NTNU

Norwegian University of
Science and Technology

Embedded control system for cybernetic wrist prosthesis

Inge Brattbakken

Master of Science in Engineering Cybernetics
Submission date: June 2010
Supervisor: Øyvind Stavdahl, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

In co-operation with University of New Brunswick, Canada, we have developed an electric prosthetic wrist. The mechanics are complete, and a control system based on two AVR microcontrollers and a CAN-bus interface is produced. The system, however, is not behaving as intended, and is therefore stated useless. This assignment involves designing a new, more robust control system, based on the original circuit.

1. Give an overview of the use of device profiles in communication protocols, in particular the CAN-protocol. Discuss the existing UNB-protocol, and suggest a way for this protocol to be enhanced with device profiles.

2. Present an overview of essential functions in a new control system for the writs prosthesis. This overview should be based on the existing functional specification, but modified according to the changed premises.

3. Describe a system architecture that supports the new functional specifications, and carry out a detailed hardware design for the system.

4. Realize and evaluate the system as far as time permits.

Assignment given: 11. January 2010
Supervisor: Øyvind Stavdahl, ITK

Preface

This thesis is my final work as a part of the two-year MSc-program at Norwegian University of Science and Technology (NTNU). The choice of subject was based on my wish to use my knowledge to make someones everyday life easier.

I would like to thank my supervisor Associate Professor Øyvind Stavdahl, for invaluable guidance during the semester. His eagerness on this subject has been a great motivation.

Last, but not least, I would like to thank my fellow students for making my stay at NTNU a true pleasure, both in educational and social manners.

Abstract

This master thesis treats the NTNU Rotary Wrist Device(NRWD). The wrist has been developed through several projects and assignments, all derived from Øyvind Stavdahls PhD thesis from 2002, which led to a document of functional specifications. This thesis follows from a specialization project by the writer during autumn 2009. The project looked into the error-prone circuit board that was made to control the NRWD, and came to the conclusion that it could not be used, and that a different approach was necessary. Based on the experience achieved by previous assignments a suggestion for the complete hardware circuitry has been designed.

In September 09 the first revision of a protocol for communication in electrical prosthesis were released. The protocol was developed at University of New Brunswick (from here referred to as the UNB-protocol), and is presented as a proposal for a standard communication protocol in the world of electrical prosthesis. This thesis suggests an expansion to this protocol. The expansion is about device profiles, meaning that a device (e.g., a wrist prosthesis, elbow prosthesis, sensory or the like) connecting to an electrical prosthesis system will let the system know what functions it can provide, without the need of updating of the system.

A complete functional specification for the NRWD has been written. The specifications are based on those from the originals made by Stavdahl, with revisions in the requirements involving communication. This to make them comply with the UNB-protocol. The protocol is build on a CAN-bus, so in practice this means that all digital communication interfaces except CAN has been removed. In addition there have been added a requirement to prevent the motor from overheating.

During the assignment a complete circuit diagram for a new control system has been developed. The circuitry is ready for testing and construction. It has been emphasized to use small components and making the NRWD hardware compatible with the UNB-protocol. There are also suggested some guidelines for the software development.

Contents

List of Figures

Terms and abbreviations

- NRWD - NTNU Rotary Wrist Device
- DOF - Degree(s) Of Freedom
  In prosthetics DOF is a term that states how many axis a joint can travel about, i.e., the human knee and elbow has one DOF, while a wrist has three.
- EMG - ElectroMyoGraphy
  EMG is a technique for evaluating and recording the electrical activity produced by skeletal muscles.
- OSI - Open System Interconnection
  The OSI model is a way of sub-dividing a System into smaller parts (called layers) from the point of view of communications. A layer is a collection of conceptually similar functions that provide services to the layer above it and receives services from the layer below it. [14]
- CAN - Controller Area Network
  A communication bus widely used in the automotive industry, now headed towards electric prosthetics.
  CAN-terms:
  - RTR - Remote Transmission Request
  - IDE - IDentifier Extension
  - SRR - Substitute Remote Request
  - CRC - Cyclic Redundancy Check
  - ACK - ACKnowledge
  - MOb - Message Object
    During message transmission on the CAN-bus only relevant info of a message is stored in the buffer, error checking parameters are diregarded. This is referred to as a MOb.

- SPI - Serial Peripheral Interface
  A serial interface used by microcontrollers for communication.
- ISP - In-System Programming
  A serial interface used for programming of microcontrollers that are located in a complete circuit.
- PWM - Pulse-Width Modulation
- UNB-protocol
  A protocol for communication developed at University of New Brunswick
- RFI - Radio Frequency Interference
  A definition of disturbance that effects electrical circuits. Also called ElectroMagnetic Interference (EMI).

Functional Requirements abbreviations:

- GEN - General requirements
- WJF - Wrist Joint Function
- WMF - Wrist Motor Function
- WSF - Wrist Servo Function
- WCF - Wrist Communication Function
- WPF - Wrist Power Function
- PAF - Proximal Attachment Function
- DAF - Distal Attachment Function

Chapter 1

Introduction

Replacing an amputated arm with a prosthesis that has the functionality of the human arm have been tried innumerable times. After the wounded soldiers returned from World War II a new industry arose, where scientists and engineers tried to compensate for the loss of limbs. The quality of the results have been varying, and the assignment has seemed somehow impossible using the technology available. The outcome of this is to make simplified prosthesis, specialized for solving specific tasks.

The development of electrical prosthesis is getting more and more advanced and the research break barricades at a great pace. Prosthesis systems gets larger and more complex and communication between different parts of a prosthesis is getting quite common. To make sure the growth is going in the same direction, some standards are necessary. A large group of scientist and engineers are already looking into this and a proposal for a standard protocol for communication is being developed as this assignment is written. The first revision was released in late 2009, and a revised version is expected any time soon. The first part of this assignment will look into one aspect of this protocol and suggest an expansion by adding the use of device profiles.

The second part of this assignment will be about hardware for the NTNU Rotary Wrist Device (NRWD). The wrist has been worked on in several projects and thesis, but still does not behave as desired. In addition the release of the standard protocol demands adjustments. The work done by Håkon Skjelten in 2005[13], and the complete circuit diagram made by Torgrim Gjelsvik in 2006[7] has been the basis for a greater part of the work done in this thesis . This assignment will propose a complete circuit diagram that fulfills the new requirements and removes the previous error.

Chapter 2

Communication

2.1 Background

During the development of new and more advanced electrical prosthesis the need of a standard communication protocol arises. Many buses are available where each one provides different functionality, e.g., different voltage levels, different number of permitted nodes on the bus, and satisfies different needs and therefore manufacturers choose different buses. Even one manufacturer may not have a standard bus they use, but tend to choose the one best fitted for the product being made at the time. A standard choice of bus, and in addition a standard protocol for communication on this bus would make it easier to combine devices from different manufacturers. There is already a group of people that is interested in making this happen, they got their own internet-site[1] and a group at the Google Groups society[2]. At University of New Brunswick it is prepared a proposal for a standard protocol which will be described in Chapter 2.5. Throughout this chapter the previous work on this subject will be discussed and in addition some possible future expansions with the use of device profiles are suggested.

2.2 Available Communication Buses

Many buses and protocols for communication can be used in prosthesis systems. Some of those being CANopen, ProfiBus and DeviceNet, that are all well-developed buses and widely used in other industrial systems. However, the one that seem to be the preferred choice for an increasing number of manufacturers is the CAN-bus. In addition the protocol to be presented is based on the

[1] `http://openprosthetics.wikispot.org/Open_Standards_for_Prosthetics`
[2] `http://groups.google.com/group/scip-forum`

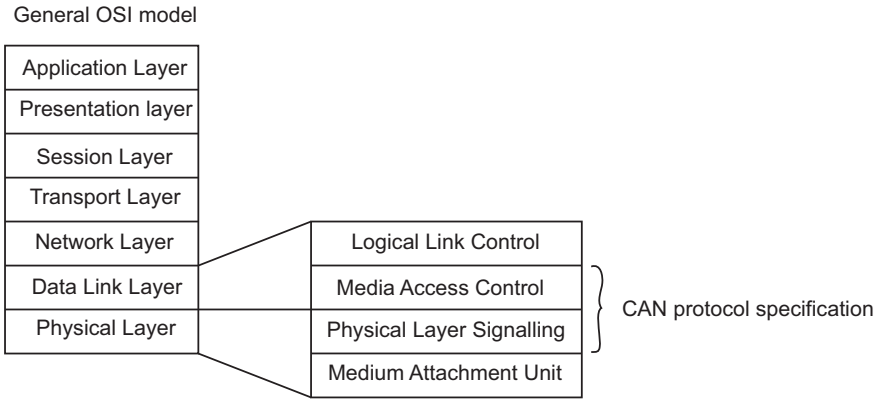


Figure 2.1: The CAN Bus in OSI

CAN-bus, and its versatility and robustness makes it a safe choice, and therefore it will be the bus of choice further through this thesis.

2.3 The CAN Bus

A severe amount of the information in this section is retrieved from Siemens' "Introduction to CAN"[12], and the "CAN Specification" from Bosch[11].

The CAN bus is very versatile and already the preferred choice for many manufacturers. In view of the OSI-model CAN handles the Data Link- and Physical Layer as shown i Fig.2.1. The CAN specification does not include any standard interface for connecting CAN-nodes together, and message handling must be done by application. It is simple in the fact that it communicates by only two wires and transmits high and low bits onto the physical wire (e.g. twisted pair). There are many things that makes the CAN-bus a preferable choice in the manner of electrical prosthesis:

- **Speed**
  It can provide transmissions up to 1Mbit/sec within distances of 40m, which is a distance likely never to be exceeded within prosthesis.

- **Error handling**
  The bus' error detection and confinement make it very reliable. A bus operating at 500kbit/s, 25% bus load, working 2000hr/year will produce one undetected error every 1000 years[12]. A detected error automatically initiates retransmission, and a node that keeps sending error-prone messages will be excluded from the bus.

- **Low cost**
  Low cost CAN bus devices are made in large quantities by many different manufacturers.

- **Well-known**
  The bus is used in many other industries making it well-known and support should be easy to achieve.

2.3.1 Frames

CAN communicates by messages, called frames. Each frame consists of seven fields for indicating the type of data being transmitted, checking for errors, and a field for the data itself. The data field of a message can contain up to eight bytes. CAN does not operate with addresses from one node to another, but all frames are broadcasted onto the bus with an identifier describing what type of data the frame contains and a priority, this way every node that has interest in that data can receive it. In practice all messages are received by all nodes and then ignored if it is of no use. In software, however, the identifier field can be used as a node address.

There are four different main types of frames:

- **Data Frame**
  A standard frame that contains information transmitted from a node. Fig. 2.2 shows a typical Data Frame. This is the "ordinary" frame that is used when one node wants to send information to another node.

- **Remote Frame**
  A frame from a node requesting certain information specified in the message. Looks like a Data Frame, except the RTR bit is set recessive and there is no Data Field in the Remote Frame.

- **Error Frame**
  A frame sent from a node that has detected an error on the bus. Consists of two fields, a Error Flag Field with 6-12 bits according to the type of error, and an Error Delimiter Field which consists of eight recessive bits.

- **Overload Frame**
  Used to provide some extra delay between a Data- or Remote-message. Same format as an active Error Frame, with 12-20bits, where the first six bits is the Overload Flag and shall be dominant, the next 0-6 is for Superimposition of Active Overload Flags which represent Overload Flags set by other nodes. And the last eight are the Overload Delimiter and shall be recessive.

