microcontroller used in the ACU is rather outdated, slow and only an 8-bit microcontroller. Due to this the calculations are very simplified and the controller includes discrete conditions which give rise to sharp transitions. For example, the controller works in two different modes depending on if the boat turns or not. The transition between these modes is sharp and thus the detection of small turns may fail and thus also the proper compensation. Also when an autopilot is installed and used in a boat together with an ATS, the systems seem to counteract each other which give rise to oscillations in the roll and yaw angle, [9].

In summary two areas of improvement can be identified. The estimated roll and pitch based on data from the heavily low-pass filtered accelerometer and the rather slow GPS turns out to be too inaccurate or too delayed. Thus better estimates of the roll and pitch angles are required. The second area involves the controller which does not cope with various boat sizes, weight to horsepower ratios and trim plane sizes as well as the detection of small turns.

4.3 Estimating roll and pitch: test results

The behavior of an accelerometer, a gyroscope and different filtering algorithms is examined in order to be able to propose improvements. In Table 2 a summary of the sensor attributes are repeated.

Table 2 Main attributes of sensors used to estimate roll and pitch.

Sensors	**Positive**	**Negative**
Accelerometer	Provide absolute angle information.	Affected by horizontal acceleration.
Gyroscope	Insensitive to horizontal accelerations.	Measures angular speed. Small bias error leads to drift upon signal integration.
Horizontal acceleration sensor	Can be used to compensate an accelerometer for horizontal accelerations.	Operation principle implies signal derivation, i.e. delay.

To evaluate the performance of different configurations in estimating the roll and pitch angles, a test rig is used, depicted in Figure 8. The core elements in the rig are an accelerometer and a gyroscope. Also a potentiometer is attached to the board and fastened in the steel frame which makes it possible to measure the true angle and thus it is able to determine the quality of the estimation from different setups of sensors and filtering algorithms.

The three quantities are acquired by the development board through SPI (Serial Peripheral Interface) and forwarded via RS-232 to a PC running MATLAB with a sampling frequency of 200 Hz. The benefits of running the calculations in MATLAB over the microcontroller are mainly the ability to save data series and visualize the results. The associated MATLAB script is found in appendix B.

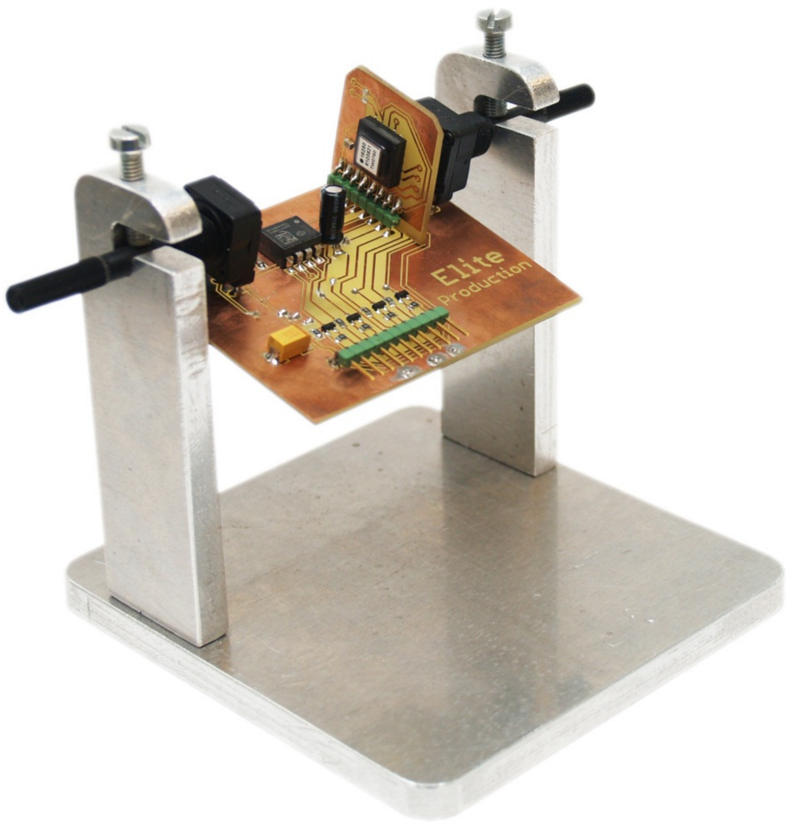


Figure 8 The test rig with the sensor board attached to the aluminum frame.

4.3.1 Accelerometer

According to section 2.2 an accelerometer used for measuring inclination (in this case roll and pitch) is disturbed by horizontal acceleration. Since one of the main tasks of the ATS is to help the boat up on plane during acceleration, the boat acceleration becomes a great disturbance when estimating the pitch. The situation is illustrated in Figure 9.

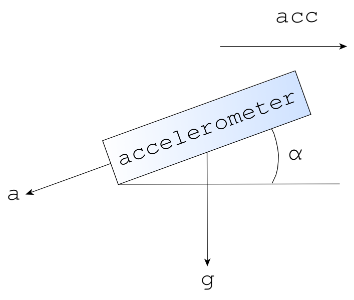


Figure 9 A tilted (α) accelerometer sensing an acceleration a when affected by gravity g and the horizontal acceleration acc.

The formula for calculating the angle thus becomes,

$$\alpha = \sin^{-1}\left[\frac{a - (\mathrm{acc} * \cos\alpha)}{g}\right] \tag{1}$$

and when approximating $\cos\alpha = 1$ for small α it can be rewritten as:

$$\alpha = \sin^{-1}\left[\frac{a - acc}{g}\right] \tag{2}$$

From this it is concluded that an accelerometer is directly affected by horizontal accelerations and therefore also vibrations. The term “horizontal acceleration” is used to underline the fact that it is referred to acceleration in the direction the boat moves toward. In fact, an accelerometer is affected by all accelerations. But in order to measure for example the roll and pitch angles correctly, the ideal case would be if only the gravitational force was present. When used in a boat, the present accelerations are mainly the ever existing gravitational force and the horizontal boat acceleration that needs to be considered.

The vibrations are easy suppressed by low-pass filtering while the horizontal acceleration implies that the estimated angle will be smaller than the true angle if using equation (1). This is the case in Figure 10, the horizontal movements give rise to faulty readings. It should however be noted that the average of the accelerometer output coincides with the true potentiometer angle. In the current ACU, the horizontal acceleration is compensated by subtracting the differentiated speed reading from a GPS as in equation (2) (i.e. the horizontal acceleration of the boat). The problem is that a big dead-time in the estimate of the horizontal acceleration from the GPS makes the overall pitch control system to react too slow [9].

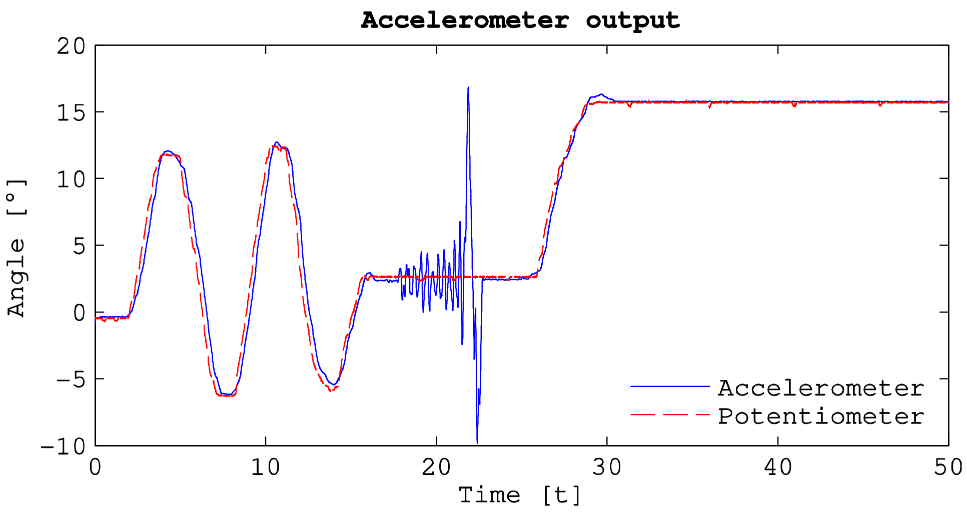


Figure 10 The accelerometer output from the test rig. At t = 17 seconds the rig is disturbed by vibrations and at t = 22 seconds a sudden move is made.

One possible solution to this is to use a better sensor to measure the horizontal acceleration, for example a faster GPS, a paddle wheel or by using information about motor speed and gear. Both the fast GPS and the paddle wheel increases cost if those are not already installed in the boat. Using the motor speed and gear is further discussed in section 4.4.

4.3.2 Accelerometer and gyroscope

Another sensor often used to measure angles is the gyroscope (see section 2.1) which is not affected by accelerations as much as the accelerometer. The gyroscope therefore does not have the problem concerning horizontal acceleration. The output of a gyroscope is angular rate and thus needs to be integrated in order to get the relative angle. However, the zero angular rate level varies slightly with time and temperature and hence the angle (integral of the angular rate) drifts away as shown in Figure 11.

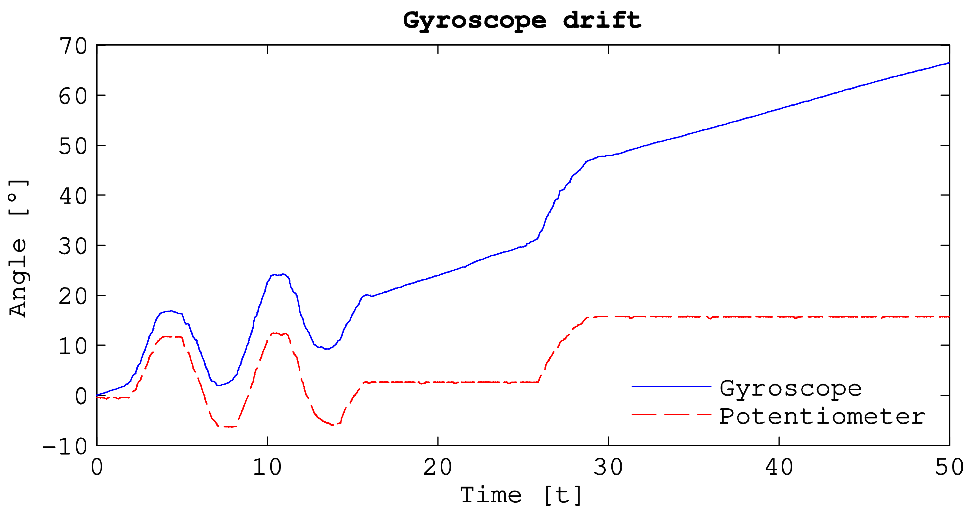


Figure 11 The angle (integral of angular rate with nominal bias subtracted) from the gyroscope gives an error called drift. The potentiometer reflects the true angle.

In summary, the accelerometer and gyroscope have properties that complement each other and a fairly good estimate of the angle can be achieved by using both types of sensors. The main idea is to integrate the gyroscope output (which is not so sensitive to horizontal accelerations and vibrations) and correct the bias using the accelerometer which averaged over a long time (relative to the time constant of the gyroscope response) gives an accurate reading of the true angle and hence also the bias of the gyroscope.

This is realized with a Kalman filter based on the discrete time state space model given by:

$$x = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \qquad u = \omega$$

$$x_{k+1} = \begin{bmatrix} 1 & -h \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} h \\ 0 \end{bmatrix} u_k$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k,$$

where α is the angle, β is the bias of the gyroscope and ω is the angular rate of the gyroscope. The model simply describes the angular rate from the gyroscope being integrated with the related bias subtracted. The modeled bias is however constant. This is managed by the Kalman filter which corrects both the angle and the bias with the estimation error.