In addition there are two different Data Frame formats which differ the length of the Arbitration Field. What is called a Standard Frame (CAN 2.0A) use 11 bits identifiers, and the Extended Frame (CAN 2.0B) use 29 bits. The CAN 2.0B is added in a revised version and provides some extra identifier bits to relieve a system designer from compromises when defining naming schemes. Both standards can still be used, the 2.0B is not a replacement for the 2.0A.

2.3.2 Dominant and Recessive Bits

The frames transmitted consists of high and low bits. The low bit is called dominant, and the high bit is called recessive. This is because if two nodes tries to transmit a message at the same time, and one of them transmits a low bit and the other a high bit, the signal bus will remain low. This is how the identifier field also implements priority. The transmitting node senses the bus level and will abort message transmission if the bus stays low when it tries to transmit a high bit. This way, messages with the lowest numerical value are prioritized.

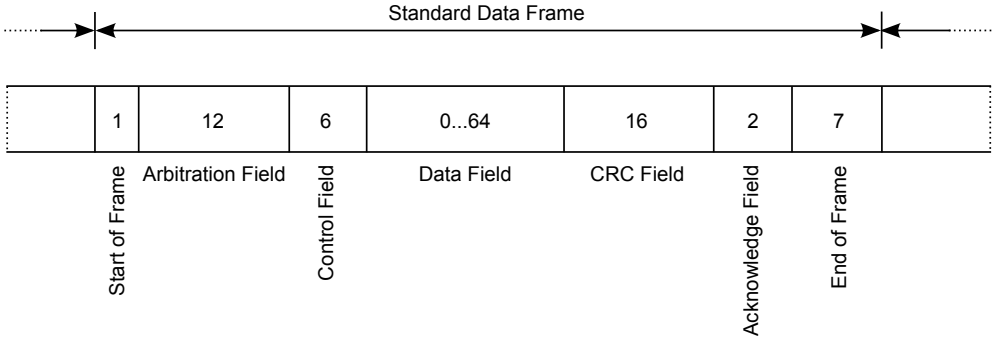


Figure 2.2: A Standard Data Frame

2.3.3 Start of Frame

The first part of a message frame is one single dominant bit as seen in Fig.2.2

2.3.4 Arbitration Field

The Arbitration Field follows the Start Of Frame-bit and contains the identifier which reflects the contents and priority of the frame. It consists of, if the Standard Frame is used, 12 bits, as shown in Fig.2.3. If the Extended Frame is used, the Arbitration Fields consists of 32 bits, as in Fig.2.4. This is where the frames are given their identity, to inform receiving nodes what the message contains, and at the same time their priority. If the Arbitration Field is identical for two separate frames, meaning they have the same priority, the priority-check will continue through to the Control and Data fields. The last bit of this field is the Remote Transmission Request bit (RTR-bit) which denotes if the frame being sent is a Remote or Data frame.

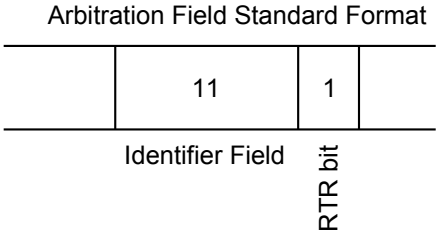


Figure 2.3: The Arbitration Field Standard Format

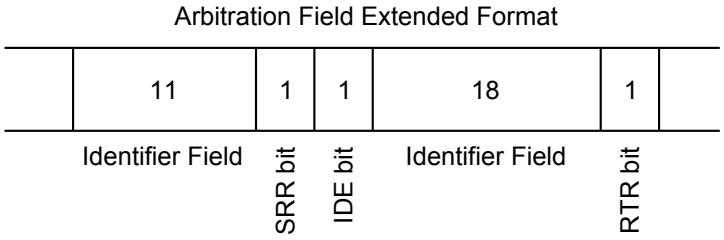


Figure 2.4: The Arbitration Field Extended Format

2.3.5 Control Field

The Control Field contains information about the size of the message and consists of six bits. The first bit is the IDentifier Extension bit(IDE-bit) which denotes if the Standard or Extended Frame is being used. A dominant IDE means a Standard Frame is used. If the Extended Frame is used the RTR-bit is replaced by the Substitute Remote Request bit (SRR-bit) that must be recessive, followed by a dominant IDE to identify that the Extended Frame is used, and then the remaining 18 identifier bits follow. The IDE-bit is then a part of the Arbitration Field. The following bit is reserved and shall be dominant. The four remaining bits denotes a number between 0-8, which represent the number of bytes in the following Data Field, 0 - 8 bytes. The Control Field is shown in Fig.2.5.

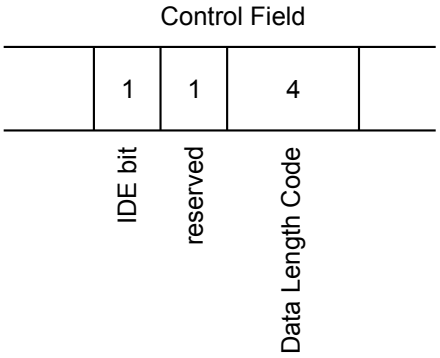


Figure 2.5: The Control Field

2.3.6 Data Field

The Data Field contains the information the node wants to transmit, and is shown in Fig.2.6. Its length shall be as denoted in the Control Field and no more than 64 bits. The contents and setup for how the Data Field should be is application dependent.

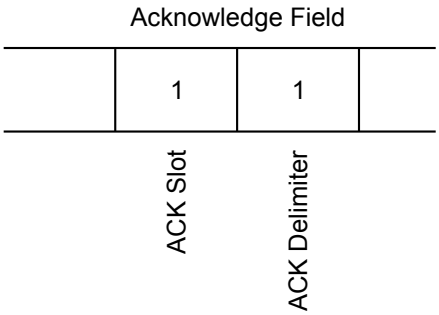


Figure 2.8: The Acknowledge Field

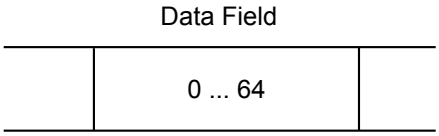


Figure 2.6: The Data Field

2.3.7 CRC Field

See Fig.2.7. The Cyclic Redundancy Check Field consists of 16 bits. 15 bits are used to check for transmission errors. The last bit is the CRC Delimiter bit and shall be recessive.

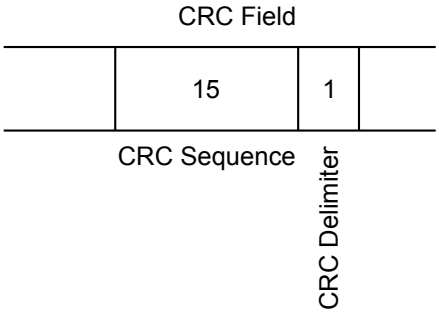


Figure 2.7: The Cyclic Redundacy Check Field

2.3.8 Acknowledge Field

See Fig.2.8. The next two bits make the Acknowledge Field. Any node that receives an error free message returns the same message out on the bus with a dominant bit in the ACK Slot, regardless of who the message was intended for.

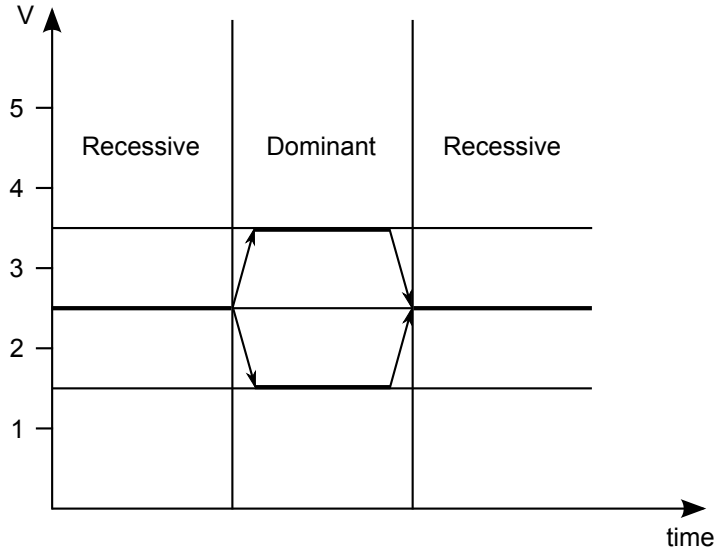


Figure 2.9: The CAN Voltage Levels

2.3.9 End of Frame

The last part of a frame is seven recessive bits that ends the message, End of Frame, like seen in Fig.2.2.

2.3.10 Physical Levels on the Bus

The CAN differential bus is a two-wire bus, which operates with two logical levels, as previously mentioned. A recessive bit on the bus physically means that the voltage level on both wires are approximately 2.5V. A dominant bit brings the CAN Low (CANL) to 0.5V - 2.25V and CAN High (CANH) to 2.75V - 4.5V. This will exclude any common mode voltages that could pollute the signal. See Fig.2.9.

2.4 Device Profiles

In order to make it easy to connect different types of equipment to a prosthesis without the need of advanced updating of the system (i.e. manually installing drivers), a requirement for a standard that handles exactly this arises. One solution is to make use of device profiles. Like a storage unit or mouse connects to a computer through USB and lets the computer know what functions are available, a wrist can be connected to an existing prosthesis-system and let the rest of the system know that the newly connected device is a wrist and what functions the wrist can provide.

A requirement for using device profiles is that the nodes connecting are using the same protocol for communication. When connecting a new node there will usually be a master node on the bus. The master node and new nodes connecting will have to be able to transmit and receive, and understand each others messages. Then the setup for a new profile can be a part of the master node accepting a new node on the bus. A specific example will be presented in Section 2.6, where device profiles has been added to the UNB-protocol. All the node-profiles in the system will origin from a generic parent profile. Meaning that all of the profiles will have the same basic functionality. Each node will in addition have certain node-specific abilities, according to what type of device it belongs to. A wrist with one actuator, for example, would have the ability to communicate with all other nodes on the bus according to the parent profile. In addition it would have the ability to receive setpoints. Since the wrist shared its profile on initialization, the master node would expect the wrist to tolerate setpoints. A seen in Fig.2.10 a wrist with one rotational axis (1DOF) will inherit the functionality of all the above profiles. It will have some functionality because it has one degree of freedom and some functionality because it's a wrist. The wrist functionality will be the same no matter if it's a 1-, 2- or 3DOF wrist. In addition it will inherit some functionality because it's an actuator; these abilities will be shared amongst all actuators. In this setting an actuator will be any prosthesis joint with the possibility of controlled movement. The hierarchy could continue further downwards adding different specifications on functionality.