The Kalman filter equations are divided in two steps, predict and update [10]:

$$\hat{x}_{k+1|k} = \begin{bmatrix} 1 & -h \\ 0 & 1 \end{bmatrix} \hat{x}_{k|k} + \begin{bmatrix} h \\ 0 \end{bmatrix} u_k$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K(y_k - \begin{bmatrix} 1 & 0 \end{bmatrix} \hat{x}_k)$$

Here $\hat{x}$ is the state estimation vector and $y_k - [1 \quad 0]\hat{x}_k$ is the estimation error (also called innovation) formed by subtracting the estimated angle $\alpha = [1 \quad 0]\hat{x}$ from the measured accelerometer angle y.

This filter, tuned by properly selecting the covariance matrices for the process and measurement noise respectively, gives a system that very much relies on the gyroscope which gives a good angle estimate if the accelerometer measurements compensates for the bias of the gyroscope. In other words, by using the gyroscope to estimate fast changes and the accelerometer for slow variations, a good estimation can be made. However the problem is to determine when the boat is not accelerating since that disturbs the angle measurements from the accelerometer. That is why the filter is tuned to be rather insensitive to accelerometer measurements and to rely more on the gyroscope measurements. Another way of treating this problem is described in section 4.3.3.

The Kalman filter output is shown in Figure 12. As can be seen the filter follows the true angle well, though with a delay of about 200 ms caused by the FIR (Finite Impulse Response) low-pass filter which the accelerometer and gyroscope measures are filtered through. This delay, in comparison to the time constants that characterizes a vessel dynamic behavior, is rather irrelevant.

Also in comparison to Figure 10 and Figure 11, which shows the pure reading from each sensor, the result in Figure 12 shows how the different sensor can complement each other by compensating for their specific shortcomings. The same data series is used in these three figures.

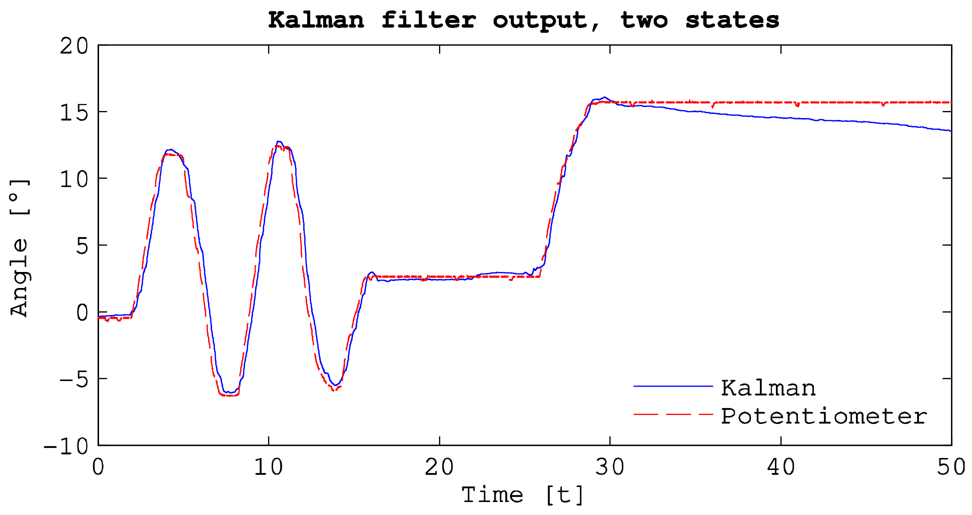


Figure 12 The filtered output follows the true potentiometer angle very well. However, the kalman filter output is delayed about 200 ms since the accelerometer and gyroscope is low-pass filtered.

Although the output of the Kalman filter (Figure 12) is good it can be seen that the angle decreases linearly for t > 30 s. This is related to the bias of the gyroscope being dependent on the angle. This statement is verified by letting the bias (β) settle for different angles. It is found that the bias changes approximately 0.01α °/s (i.e. the gyroscope output should be corrected by 0.01 °/s for every degree α the test rig deviates from the horizontal). This dependence is called “bias linear acceleration effect” and is specified in appendix A.2.

In order to cope with this the sensor model is extended to three states and slightly modified:

$$x = \begin{bmatrix} \alpha \\ \beta_0 \\ \beta_k \end{bmatrix}, \qquad u = \omega$$

$$x_{k+1} = \begin{bmatrix} 1 & -h & h \\ 0 & 1 & 0 \\ 0.01 & 0 & 0 \end{bmatrix} x_k + \begin{bmatrix} h \\ 0 \\ 0 \end{bmatrix} u_k$$

$$y_k = [1 \quad 0 \quad 0] x_k,$$

where α still is the angle, β_0 is the zero level bias and β_k is the proportional (anlge dependent) bias. Note the 0.01 term in the lower left of the state matrix which represents the correction of the bias due to the angle deviation α from the horizontal. The related predict and update equations are as follows:

$$\hat{x}_{k+1|k} = \begin{bmatrix} 1 & -h & h \\ 0 & 1 & 0 \\ 0.01 & 0 & 0 \end{bmatrix} \hat{x}_{k|k} + \begin{bmatrix} h \\ 0 \\ 0 \end{bmatrix} u_k \tag{3}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K(y_k - [1 \quad 0 \quad 0]\hat{x}_k) \tag{4}$$

The output of the modified Kalman filter is illustrated in Figure 13 and gives a satisfying output, still with the same data series as the previous figures. The signal is however still delayed by 200 ms, caused by the FIR-filter.

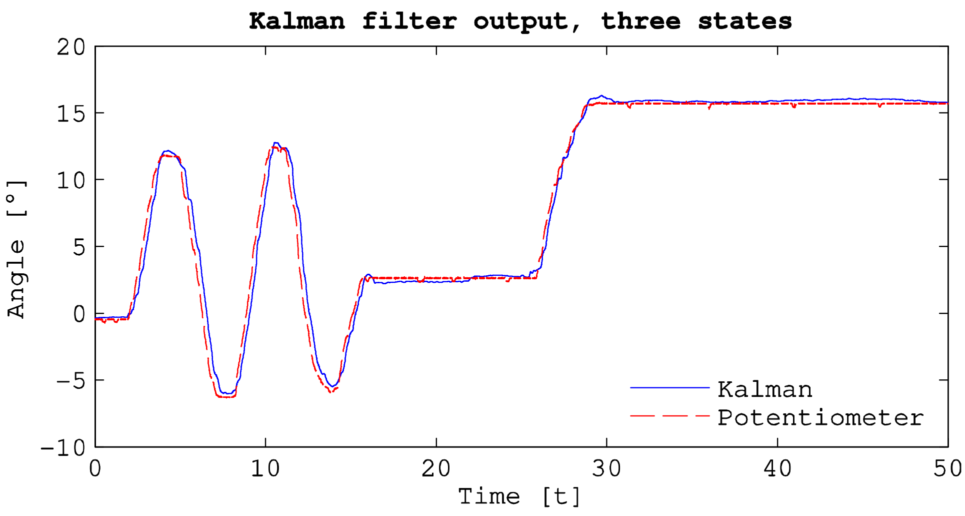


Figure 13 The output from the modified Kalman filter gives a satisfying estimate of the angle despite the effect of horizontal acceleration on both the accelerometer and gyroscope.

4.3.3 Accelerometer, gyroscope and horizontal acceleration sensor

As briefly mentioned in section 4.3.1 the accelerometer reading can be compensated for horizontal accelerations for example by subtracting the boat acceleration measured from a GPS receiver. The horizontal acceleration can however be measured or estimated in many ways. In practice that information could be estimated from data sourced from a GPS, a speed-through-water sensor (paddle wheel, hydrostatic pressure sensor) or through engine speed and gear.

The filter presented in the last section can cope with relatively small and short time horizontal accelerations due to the filter being rather insensitive to the accelerometer in a short time perspective. However, if they are long lasting and with great magnitude the accelerometer will though affect the filter too much and the angle estimate can no longer be trusted. By using a horizontal acceleration sensor, compensation during long time boat acceleration can be applied and hence improves the angle estimation during these conditions. Depending on the quality of this measurement two methods can be applied; direct compensation by subtraction as in equation (1) or (2) or by cease running the update equation (4) and only run the predict equation (3) in the Kalman filter.

The advantage of the second method is that it works better than the first when the horizontal acceleration data is delayed or erroneous as in the case with a GPS receiver. Since this method only needs an indication for when the acceleration is above a certain level the horizontal acceleration sensor can be very simple. A test is made with a simulated horizontal acceleration of 1 m/s^2 as shown in Figure 14. The result of the three state Kalman filter with and without horizontal acceleration detection is shown in Figure 15.

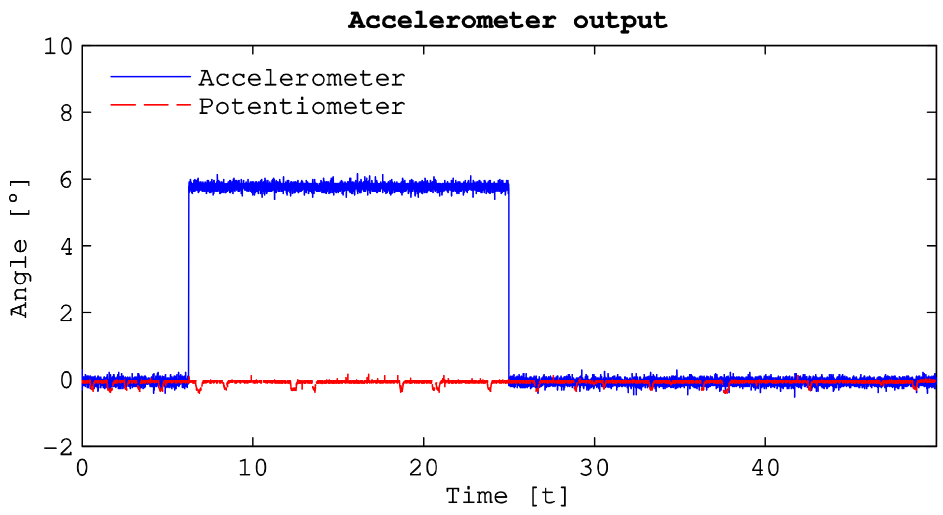


Figure 14 A simulated horizontal acceleration disturbance of 1 m/s^2 is applied to the accelerometer reading resulting in faulty angle measurement for t between 6 s and 25 s.

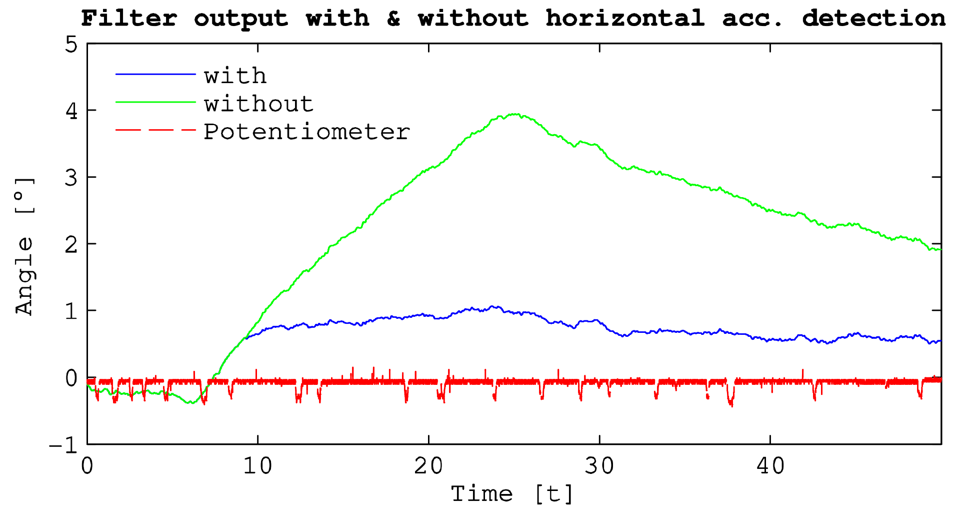


Figure 15 The effect of the horizontal acceleration detection is almost an unaffected angle output. In this case the detection is delayed three seconds (to simulate a slow GPS) and the update part of the Kalman filter are thus shut off at t = 9 s.