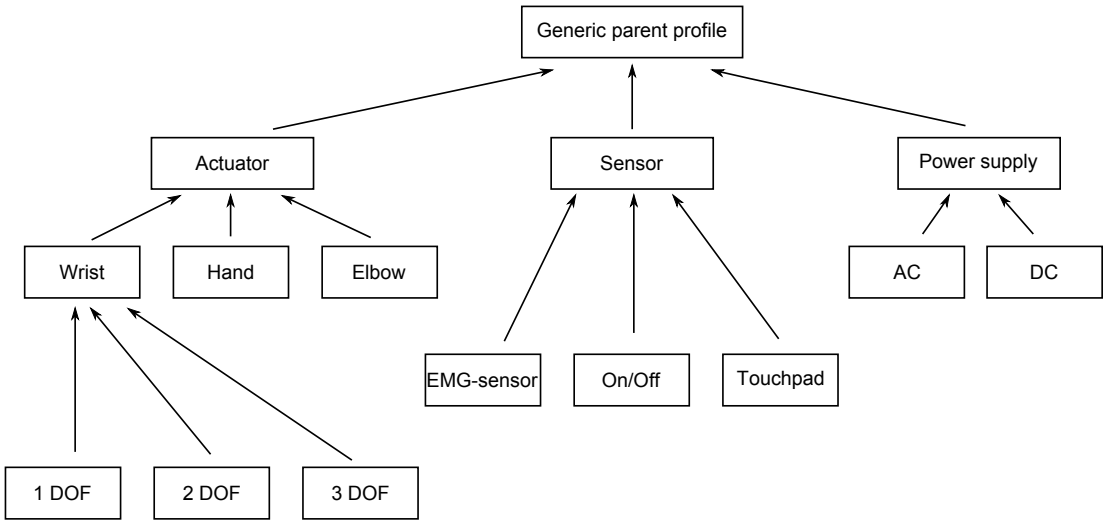


Figure 2.10: A Device Profile Hierarchy

Like it is shown in Fig.2.11 the CAN-bus, and the other devices connected to it, has no knowledge (or use) of the signals used to make the wrist work internally, and will only communicate with the CAN-controller and handle predefined messages involving for example a new setpoint for an actuator or the updated value from a sensor. In the same way equipment for development and analysis can also easily be connected. The clever thing about device profiles is the fact that no information about manufacturer or brand is needed, because all the information about functionality is given through the profile. As long as they communicate according to the same protocol, any node can connect to any prosthesis system.

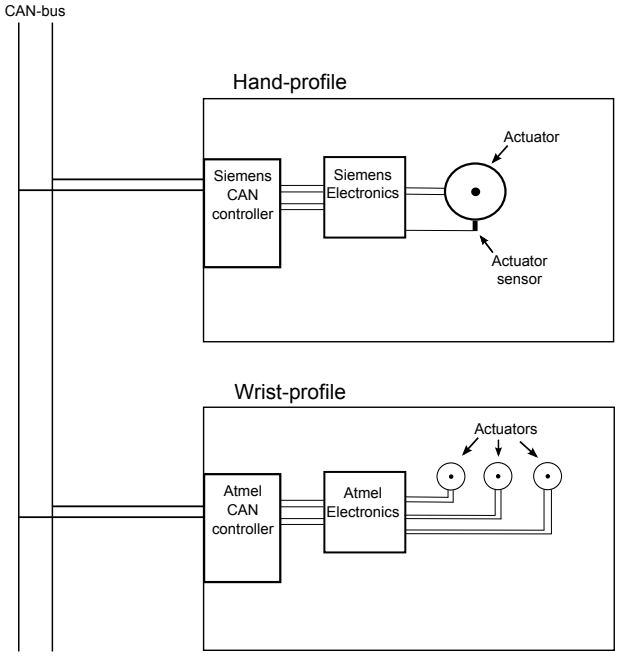


Figure 2.11: Device Profiles Interface

2.5 The UNB Protocol

In September 09, Mr Yves Losier at University of New Brunswick released a document describing the communication protocol they used for their AIF UNB Hand Project[9]. It's presented as a suggestion for a standard for communication between devices in the world of electrical prosthetics. The document is based on the CAN differential bus at a rate of 1Mbps, and the report applies to the communication traffic between a master controller module and any other system module found within the prosthetic limb. It explains the structure of a basic CAN-frame, and suggests how the arbitration- and datafield should be composed. It uses the CAN 2.0A Standard Frame, meaning that the Arbitration Field consists of 11 bits. The entire document is available at the Google Group society[9].

2.5.1 Arbitration Field

The document suggests setup for the Arbitration Field as shown in Fig.2.12. Where the bits b9-b10 is used to assign a priority of the message. Even though four values are available, only three are used. 0 = High priority, 1 = Normal priority, and 2 = Low priority. Bit b8 is the message mode, if the message is intended for the master or a slave node. b8 = 1 for message for master, b8 = 0 for message for slave node. The final b0-b7 is the Node Identifier, the address of the node the message is intended for. Ergo the protocol sets a node-specific address, instead of labeling the message with it's contents. If the message is sent in mode 0, message to the master, the Node Identifier is the address of the transceiving node.

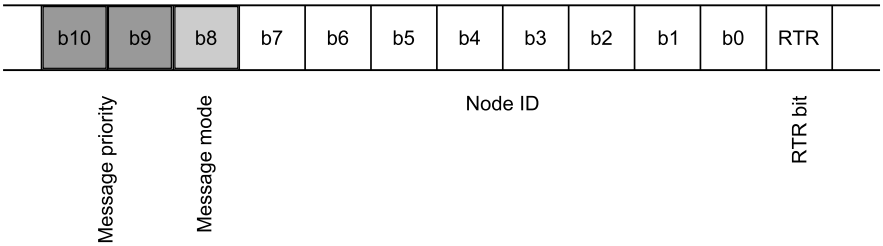


Figure 2.12: The Arbitration Field in UNB-protocol

2.5.2 Data Field

The protocol suggests a setup for messages like shown in Tab.2.1. The first byte, of the eight possible, will be the function code. The remaining bytes contains function-specific parameters. Each function is specified below.

Table 2.1: Data Field as proposed in the UNB protocol

Function Code	**Function Code Description**	**Sender**	**Recepient**
0x00	ResetDevice	Master	Any Node
0x01	BindRequest	Any Node	Master
0x02	GetDeviceInfo	Master	Any Node
0x03	GetDeviceParameter	Master	Any Node
0x04	SetDeviceParameter	Master	Any Node
0x05	IndGetDeviceParameter	Any Node	Master
0x06	SetNodeID	Master	Any Node
0x07	SuspendDevice	Master	Any Node
0x08	ReleaseDevice	Master	Any Node
0x09-0x7F	Reserved for future commands	N/A	N/A
0x80-0xFF	Module-specific commands	Variable	Variable

0x00 - Reset Device

This function is sent by the master module to reset the specified module. The parameter value, ack, can be set to 0 if no response is desired from the recipient module.

0x01 - BindRequest

This function is sent by a module immediately following power-on or software reset. The master module responds to the request by simply acknowledging the module's presence on the bus.

0x02 - GetDeviceInfo

This function is sent by the master module to request the vendor Id, product Id, firmware version, and serial number of the specified module.

0x03 - Get Device Parameter

This function is sent by the master module to request a parameter value of the specified module.

0x04 - Set Device Parameter

This function is sent by the master module to set a parameter value of the specified module.

0x05 - Indirect Get Device Parameter

This function is sent by the any module to request a parameter value of the specified module indirectly via the master module.

0x06 - Set Node Id

This function is sent by the master module to set the specified module's node Id.

0x07 - Suspend Device

This function is sent by the master module to request that the specified module will cease to transmit messages on the bus for the specified period of time.

0x08 - Release Device

This function is sent by the master module to enable the specified module to transmit messages on the bus.

2.6 Device Profiles in the UNB Protocol

It is desirable to add the functionality of device profiles to the presented UNB protocol. The following section will propose such an expansion. This suggestion involves adding one message to the existing ones. The message is thought of to be part of a "hand-shake" that should occur during initialization of a new connection. That is, the connecting node is given an address and responds with a ProfileInfo message so the master node knows what behavior to expect from that address.

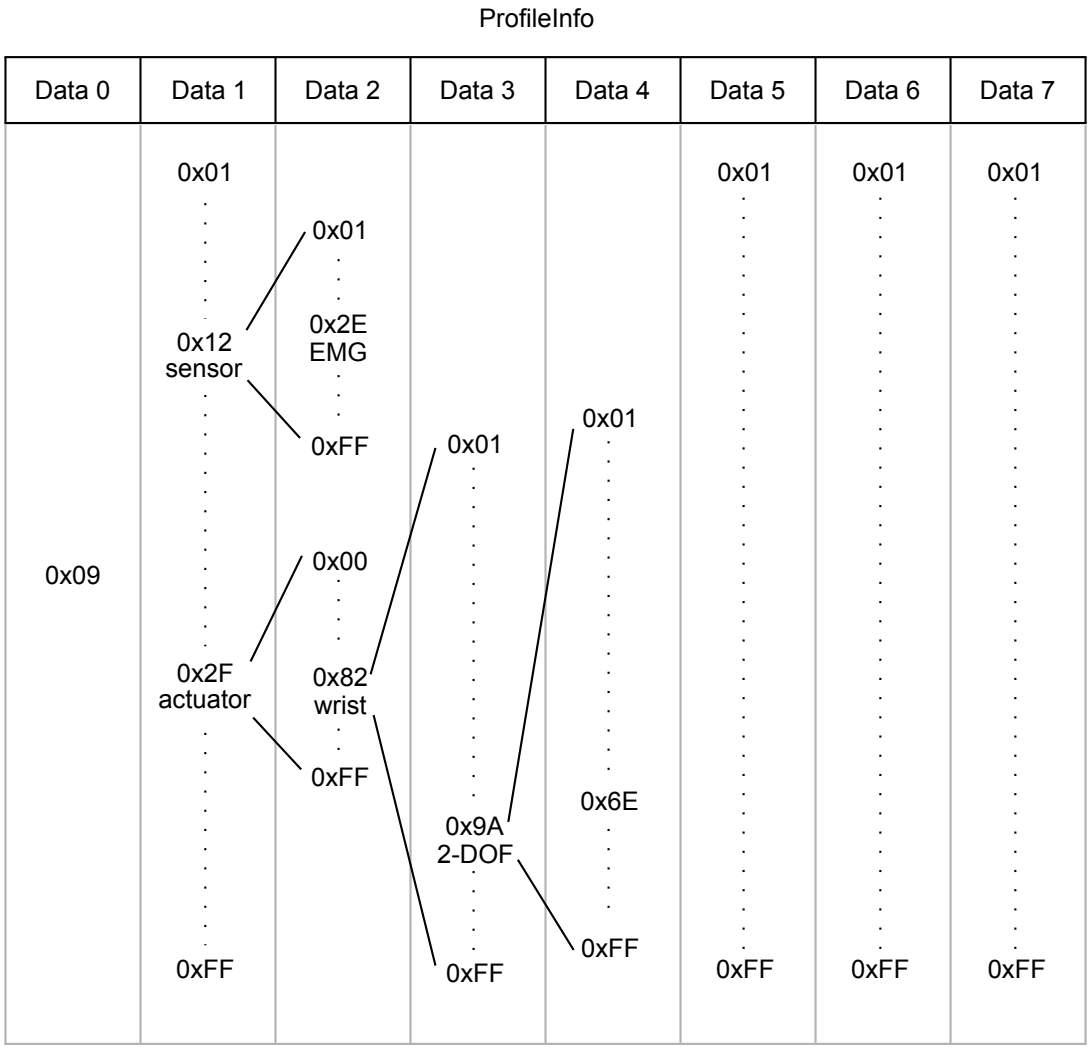


Figure 2.13: The ProfileInfo Message

The new function will have the first available function code in the UNB pro-

tocol, i.e., 0x09. Fig.2.13 shows the Data Field of the new message. The first byte of data is given by the function code. The second byte of data will define the main category of the new node. In that category the node could be one of potentially 256 subcategories, which again could be divided into 256 subcategories. This continues throughout the following three bytes of the message, meaning that these bytes will be dependent of preceding byte. The three bytes at the back end will contain specific functionality. If future revisions come up with different ways to transmit setpoints, this will be the bytes defining which setup to be used with this node. If the nodes handles more than one setup, more bits can easily be set. The last three bits can also be left blank, provided the appropriate message length has been set in the Control Field. This message would be of direct use for a master node, when it simply looks into a table using the data as an index, in order to find functions available from the node that sent the message. The first five bits of the message data should be arranged as shown in Tab.2.2 and the corresponding bits like Tab.2.3. The last three bits should be arranged like in Table 2.4, with bits like in Table 2.5.

Table 2.2: Profile Data, first five bytes

Data 0	Data 1	Data 2	Data 3	Data 5
FunctionCode	Actuator	Wrist	1 DOF	Details1
				Details2
				Details3
			2 DOF	
			3 DOF	
		Hand		
		Elbow		
	Sensor	On/Off		
		EMG		
		TouchPad		
	Power			
	Analysis			

Table 2.3: Profile Bit Specification, ProfileInfoH

Data 0	Data 1	Data 2	Data 3	Data 5
0x09	0x01	0x01	0x01	0x01
				0x02
				0x03
			0x02	
			0x03	
		0x02		
		0x03		
	0x02	0x01		
		0x02		
		0x03		
	0x03			
	0x04			

Table 2.4: Profile Data, last three bytes

Data5		Data6		Data7	
Data 5H	Data 5L	Data 6H	Data 6L	Data 7H	Data 7L
Different Message Setups	Different Operating Voltages	Varying Setpoint Ranges	Various Settings	Various Settings	Various Settings

Table 2.5: Profile Bit Specification, ProfileInfoL

Data5		Data6		Data7	
Data 5H	Data 5L	Data 6H	Data 6L	Data 7H	Data 7L
0x1X	0xX1	0x1X	0xX1	0x1X	0xX1
⋮	⋮	⋮	⋮	⋮	⋮
0xFX	0xXF	0xFX	0xXF	0xFX	0xXF

The Data Field in a ProfileInfo message sent from an actuator wrist with 1 DOF, would then look like Tab.2.6. The Data4-7 has been arbitrarily chosen.

Table 2.6: Message from a actuator wrist, 1DOF

Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7
0x09	0x01	0x01	0x01	0x02	0x1F	0x00	0x00

The Data Field of a ProfilInfo message from an EMG-sensor would look like Tab.2.7. The Data4-7 has been arbitrarily chosen.

Table 2.7: Message from an EMG-sensor

Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7
0x09	0x02	0x01	0x01	0x02	0x1F	0x00	0x00

The proposed functionality could be added to the list of UNB protocol functions:

0x09 - ProfileInfo

This function will let the master module know what functions the sender can provide. The sender would be any newly connected node, and the receiver would be the master node. The length of the message could vary from 5-8bytes dependent on what extra functionality is specified in the last three bytes.

Initialization Example

The initialization of a newly connected wrist module could then look like the figures 2.14 through 2.16. The figures shows a wrist module connecting to an already initialized system with a master module and one electrode. After the hand-shake the master node will have an overview over the functions associated with the address given to the wrist module.

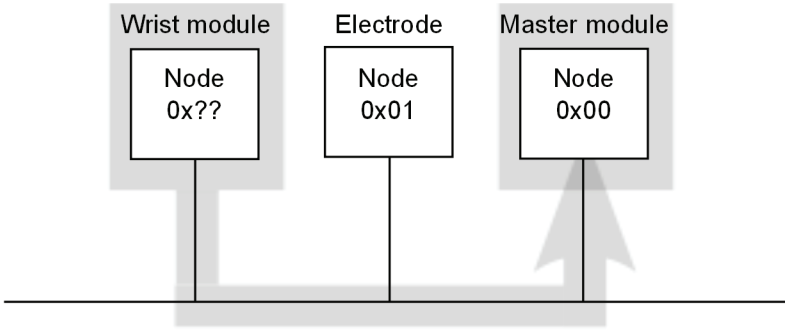


Figure 2.14: BindRequest

The wrist module sends a BindRequest to the master module on the bus immediately after connection.

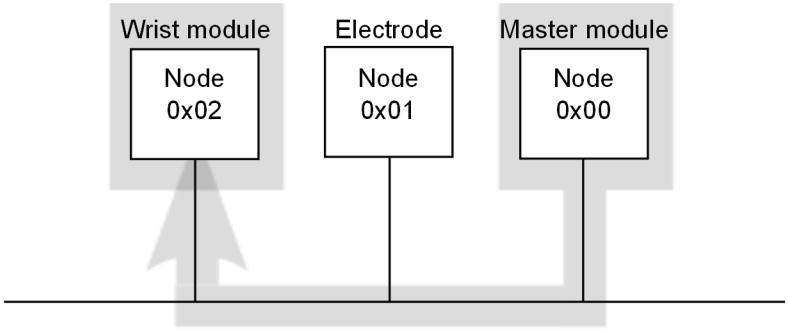


Figure 2.15: SetNodeID

The master module responds, and sets a nodeID for the newly connected module.

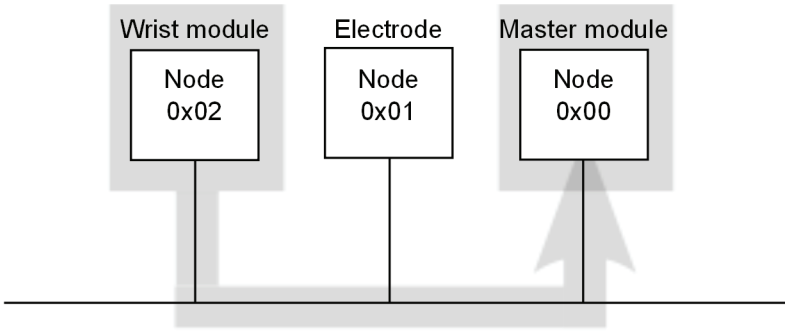


Figure 2.16: ProfileInfo

The wrist module responds to the SetNodeID by sending the ProfileInfo message.

It should be kept in mind that the UNB-protocol has not mentioned any details concerning a handshake on initialization, and the messages used in this example may not comply to a handshake the way they are presented here, but it is likely to believe that some kind of initialization will be necessary and thereby implement the ProfileInfo message as exampled.

The details concerning what profiles that should be included and the specific bits connected to those profiles is due to future development and should be considered by someone with more expertise in prosthetics than the writer of this thesis.

Chapter 3

The NTNU Rotary Wrist Device

3.1 Background

In 2002 Øyvind Stavdahl delivered a PhD thesis where he looked into an optimal rotational axis for wrist prosthesis[15]. The thesis further led to an functional specification for a prosthesis, called the NTNU Rotary Wrist Device (NRWD). Since then the NRWD has been through a long process consisting of several projects of research and development. The latest documented contribution is the complete circuit board design, made by Torgrim Gjelsvik in 2006[7], and production of the circuit board. The work was based on the functional specifications derived by Øyvind Stavdahl, and the consecutive embedded hardware design by Håkon Skjelten[13]. Skjelten suggested the use of one controller for handling external communication, and one controller designated for motor control. This setup did not behave as intended when it was embedded in the complete circuit diagram by Gjelsvik, and a project by Inge Brattbakken in 2009[4] found the communication between the two microcontrollers to be the problem. For communication between the motor- and communication controller the SPI-interface were used. The setup is shown in Fig.3.1. On the motor controller the SPI interface shared the same pins as ISP and debugWire which also had to be available for programming and debugging of the chip. The idea was to be able to program and debug the controller over ISP and then enable SPI to communicate with the main controller. This setup caused the debugging to stall, and the connection to the motor controller was lost.

This thesis will combine all the previous work, derive revised functional specifications and conclude with a solution that eliminates the problem and satisfies the requirements. It will also be emphasized to choose components that can easily be combined on a circuit board that fit into the already developed housing for the wrist. The greater part of the hardware design that has been developed

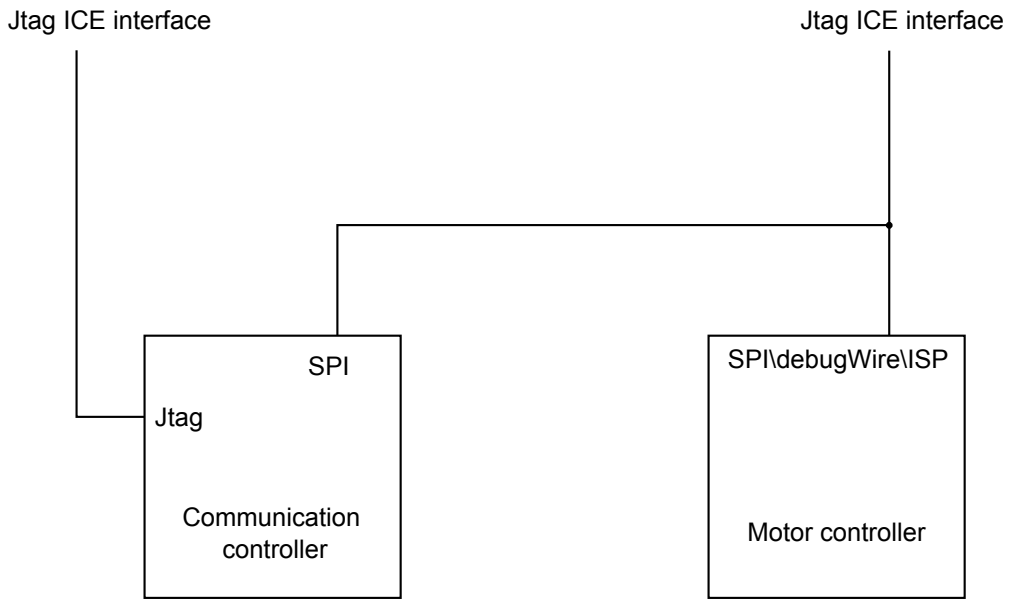


Figure 3.1: Old Communication Setup

through the previous projects is of high quality, and it is desirable to use as much of it as possible. The thesis will however go in depth of what have been done, to get a complete outline.

3.2 Revised Functional Specifications

The progress of developing the NRWD demands some modifications in the requirements. New approaches for external communication have arisen and some of the old requirements have been made superfluous. The biggest change is the creation of the UNB-protocol, and the decision to make the NRWD compliant with it. The protocol makes several of the previous interfaces for external communication unnecessary. The revisions are made from experience gained from the work of Håkon Skjelten[13] in 2005, Torgim Gjelsvik[7] in 2006 and Inge Brattbakken[4] in 2009. The complete revised functional specifications can be found in Appendix A. The following sections will state and explain the revised requirements.

3.2.1 Configurations

Previous requirements stated that the NRWD should handle four different configurations, involving both digital and analog interfaces. The creation of the UNB-protocol implies that future communication in electric prosthesis will be all digital and over the CAN-bus. However, it is still desirable to keep the possibility of using the wrist in a setting with an EMG-sensor without the need of a CAN-controller in the sensor, and therefore one proximal analog interface is kept. This leads to the removal of two of the configurations. The NRWD are now expected to be used in two possible settings:

- Both distal and proximal electronics connected through a digital interface i.e., through a CAN-interface.
- Distal electronics digitally connected, and proximal electronics connected through an analog interface.

The configurations removed from the specifications are:

- Distal electronics connected through an analog interface, and the proximal electronics connected through a digital interface.
- Both distal and proximal electronics connected through an analog interface.

3.2.2 Wrist Motor Function, WMF

- Added requirement for motor protection. WMF-04 "The motor **shall** be protected from overheating. The protection **should** be implemented by hardware."

3.2.3 Wrist Communication Function, WCF

- Changed:
  - WCF-02 "*The PCI **shall** be configurable so that it implements two (0 V, 7.2 V) analog input lines or a bidirectional two-wire CAN interface with a protocol TBD*" has been revised to demand that the proximal CAN-communication uses the UNB-protocol.
  - WCF-03 "*The DCI **shall** be configurable so that it implements two (0 V, 7.2 V) analog output lines or a bidirectional two-wire CAN interface with a protocol TBD*" has been revised to demand that the distal communication will be merely digital and according to the UNB-protocol.
- Removed:
  - WCF-04 "*The WCF shall be configurable to an all-analog mode, with the PCI as an analog input interface and the DCI as an analog output interface*" has been removed.
  - WCF-06 "*The WCF should be configurable to a hybrid mode in which the PCI acts as a bidirectional CAN interface while DCI acts as two analog output lines (cf. WCF-02)*" has been removed.
  - WCF-08 "*The WCF should include an I2C interface, with a protocol TBD, that can be connected to the PCI and DCI in the same way as the CAN interface described in the previous requirements*" has been removed.
  - WCF-09 "*The protocols developed for CAN (and possibly I2C) communication should comply with the outlines given in [3]*" is removed. The protocol is completely defined by the UNB-protocol.