The reason to why this works is that the bias only varies very slowly and turning of the update part of the filter will not have a big effect since the innovation is close to zero when no or little horizontal acceleration is present. This final setup is illustrated in a block scheme in Figure 16 with the horizontal acceleration sensor being a GPS receiver.

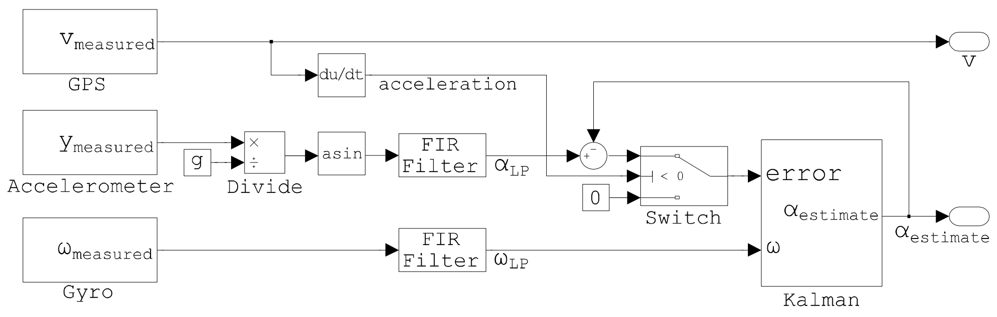


Figure 16 The resulting filter with a third order state space model and horizontal acceleration detection.

4.4 A new ACU

In the development of a new ATS a good estimation of roll and pitch is essential for the performance. Besides this estimation there are also other improvements that can be made on system level. Connecting the ACU to a synchronization bus is an important feature of the new ACU. A synchronization bus is a communication channel where many signals from different systems are collected in order for the signals to be available to systems that can take advantage of using them.

The most obvious benefit of connecting the ACU to the synchronization bus is to use speed data from the bus instead of having a dedicated sensor integrated with the ATS. The available information on the synchronization bus is however not always the same, it varies depending on the installed equipment on the boat. The speed needed for the controller of the ACU could be estimated by for example fusing data from the engine (speed and gear), tridacer (a combined sensor for depth, speed and temperature) and a GPS receiver if available on the synchronization bus in a similar manner as with the Kalman filter for the roll and pitch angles.

Another benefit of connecting the system to the synchronization bus is that other system can get access to the roll and pitch angle of the boat. For example this information can be valuable to prevent dangerous leaning of the boat by limiting motor power when such a position is detected. Also a communication between the autopilot and the ACU could then be established to prevent the systems from counteracting each other as mentioned in section 4.2.

A new ACU based on the STM32 microcontroller compared to the current controller has much more computing power and can more easily implement advanced control functions. The existing solution with several discrete conditions, as mentioned in section 4.2, could then be avoided. A controller with more smooth transitions could be implemented and hence eliminate some of the problems related to discrete conditions, especially those during turns. By replacing the discrete conditions for how fast the trim plane angle can be changed, a faster and smoother behavior can be achieved which could help eliminate problems during acceleration and normal run.

The final solution for a new ACU depends on which information that are available on the synchronization bus which in turn will decide which sensors that are integrated in the ACU. The availability of low cost accelerometer and gyroscope MEMS-sensors makes it easy to integrate two or even three axis sensors to get full information of the boat position. This would make the ACU to be able to also get information about turn rate from the gyroscope and hence not be dependent on the slow GPS receiver for detecting turns.

The ACU is a good example of the benefits there are when integrating systems by combining the functionality of a gateway with external devices.

5 Conclusion

5.1 A multi-purpose gateway

Translation from one protocol into another is a main functionality of the developed gateway. A well-structured software platform has been designed for the STM32 microcontroller which makes it easy to modify the protocols the gateway handles, or even to implement new ones.

The principles defined within the Open System Interconnection (OSI) model were applied as basis to the software design and the application were therefore structured across a number of different layers, featuring well defined and standardized interfaces. This was particularly important since the work accomplished is now delivered to a company that will turn that into a product. Good software coding practice is a definite requirement in such a context.

The validity of this approach is demonstrated also by the fact that the device operates properly. The implemented code support CPAC Systems' Multilink/INOC protocol, as well as the international standards NMEA 2000 and NMEA 0183.

5.2 Estimating roll and pitch: from sensors to Kalman filtering

Kalman filtering is the key element of the solution and the state space model that models the sensor system, consisting of an accelerometer and a gyroscope, contains three states. Basically, the state that estimates the angle is calculated by integrating the gyroscope measure and subtracting the gyroscope biases, one slowly time-varying bias and one bias that is proportional to the angle from the horizontal. These calculations are called *prediction*.

The accelerometer compensates for errors in the second step of the calculations in the Kalman filter, called *update*. It mainly compensates (tunes) the time-varying gyroscope bias. In summary this gives a responsive sensor system, thanks to the gyroscope, which also corrects for slowly time-varying parameters, thanks to the accelerometer. By being able to turn of the update calculations when horizontal accelerations are detected, the system can cope with horizontal boat accelerations of great magnitude and duration. These would otherwise disturb the angle estimation since the nature of the sensor model makes the angle estimate to always converge towards the accelerometer measurement.