3.3 Specification Analysis

The functional specifications are presented in Appendix A. The specification is based on the one made by Øyvind Stavdahl, but with the modifications outlined in Section 3.2. The requirements will be briefly analysed in the following sections.

3.3.1 Wrist Joint Function, WJF

The WJF is mainly about mechanical requirements, and will not be highly considered throughout this project. It still need to be kept in mind that WJF-03 requires a rotation of at least 180 degrees, and suggests the possibility of unlimited excursion.

3.3.2 Wrist Motor Function, WMF

WMF-01 requires a maximum angular velocity of at least 81deg/s on the output of the gear train, with no upper limit. WMF-02 requires a maximum torque of at least 34,3mNm. WMF-04 demands protection of the motor from overheating, suggesting it should be implemented by hardware.

3.3.3 Wrist Servo Function, WSF

The hardware requirements on the servo functions demands three methods for control:

- On/Off-mode. This could be easily obtained by applying full supply voltage to the motor, and requires only a transistor bridge.
- Position mode. The wrist should be controllable to a specific position. This can be solved by the position sensor mentioned in WSF-01-02-01: a position sensor with a resolution no less than 10 bits per revolution.
- Velocity-mode. The wrists angular velocity shall be controllable to a desired setpoint. This can be solved by the velocity sensor or estimator mentioned in WSF-01-03-01. The velocity measurement is given through the hall-elements mounted in the brushless DC-motor.

3.3.4 Wrist Communication Function, WCF

The WCF requires the wrist to have a two-wire communication interface both proximal and distal to the NRWD. The proximal interface shall handle analog

inputs between 0 and 7,2V, but also handle a CAN bus according to the UNB-protocol[9]. This means that the wrist needs an AD-converter, and a CAN-controller. The wrist shall also include a serial interface for programming and debugging of the controller.

3.3.5 Wrist Power Function, WPF

According to WPF-02 the NRWD shall run normally with supply voltages between 6V and 12V, and take no damage or behave unpredictably of voltages between 0V and 18V.

3.3.6 Proximal and Distal Attachment Function, PAF and DAF

The requirements regarding DAF and PAF are strictly mechanical and will not be considered during this assignment.

3.4 Architecture

The overall architecture for the NRWD is shown in Fig.3.2. The NRWD will receive power and input signals from the proximal hardware, and will use the inputs received to control a motor to a desired setpoint. The CAN bus will be forwarded to any distal hardware.

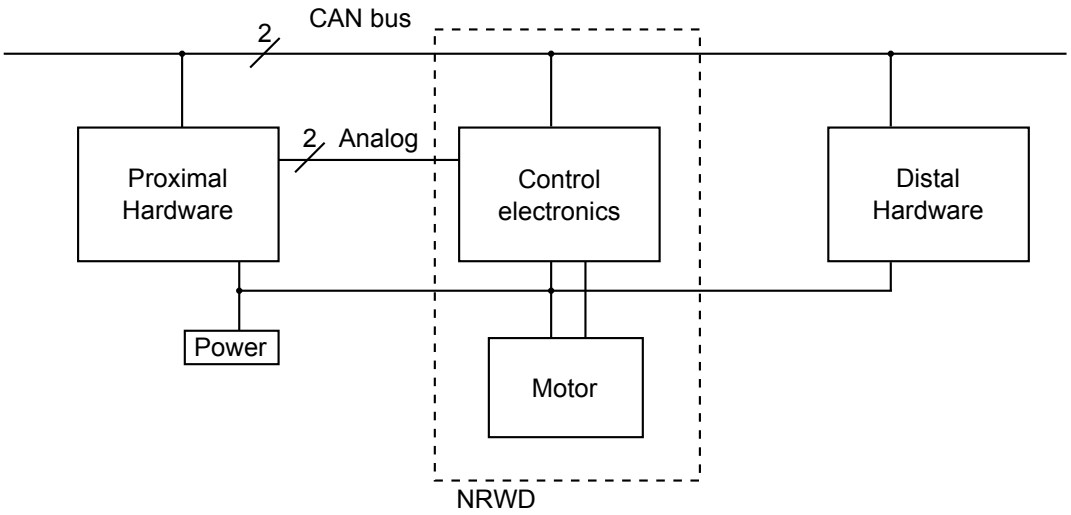


Figure 3.2: High-level Architecture

It's convenient to divide the total system into several subsystems. The overall system will include:

- Data Processing Unit
- Communication
- Motor
- Voltage Regulation
- Programming Interface
- Positioning
- Motor Protection

In practice the first three systems blend together to a great extent. A data processing unit in the wrist is necessary, but its function will be to handle communication and motor control.

The following sections will look into each subsystem.

3.4.1 Data Processing Unit

The former solution used one controller for communication and one controller designated for motor control, with communication between the two over SPI. The solution did, as mentioned, not function as intended, and therefore a new design is needed. Several strategies for control can be chosen:

- **Original communication controller and new motor controller**
  Keep the existing communication controller and choose a new motor controller is a possible solution. However, the possibility of an communication error between the two controllers will still exist. One would also have to find a controller that match the original controller both in size and performance.

- **Same controllers with different approach for internal communication**
  Not using SPI, but another protocol for communication. The controller used for motor control have no other interfaces for communication. A solution could be "Bit-Banging", i.e., creating an interface by software using ordinary I/O-pins. This is interesting, but demands a great deal of software development, and could generate more error-prone code. The possibility of an error in the communication will still exist.

- **One controller handling both communication and motor control**
  Without an internal communication, there can be no communication errors. This approach will add greater requirements to the controller, than what has been before. The controller will have to handle motor control, and in addition handle external communication and calculate setpoints. This solution will occupy less space on the circuit board and one less controller means less power consumed (although the percentage of energy consumed by the microcontroller is relatively small in comparison with the motor).

Because the problem with the previous circuitry was the communication between the two controllers, it would be interesting to try using only one controller and let it handle all data processing aspects of the wrist. This solution would save some space on the circuit board, but will result in some strict requirements to the controller. It will have to respond to every interrupt given by the Hall-elements in the motor and reply with the next commutation-pattern immediately. In addition it has to read and respond to incoming messages on the CAN-bus, or sample the analog input-level, and generate new setpoints. All of the assignments are critical and none of them can be ignored in any manner, but they can be given a priority. The interrupts from the motor must be handled first, and no interrupt can under any circumstances be lost. Next the CAN messages or analog input-level have to be handled, and new setpoints calculated.

It is desirable to use one of the controllers from Atmel for these tasks. The Atmel controllers are of high quality and the programming interface is known. A controller that fits all the mentioned requirements is the AT90CAN128. It includes the desirable CAN-controller, timers with PWM for motorcontrol, ADC for the analog inputs and several serial interfaces available for programming and debugging. It is also available in a small package (QFN), in order to minimize the area used on the circuit board.

The most time-consuming assignment is doubtlessly the interrupts from the Hall-elements. The motor runs at 100 000RPM/min at the most, and the Hall sensors invokes an interrupt every time their logical level change. There are three Hall-elements with 60° between each, meaning there will come an interrupt six times per revolution.

$$\frac{100000 \text{ RPM}}{60 \text{ sec}} * 6 \text{ Hall-elements} = 10000 \text{ interrupts/sec} = 10\text{kHz} \tag{3.1}$$

Eq.3.1 shows that the motor will interrupt at a frequency of 10kHz. From one interrupt to the next, the controller will have to respond with the next commutation pattern, and in addition handle any possible messages in the MOb-buffer. This generates great requirements for the code. An application note from Atmel on controlling a brushless DC-motor by the use of Hall-elements and a AT90CAN128 controller[1], states that the interrupt will be responded to in less than $3\mu s$. This results in $97\mu s$ left for the controller to handle other tasks. This is pure theoretically calculated time limits and the real timing will most likely differ. The limits are however calculated at the upper bounds of what the motor can perform, and it needs up to 30V to achieve the speeds used for calculation. The angular velocity obtainable with the 12V that are given as a max-limit in the specifications for the NRWD, is at the most 40000RPM. This gives an interrupt frequency of 4kHz, and $250\mu s$ between each interrupt. Meaning the controller will have $250\mu s - 3\mu s = 247\mu s$ left for other tasks. Atmel states that the AT90CAN128 performs up to 8MIPS at 8MHz, equal to 8 instructions per microsecond, almost 2000 instructions between each interrupt. This indicates that this approach is possible.

3.4.2 Communication

Communication Interfaces

From the specification the NRWD shall be implemented with a CAN-interface and an analog input-interface. The CAN-interface implies that the NRWD must include a CAN-controller and a CAN-transceiver. The AT90CAN128 includes, as mentioned, a CAN-controller. Any CAN-transceiver that can be supplied by

the voltage-levels specified can in principle be used, but it is desirable to choose one with a limited size. The PCA82C250 from NXP is available in SOIC casing and is chosen for this assignment. The CAN-transceiver will handle the physical aspect of message transfer, while the CAN-controller handles the data contained in the messages.

The analog input is expected to come from an EMG-sensor. The signal from EMG-sensors is stated in the specifications to have a bandwidth of at least 500Hz. The circuitry made by Gjelsvik includes low-pass filtering of the signal, with a cut-off frequency at 160Hz. This is significantly below the specifications, but is most likely a better choice since it would make it easier for the user to maintain a stable control-level. The signal is then converted with the internal ADC in the AT90CAN128. This design will be kept like proposed by Gjelsvik.

Communication Control

The full CAN controller combined with the CAN transceiver offers complete CAN message handling. If a message on the bus matches the identifier programmed in the controller a MOb(Message Object) will be stored in the CAN Message Buffer. The controller have room for 15 MOb's. A MOb includes the most vital information from a CAN message, including a Identifier Tag, Identifier Mask, 8 byte Data Buffer and a time stamp.

The CAN controller can be programmed to handle several actions without interacting with other tasks: what messages to store is decided in advance, the controller can be programmed to auto-respond to certain messages, and the main control will only be interrupted when necessary. The controller can interrupt on the following actions:

- Receive completed OK
- Transmit completed OK
- Error
- Frame buffer full
- "Bus OFF"-setting
- Overrun of CAN timer

The most relevant interrupt for the NRWD is the "Receive completed OK". Because the controller can then focus on motor control until a relevant message is completely received and stored in the data buffer. The "Bus-OFF"-interrupt could also be an indicator for going into sleep-mode, since no setpoints will be

available. It should be emphasized to utilize as much of the integrated CAN-message control as possible, reducing CPU-load for the controller.

The analog input will be available through an AD-converter, and should be read with an appropriate frequency. An appropriate sampling frequency could be twice the cut-off frequency of the low-pass filter. The measured signal will be interpreted directly as a setpoint.

There should also be a possibility for changing what input method to be used. In the previous circuitry this was solved by leaving open soldering pads for zero-ohm resistors. A convenient method is to make use of jumpers. This makes it easy to change the sensor-input method, and it can be done without having to disassemble the entire wrist.

3.4.3 Motor

The motor chosen for the NRWD is the 0620K012B by Faulhaber. The motor is a delta-coupled three-phase brushless DC-motor. The fact that it is delta coupled, as opposed to wye, makes it utilize speed over momentum. A sketch of the motor is shown in Fig.3.3. A three-phase brushless DC-motor has a static magnetic rotor and three electromagnets in the stator, as seen in the figure. The electromagnets are commutated by a motorcontroller to make the rotor rotate. The motor has three internal Hall-elements, which changes their logical level when magnetized by the rotor. The Hall-elements are supplied by 5V and the pattern that appears by reading the three pins simultaneously is an indicator of the position of the rotor. The six possible Hall-element patterns for one clockwise rotation are shown i Tab.3.1. During rotation the Hall-elements will be used as interrupt for the motor controller. A change in logical level will invoke the interrupt, the controller then reads all three pins and sets the output pins according to the commutation scheme.

Table 3.1: Hall-element pattern

A	**B**	**C**	Comments
1	0	0	The only Hall element magnetized is A
1	1	0	Element B is covered by the rotors magnetic field
0	1	0	Element A is outside the magnetic field
0	1	1	C is included
0	0	1	B left out
1	0	1	A included
1	0	0	The rotor is back at the initial posititon

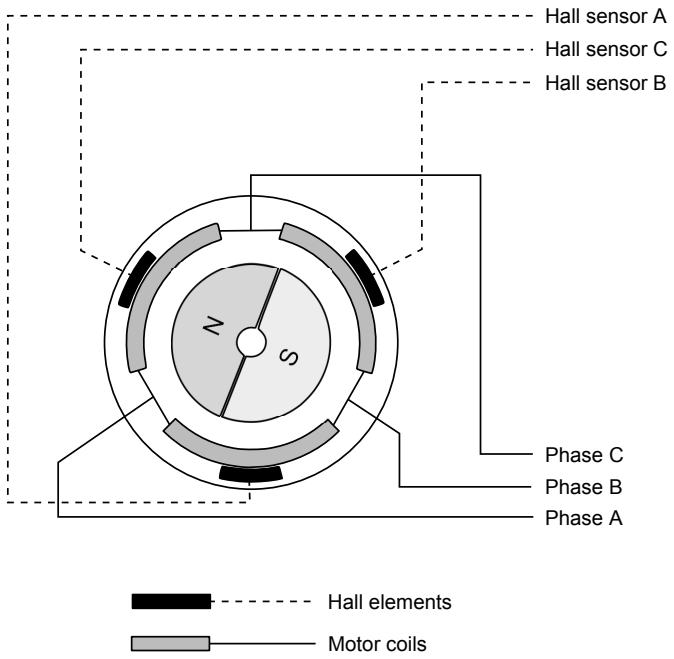


Figure 3.3: Physical Structure of the Motor

Using the motor with a gear ratio between 100:1 and 2800:1 will fulfill the speed- and momentum requirements, as one can see in Fig.3.4. The highest gear ratio will also be limited by the size of the wrist housing. The graphs are calculated with a 12V supply voltage.

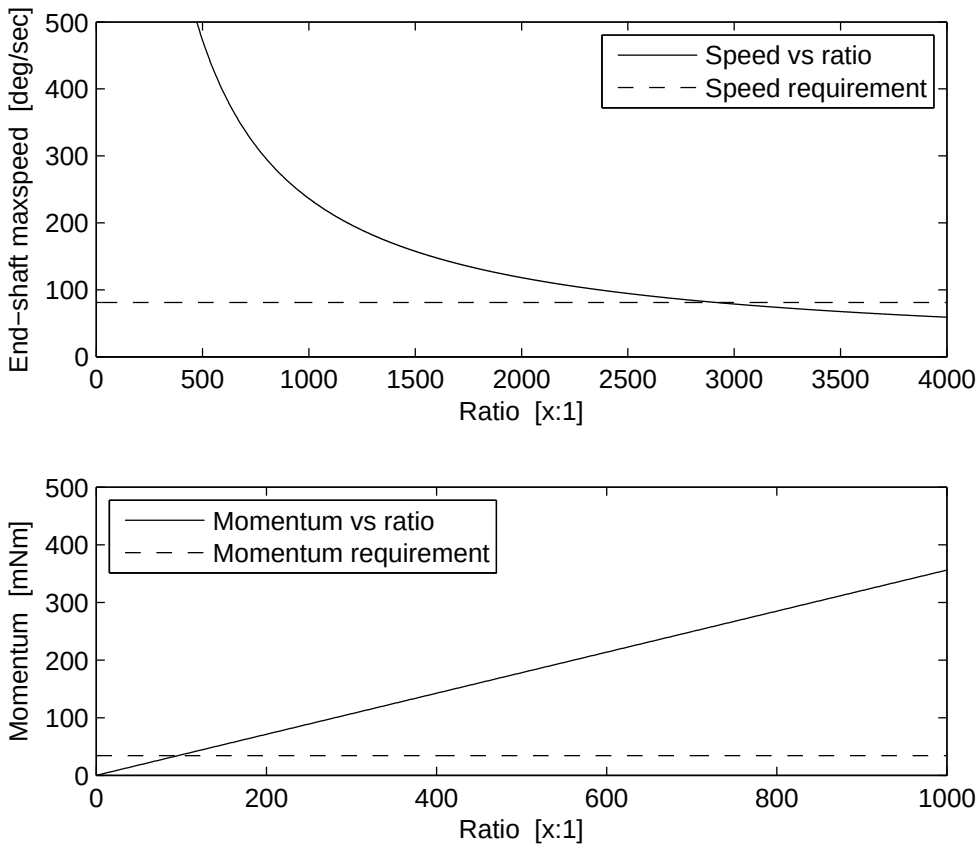


Figure 3.4: Gear Ratio

Motor Driver

The control signal is made by the motor controller and a motor driver is necessary to be able to apply full voltage to the motor coils. The driver consists of six transistors combined in a special manner, called a H-bridge. The most commonly known H-bridge is shown in Fig.3.5 (the controller in the figure is just for illustration and is not the one used later). The output of each transistor is connected to one of three phases in the motor. The easiest way to create a H-bridge is by the use MOSFET transistors. Another alternative is using bipolar transistors, but this is more seldom used. For the mentioned H-bridge, six signals are generated by the motor controller. One high- and one low-level signal for each phase, meaning that if a phase according to the commutation scheme should be low, a logical

"1" with be set at the low-side transistor, opening to ground. And likewise, a logical "1" on the high-side transistor will route 12V to the motor. However, a different solution for the driver circuitry exists, and Gjelsvik introduced this to the NRWD. By using the EL7202 high-speed MOSFET Drivers by Intersil, and only one signal for each phase one gets a very area-effective motordriver. The setup is shown in Fig.3.6. The latter alternative demands the possibility of setting the output from the motorcontroller in a high-impedance state to prevent opening to any voltage potential.

A problem when using brushless DC-motors is the counter-electromotive force, back-EMF. The magnetic field in the motor induces current when the motor is rotating. This results in voltages pushing in the opposite direction of what is produced by the motor controller. This problem is solved by installing some fast flyback-diodes on the motorphases, leading the current to ground.

Motor Control

The motor will be controlled by the AT90CAN128. The Hall-elements rises an interrupt on any 60° rotation, and the controller will respond by setting the output pins according to the commutation scheme. In addition the signal must be Pulse-Width Modulated (PWM) to be able to control the speed. PWM is used in order to control the speed and at the same time maintain full momentum. The motor could have been controlled by adjusting the voltage directly, but in cost of momentum, and at low speeds the motor could easily stall. By applying PWM the supply voltage is switched on and off, and it is the average voltage of one period that controls the speed. Combined with the motor driver described above, the output pins needs to be either PWM'ed, "0", or high-impedance. The high-impedance state could be imposed by changing the data direction register, i.e., setting the port as input. The commutation scheme for respectively clockwise and counter-clockwise are shown in Tab.3.2 and 3.3, where "P" means PWM, "0" means that the signal should be zero, and "X" means the pin connected to that phase should be high-impedance.

3.4.4 Voltage Regulation

The motor chosen for the NRWD needs 5V for the Hall-elements. Its therefore convenient to use 5V as a supply voltage for the rest of the electronics. The regulator shall be able to regulate voltages in the interval 6-12V. It is also desirable to make the regulator circuitry as small as possible.

The switch-mode regulation used by Skjelten generated too much ripple voltage and he recommended replacing this with a different solution. Gjelsviks circuit uses an LT1765 by Linear Technology, which is also a switch-mode regulator. A