All in all, it has proven to be successful to combine an accelerometer and a gyroscope for the purpose of estimating the roll and pitch angles of the boat correctly, despite the sources of error that was identified. The proposed solution is schematized on Figure 16.

It has also been shown that a horizontal acceleration sensor is needed. However since these data often are available on the synchronization bus, the ACU does not necessarily need to have a dedicated GPS. A dedicated GPS is only needed in rare cases when the boats buses do not hold horizontal acceleration information.

5.3 Combining a gateway with filtering functionality

Sensor fusion functionalities similar to the Kalman filter estimating roll and pitch is a concept that can be used to improve other signals in a boat. Also integrating this functionality together with a gateway can sometimes also be a good solution since the gateway is needed anyway. The AutoTrim system is a good example of this. The new gateway holds many of the functionalities needed in an ACU. There among the ability to communicate with different protocols and a powerful microcontroller capable of doing the necessary calculations. A finished gateway will however not be cost efficient to hold the necessary extra sensors needed for the ACU, however it might employ the same circuit board but without the expensive sensors mounted.

5.4 Future work

By connecting the ATS to the synchronization bus the overall system performance can be improved and the cost perhaps reduced. However, which amount of information that is to be considered as the minimum for installing an AutoTrim system, needs to be further investigated. This is a tradeoff between cost, quality and how available the system will be for boats with little equipment. With the available information specified, an improved ACU connected to the synchronization bus can be developed.

The model for the gyroscope could be extended to also include sensor die temperature since this affects the bias. The effect has been observed and the performance of the filter could be further improved by also considering this. The gyroscope used in the test has a built in temperature sensor.

The controller can be improved to be able to better cope with boats of very different size. Identifying the parameters that mostly depend on boat size and motor power is important in finding a suitable control strategy. Investigating adaptive or self-tuning controllers might be worthwhile.

References

[1]. **Barbour, N.** Gyroscope. *AccessScience.* [Online] ©McGraw-Hill Companies, 2008. http://www.accessscience.com.proxy.lib.chalmers.se.

[2]. **Awan, S A.** MEMS sensors. *AccessScience.* [Online] ©McGraw-Hill Companies, 2009. http://www.accessscience.com.proxy.lib.chalmers.se.

[3]. **Goodman, M S.** Data communications. *AccessScience.* [Online] ©McGraw-Hill Companies, 2008.

[4]. CAN Specification Version 2.0. Stuttgart : Robert Bosch GmbH, 1991.

[5]. CPAC Multilink CAN Interface Specification. Gothenburg : CPAC Systems AB, 2009.

[6]. NMEA 0183 Interface Unit Product Specification. Gothenburg : CPAC Systems AB, 2004.

[7]. NMEA 2000 Gateway Requirement Specification. Gothenburg : CPAC Systems AB, 2009.

[8]. **Byrhult, M., Stensson, V.** Generic Gateway Software Design Specification. Gothenburg : CPAC Systems AB, 2011.

[9]. Auto Boat Trim System 2.0 Product Specification. Gothenburg : CPAC Systems AB, 2008.

[10]. **Åström, K J., Wittenmark, B.** Prediction and Filtering Theory. *Computer-Controlled systems: theory and design.* 3rd. London : Prentice Hall, 1997.

References [5], [6], [7], [8] and [9] are confidential. Please contact CPAC Systems AB to obtain information about these specifications.

A Test rig

A test rig is constructed to be able to test the different proposed sensor configurations. The main components are an accelerometer (inclinometer) and a gyroscope. There are also a potentiometer fastened between the sensor board and the frame to measure true angle. The rig is only constructed with one degree of freedom.

A.1 Communication

Both the gyroscope and accelerometer uses SPI to communicate with the microcontroller. There are two potentiometers, one of them only acts as a support for the turning motion and hence not connected electrically, the other one is connected to the auxiliary ADC (Analog-to-Digital Converter) input on the gyroscope. The ADC input is an extra functionality of the gyroscope that could be read via the SPI. The data is read at 200 Hz from all the sensors to the developing board. From the board it is forwarded to the computer over RS-232. The computer receives the data and processes it in MATLAB, see appendix B.

A.2 Summary of sensor properties

Gyroscope Analog Devices ADIS16250

- Measuring range ±320 °/s
- 14 bit digital gyroscope sensor output
- 12 bit digital auxiliary sensor output
- 4.75 V to 5.25 V operating voltage
- SPI compatible serial interface
- Bias stability 0.016 °/s
- Bias linear acceleration effect 0.2 °/s /g

Inclinometer VTI technologies SCA61T-FAHH1G

- Measuring range ±30°
- 11 bit digital sensor output
- 4.75 V to 5.25 V operating voltage
- SPI compatible serial interface
- Typical non-linearity ±0.11°

A.3 Overview of design

Both the gyroscope and the accelerometer operate at 5 volt. Hence the supply voltage need to be 5 volt, however the signals has to be converted to be able to connect to the developing board which operates on 3.3 volts. The converters consist of one transistor and two resistors to convert from 3.3 volt to 5 volt and two resistors to convert from 5 volt to 3.3 volt. Both the gyroscope and the accelerometer has built in low pass filters fit for the application, the potentiometer have an external constructed filter with a cut off frequency of 10Hz.