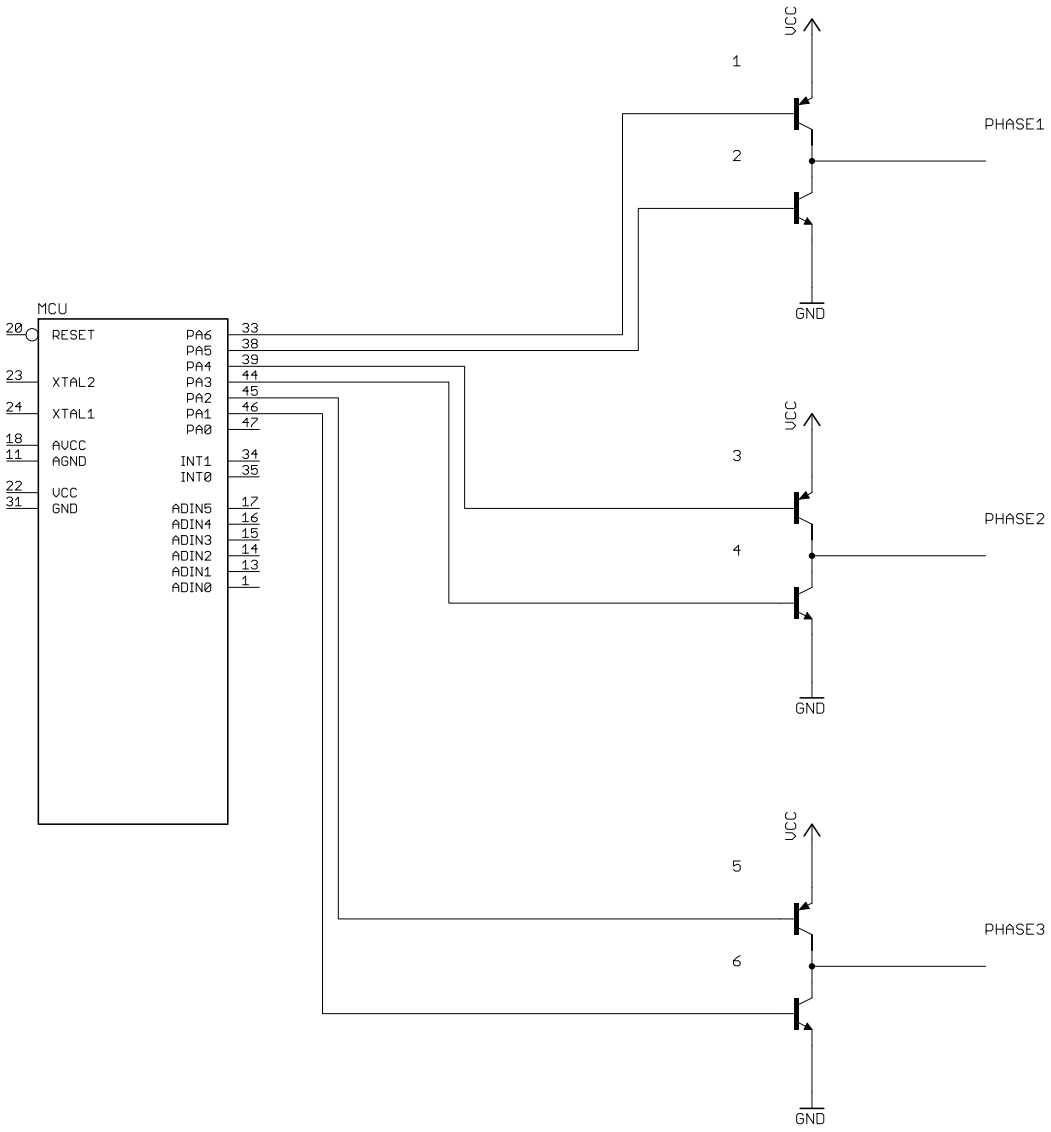


Figure 3.5: 6-signal Motor Driver

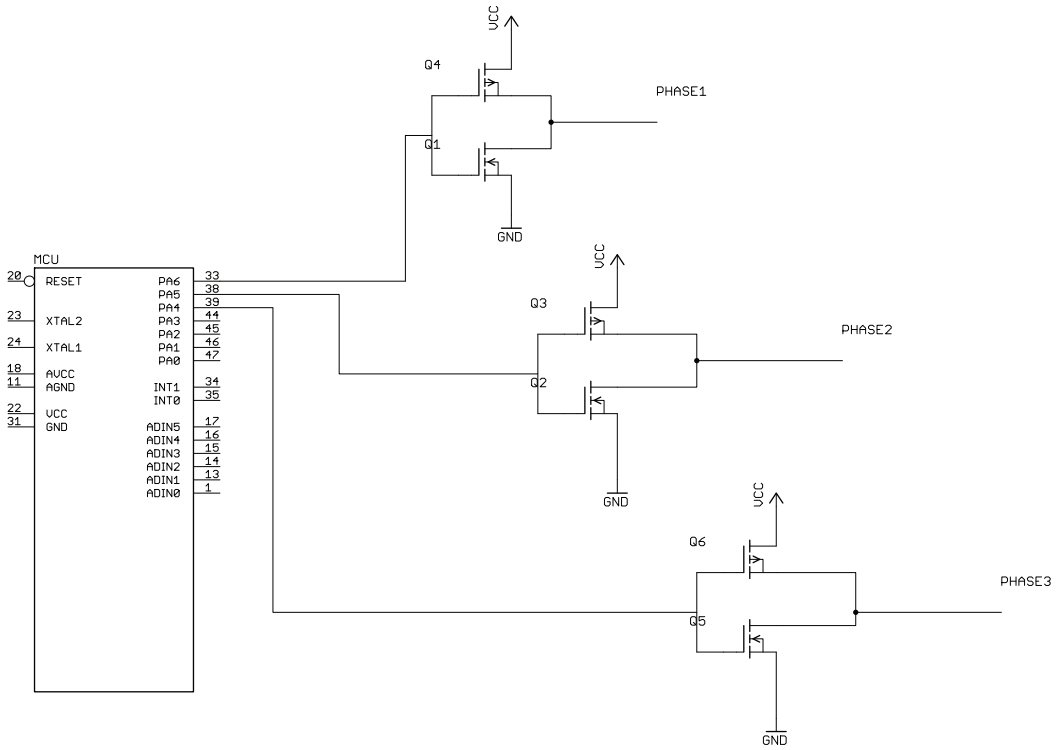


Figure 3.6: 3-signal Motor Driver

Table 3.2: Commutation scheme - Clockwise

Hall			**Phases**		
A	**B**	**C**	**A**	**B**	**C**
0	0	1	0	X	P
0	1	0	P	0	X
0	1	1	X	0	P
1	0	0	X	P	0
1	0	1	0	P	X
1	1	0	P	X	0

Table 3.3: Commutation scheme - Counter-Clockwise

Hall			Phases		
A	**B**	**C**	**A**	**B**	**C**
0	0	1	P	X	0
0	1	0	0	P	X
0	1	1	X	P	0
1	0	0	X	0	P
1	0	1	P	0	X
1	1	0	0	X	P

switch-mode regulator is highly desirable because of its high efficiency. By adding a series of capacitors between the supply voltage and ground the problematic ripple voltage can probably be avoided.

3.4.5 Programming Interface

The circuitry shall include a serial interface to make it possible for external computers to connect for programming and debugging the controller installed in the system. Some of the available interfaces are USB, RS323, fire-wire, or because an Atmel controller is chosen, the JTAG ICE interface. The interface shall consume as little power and physical space on the circuit board as possible. The JTAG-interface used previously has proven to be very efficient and easy to use. Even though this was a part of the problem in the previous system, it will this time be used for only one controller, and it will not share any pins with any other communication interface.

3.4.6 Positioning

The circuitry for exact positioning includes a potentiometer and a magnetic rotary encoder. The encoder produces a PWM-signal and the potentiometer is read through an ADC-port on the AT90CAN128. This solution has not been examined very thoroughly and is kept the way Gjelsvik proposed it[7].

3.4.7 Motor Protection

A way to protect the motor from overheating is by monitoring the current running through the coils. By installing a small resistor between the motor coils and ground, one can measure the current running through the motor. The current running through the motor coils is a direct measure on the energy consumption of the motor. There are several approaches for how to utilize this measurement. The

voltage above the resistor can be amplified by the use of an operational amplifier, it can be connected to a current sensor, or it can be connected to a comparator. Basically, all of these solutions amplify the signal for the further circuitry. This solution makes use of an op-amp, basically resulting in a comparator that outputs a logical level which can be used for signaling. The signal will be connected to the reset on an SR-flip, and the output from the SR-flip will be connected to a transistor that controls the voltage supply to the motor drivers. The flip will then need to be set by the motor controller to enable the motor. This way a shut down caused by too high motor current will not accidentally restart without an approval from the controller.

The current should be limited by the following equation:

$$I_{e,max} = \sqrt{\frac{T_{125} - T_{22} - \frac{\pi}{30000} * n * 0,45 * R_{th,2} * (C_0 + C_v * n)}{R * (1 + \alpha_{22} * (T_{125} - T_{22})) * (R_{th,1} + 0,45 * R_{th,2})}} \tag{3.2}$$

where:

$$\begin{aligned} T_x &= \text{temperature in Kelvin at x degrees celsius [K]} \\ n &= \text{velocity [rpm]} \\ R_{th,1} &= \text{thermal constant from coil to housing [K/W]} \\ R_{th,2} &= \text{thermal constant from housing to ambient [K/W]} \\ C_0 &= \text{static friction [mNm]} \\ C_v &= \text{dynamic friction [mNm/rpm]} \\ R &= \text{terminal resistance}[\Omega] \\ \alpha_{22} &= \text{temperature coefficient} \end{aligned}$$

And with numbers inserted:

$$I_{e,max} = \sqrt{\frac{398 - 295 - \frac{\pi}{30000} * 40000 * 1 * 88 * (0.023 + 1 * 10^{-6} * 40000)}{59 * (1 + 0.004) * (398 - 295) * (14 + 1 * 88)}} \tag{3.3}$$

$$I_{e,max} = 96\text{mA} \tag{3.4}$$

where the following values have been used:

$$
\begin{aligned}
T_{125} &= 398(\text{max permissible coil temperature[5]}) \\
T_{22} &= 295 \\
n &= 40000 \\
R_{th,1} &= 14 \\
R_{th,2} &= 88 \\
C_0 &= 0.023 \\
C_v &= 1.0 * 10^{-6} \\
R &= 59 \\
\alpha_{22} &= 0.004K^{-1}
\end{aligned}
$$

The equation is retrieved from the datasheet[5] and technical information[6] from the manufacturer of the motor chosen for the NRWD. The formula shown in Eq.3.2 is done with "55% reduced", hence the numbers "0.45" are included. Meaning that the surroundings of the motor will contribute to the heat exchange and cooldown of the motor. The motor sited in the wrist is not subject to these reductions and the numerical calculations are therefore done without any reductions. In proportion to its size, the surroundings consists mostly of air, which is known to have poor thermal conductivity. This gives a recommended max current of 96mA.

3.5 Construction

In this chapter the detailed circuit diagrams are presented. The complete diagram is found in Appendix B. The following sections presents the previously discussed subcircuits.

The software used for development of the diagrams is "Eagle 5.8.0 Light" from CadSoft. The program is known from previous works and has the desirable functionality. The free version has some restrictions on the size of the circuit board, but for circuit diagram design this is not a problem.

3.5.1 Communication

The communication circuit is constructed as shown in Fig.3.7. The analog signal is low-pass filtered and connected to the ADC-pins on the AT90CAN128. The low-pass filter has a cut-off frequency at 160Hz. The AT90CAN128 pins in connection with the CAN-controller is coupled to the data transfer pins on the CAN-transceiver. The transceiver can operate in three modes: high-speed, slope control or standby. This applies to the speed of the internal transistors and for the short bus lengths expected the transceiver should be set to slope-control to reduce RFI (Radio Frequency Interference). This is achieved by connecting the RS-pin to ground through a 47kOhm resistor[10].

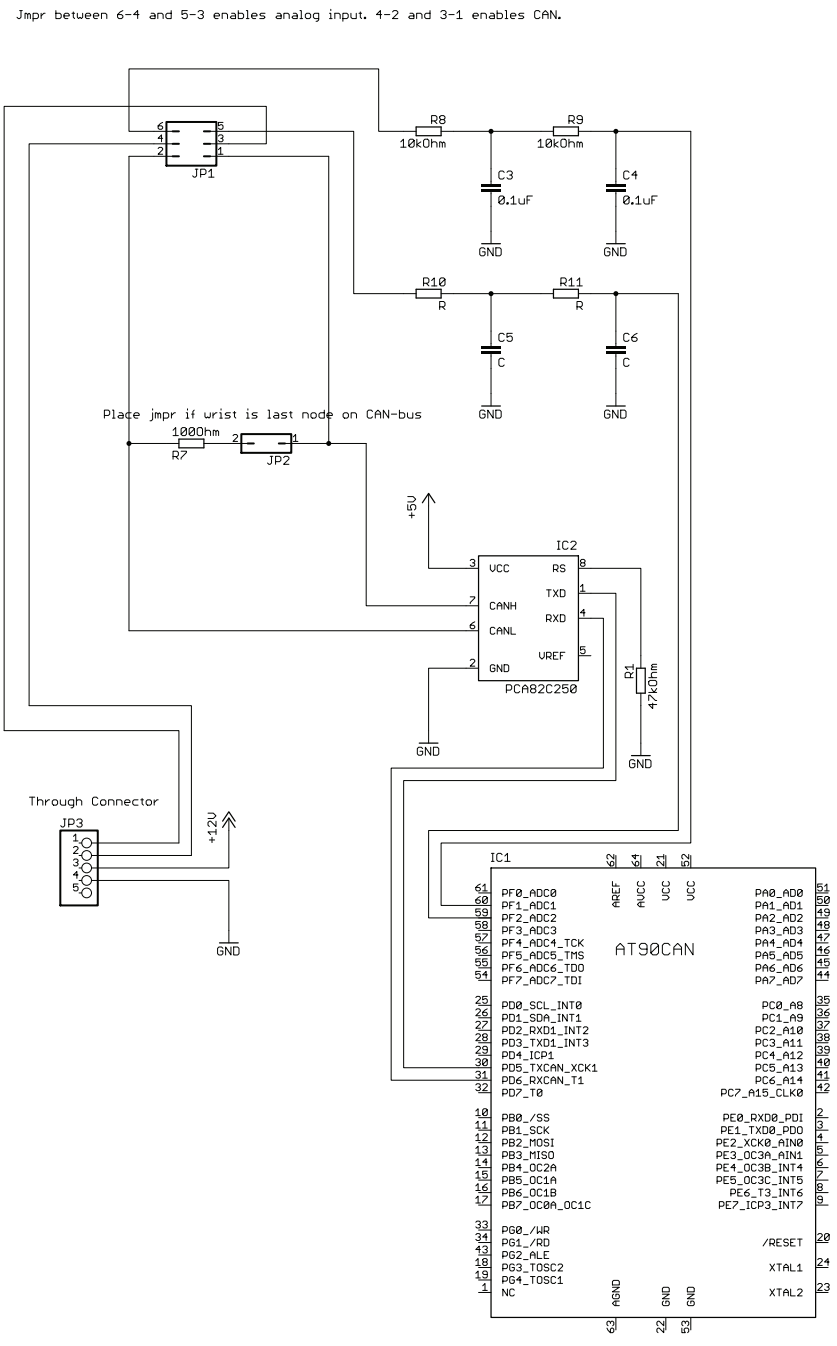


Figure 3.7: Communication Circuit

3.5.2 Motor

The complete motor circuitry is shown in Fig.3.8. The PWM signals from the controller is routed into the MOSFET drivers EL7202[8]. The output is then connected to the Molex connector. The outputs are also connected to the fly-back diodes BAV756S to let off the EMF. The Molex connector fits the cable moulded to the motor. The power supply is controlled through Q1 which is controlled by the motor protection circuitry.