A.4 Schematic

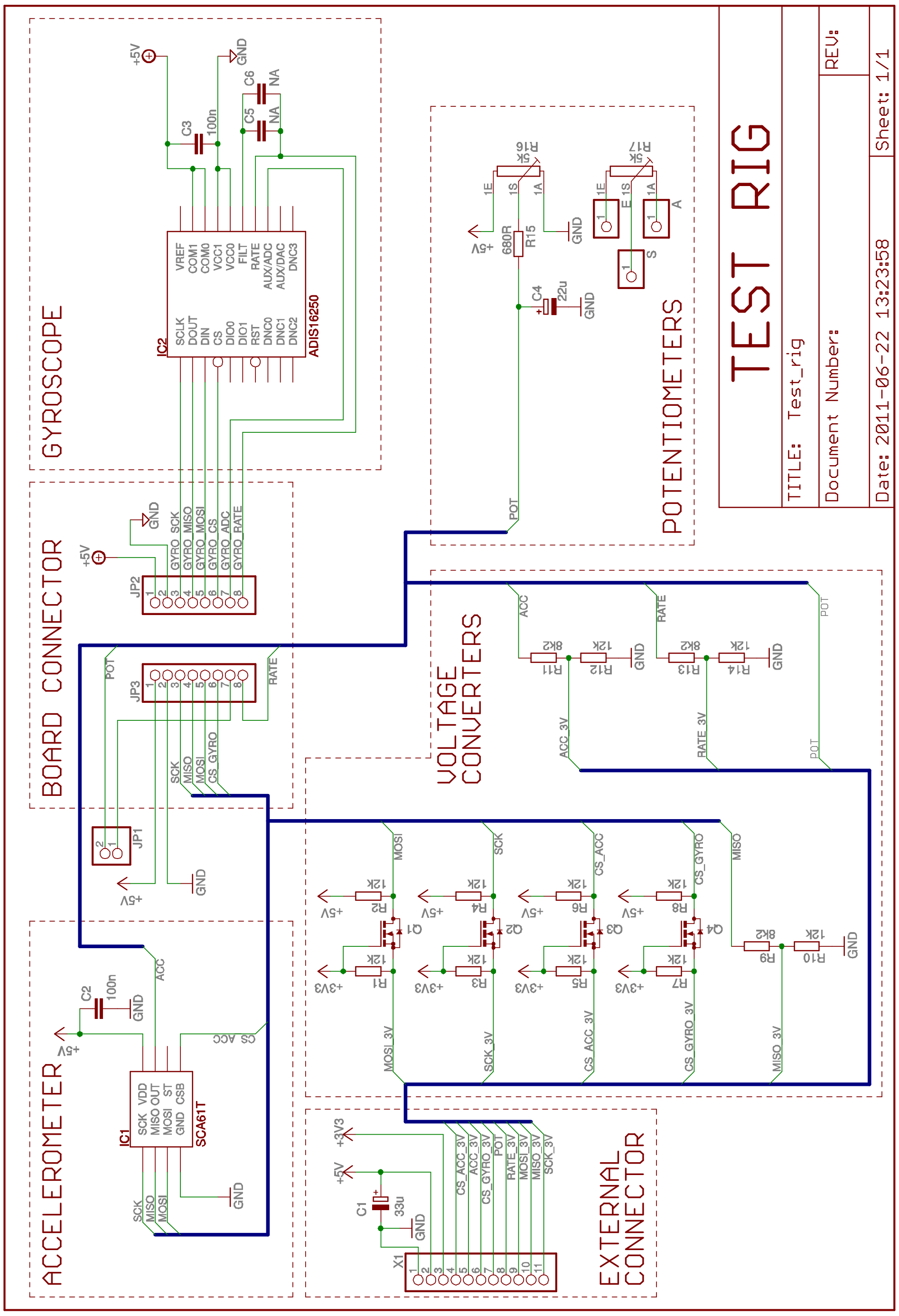


Figure A-1 The schematic of the test rig including a gyroscope, an accelerometer, two potentiometers and a couple of level converters.

B MATLAB script

The following script calculates and plots the filtered output in real-time. It continuously monitors the serial port where it acquires the 16-bit data in the order gyroscope, potentiometer and at last the accelerometer. The script includes calculating the low-pass filters as well as two different Kalman filters.

```
%% Initiate variables
clc
close all
clear all

runtime = 50;
updatetime = 50;
h = 0.005;                % Sample period
n = round(runtime/h);     % Number of samples
gyro = zeros(1,n);        % Gyro measurements
acc  = zeros(1,n);        % Acc measurements
pot  = zeros(1,n);        % Pot measurements
alp1 = zeros(1,n);        % 2 state kalman output
alp2 = zeros(1,n);        % 3 state kalman output
ghat = zeros(1,n);        % Gyro filtered
ahat = zeros(1,n);        % Accelerometer filtered
b0_1 = zeros(1,n);        % 2 state kalman bias
b0_2 = zeros(1,n);        % 3 state kalman bias
bk = zeros(1,n);          % 3 state kalman proportional bias
t = 0:h:h*(n-1);          % Time

%% Calculate kalman1 gain
A = [1 -h; 0 1];
B = [h; 0];
C = [1  0];

Plant = ss(A,[B [1;1]],C,0,-1);

Q = [1 0; 0 0.0001];      % Process noise 2x2
R = 100000000;            % Measurement noise 1x1

[kalmf1,L1,P1,M1,Z1] = kalman(Plant,Q,R);

k1_1 = M1(1);             % Kalman gain
k2_1 = M1(2);

%% Calculate kalman2 gain
A = [1     -h    h;
     0      1    0;
     0.01   0    0];
B = [h; 0; 0];
C = [1   0   0];

Plant = ss(A,[B [1 1;1 1;1 1]],C,0,-1);
```

```matlab
Q = [0.01   0       0;
     0      0.001   0;
     0      0       0.000001];  % Process noise 3x3
R = 100000000;                  % Measurement noise 1x1

[kalmf2,L2,P2,M2,Z2] = kalman(Plant,Q,R);

k1_2 = M2(1);           % Kalman gain
k2_2 = M2(2);
k3_2 = M2(3);

%% Calc LP-filter: PB <4 Hz, stop band >10 Hz 60 dB att
fs = 1/h;
[k,fo,ao,w] = firpmord([4 10],[1 0],[0.001 0.001],fs);
b = firpm(k,fo,ao,w);

%% Setup serial port
% Find a serial port object.
obj1 = instrfind('Type', 'serial', 'Port', 'COM1', 'Tag','');

% Create the serial port object if it does not exist
% otherwise use the object that was found.
if isempty(obj1)
    obj1 = serial('COM1');
else
    fclose(obj1);
    obj1 = obj1(1);
end
% Configure instrument object
set(obj1, 'BaudRate', 115200);
% Connect to instrument object
fopen(obj1);

%% Setup 3 figures for real-time plotting
scrsz = get(0,'ScreenSize');
width = scrsz(3);
height = scrsz(4);

figure('Position',[15 height/2+35 width/2-30 height/2-125])
HandleGyro = plot(t, gyro, 'red', t, pot, 'black');
legend('Gyro','Pot');
axis([0 (n-1)*h -40 40]);

figure('Position',...
       [15+width/2 height/2+35 width/2-30 height/2-125])
HandleAcc = plot([t' t'],[acc' pot']);
legend('Acc','Pot');
axis([0 (n-1)*h -40 40]);

figure('Position',[15 60 width/2-30 height/2-125])
HandlePot = plot([t' t'],[alp2' pot']);
legend('alp','Pot');
axis([0 (n-1)*h -40 40]);
```

```
hold on

set(HandleGyro,'Erase','xor');
set(HandleAcc,'Erase','xor');
set(HandlePot,'Erase','xor');

%% Initiate filters and set init-values

% Flush the data in the input buffer.
flushinput(obj1);

% Read 50 accelerometer values
for i = 1:50;
    fread(obj1, 4, 'uint8');

    raw = fread(obj1, 1, 'uint8') * 256;
    raw = raw + fread(obj1, 1, 'uint8');
    acc(i) = asin((raw - 1024)/1638)*180/pi;
end

% Load previous biases
load('b0.mat','b0_1init','b0_2init')
b0_1(1) = b0_1init;
b0_2(1) = b0_2init;

% Set init-values for filter
alp1(1) = sum(acc(1,1:50))/50;
alp2(1) = sum(acc(1,1:50))/50;
ahat(1) = alp1(1);

% Reset accelerometer readings
acc = zeros(1,n);

%% Run live plot
for i = 1:n;
    % Read gyro
    raw = fread(obj1, 1, 'uint8') * 256;
    raw = raw + fread(obj1, 1, 'uint8');
    if(raw > 2^15)
        raw = raw - 2^16 - 1;
    end
    raw = -raw;
    gyro(i) = raw*0.07326;

    % Filter gyro
    for j = 1:length(b)
        if(j<=i)
            ghat(i) = ghat(i) + b(j)*gyro(i-j+1);
            lock = j;
        else
            ghat(i) = ghat(i) + b(j)*gyro(i-lock+1);
        end
    end
```

```
% Warn for gyro overflow
if(abs(gyro(i))>320)
    warning('Gyro overflow at %d s.',i*h);
end

% Read potentiometer
raw = fread(obj1, 1, 'uint8') * 256;
raw = raw + fread(obj1, 1, 'uint8');
pot(i) = (raw*0.6105e-3 - 1.16540)*50.84100;

% Read accelerometer
raw = fread(obj1, 1, 'uint8') * 256;
raw = raw + fread(obj1, 1, 'uint8');

% Simulate horizontal acceleration
sim_on = 1;
if(i>n/8 && i<8*n/16 && sim_on == 1)
    accdetect = accdetect + 1;
    % Add horizontal acceleration of 1 m/s/s
    acc(i) = asin((raw - 1024)/1638 + ...
             1/9.82*cos(pot(i)*pi/180))*180/pi;
else
    accdetect = 0;
    acc(i) = asin((raw - 1024)/1638)*180/pi;
end

% Filter accelerometer
for j = 1:length(b)
    if(j<=i)
        ahat(i) = ahat(i) + b(j)*acc(i-j+1);
        lock = j;
    else
        ahat(i) = ahat(i) + b(j)*acc(i-lock+1);
    end
end

% Kalman1
% Update
if(accdetect == 0)
    alp1(i) = alp1(i) + k1_1*(ahat(i)-alp1(i));
    b0_1(i) = b0_1(i) + k2_1*(ahat(i)-alp1(i));
end
% Predict
if(i < n)
    alp1(i+1) = alp1(i) - b0_1(i)*h + ghat(i)*h;
    b0_1(i+1) = b0_1(i);
end

% Kalman2
% Update
if(accdetect < 200*3)
    alp2(i) = alp2(i) + k1_2*(ahat(i)-alp2(i));
    b0_2(i) = b0_2(i) + k2_2*(ahat(i)-alp2(i));
    bk(i)   = bk(i)   + k3_2*(ahat(i)-alp2(i));
end
```

```
    % Predict
    if(i < n)
        alp2(i+1) = alp2(i)-b0_2(i)*h + bk(i)*h + ghat(i)*h;
        b0_2(i+1) = b0_2(i);
        bk(i+1)   = 0.01*alp2(i);
    end

    % Update live plots
    if(rem(i,updatetime) == 0)
        set(HandleGyro(1),'YData',gyro);
        set(HandleGyro(2),'YData',pot);
        set(HandleAcc(1),'YData',ahat);
        set(HandleAcc(2),'YData',pot);
        set(HandlePot(1),'YData',alp2);
        set(HandlePot(2),'YData',pot);
        drawnow
    end
end

%% End
% Disconnect from instrument object
fclose(obj1);

% Save init-values for bias
b0_1init = sum(b0_1(round(3*n/4):n))/round(n/4);
b0_2init = sum(b0_2(round(3*n/4):n))/round(n/4);
save('b0.mat','b0_1init','b0_2init');
% Save acquired data
save(datestr(now,'dd-mmm-yyyy HH-MM-SS'))
```