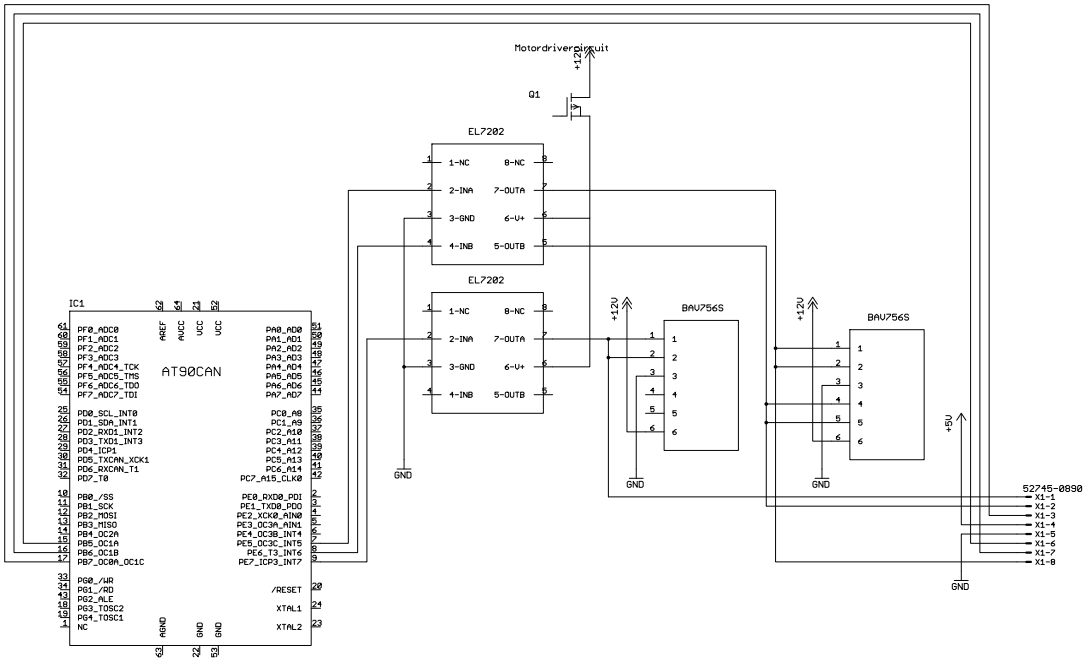


Figure 3.8: Motor Control

3.5.3 Voltage Regulation

The circuitry for voltage regulation is shown in Fig.3.9. The circuitry is like proposed by Gjelsvik, which is also identical with the one suggested in the belonging data sheet[10]. The regulator tolerates different supply voltages, and regulates according to the voltage-level measured at the pin FB. It regulates the value at FB to 1.2V, so by using a voltagedivider, the desired 5V can be achieved. High resistor values for the voltage divider should be chosen to avoid any current-flow. At the bottom of the figure the decoupling capacitors can be seen. These to filter out noise and variations in the supply voltage, to prevent damage and unexpected behavior by the components throughout the complete circuit.

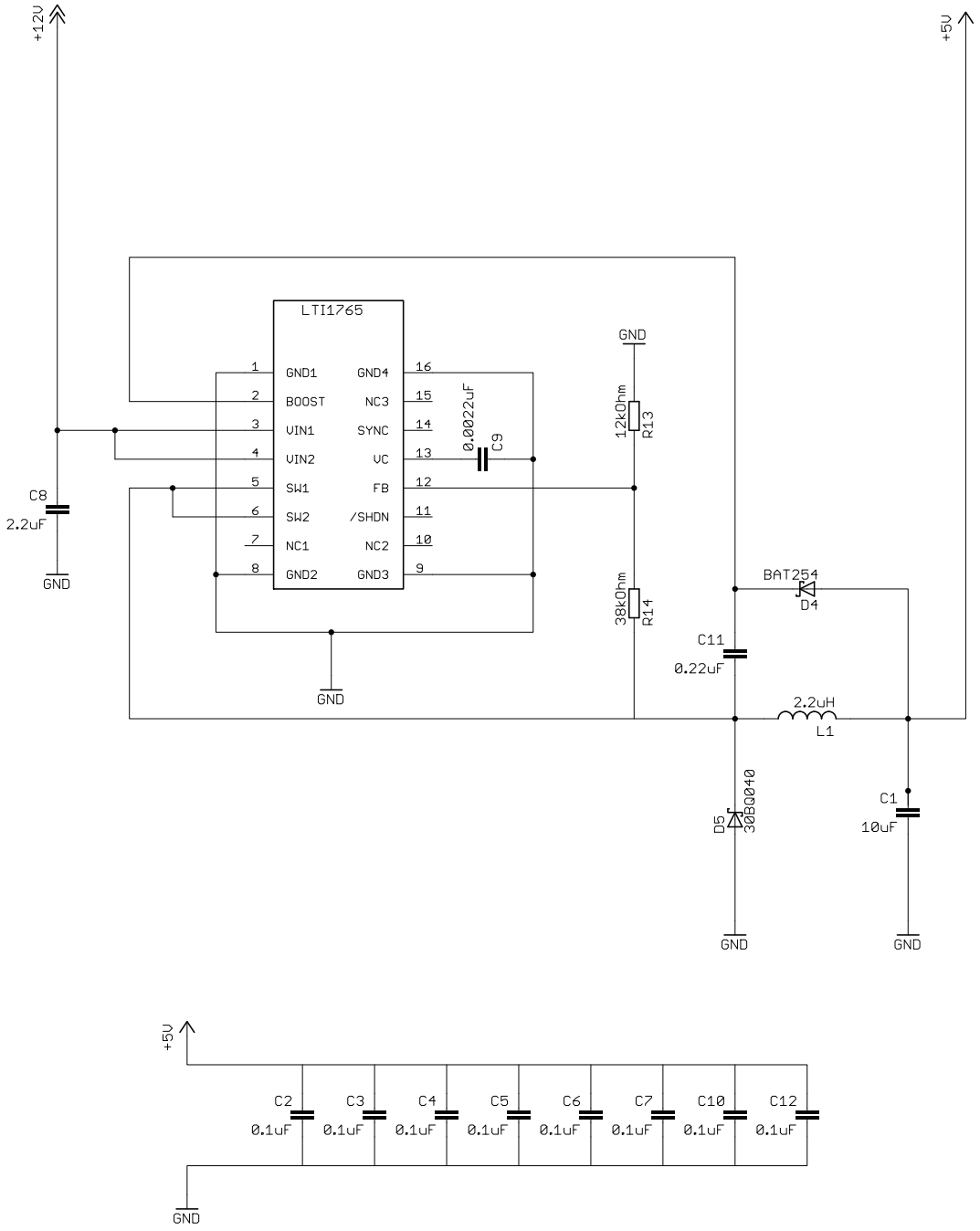


Figure 3.9: Voltage Regulation

3.5.4 Programming Interface

The programming interface chosen is JTAG. This is a simple and stable interface (when implemented as intended), and the circuitry is shown in Fig.3.10. The JTAG connector is a Molex 8-pin connector. The connection is done according to the JTAG ICE datasheet[2]. The pull-up resistor is added for the JTAG to have complete control over the reset-pin and its logical level during programming. The Molex connector is chosen because of its small size, and to keep a certain degree of consistency since the interface to the motor uses the same connector. This connection requires an adapter between the JTAG ICE and the presented circuitry, but this was developed during Brattbakkens previous project[4] and will be available.

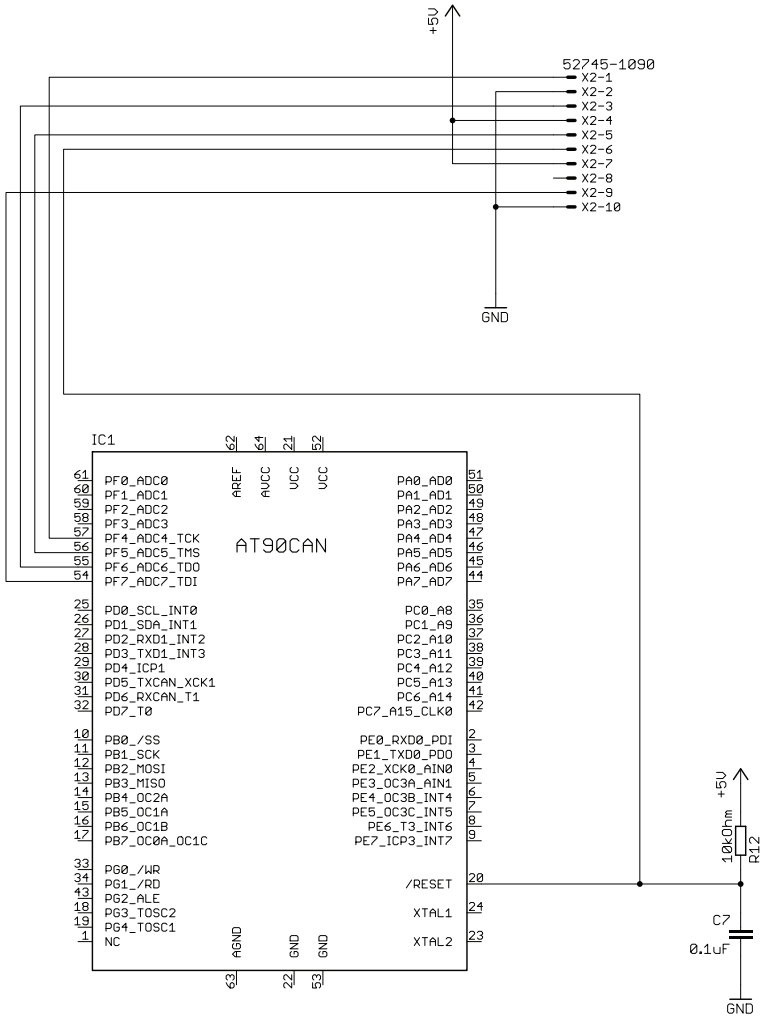


Figure 3.10: Programming Inteface

3.5.5 Motor Protection

A circuitry suggestion using low-side current sensing for protecting the motor is shown in Fig.3.11. The current through the motor coils is measured by adding a small resistor, R2, between the coils and ground. The resistor should be as small as possible, but large enough to make the voltage variations detectable. The current through the coils is equal to the current flow through the motor drivers (EL7202), and should not be exceeding the current calculated in Section 3.4.7, 96mA. The current flow will vary according to the applied PWM-signal, and is filtered through a low-pass filter. Further the signal is connected to an operational amplifier, where the voltage-level is compared to a reference created by the voltage divider, R4 and R5. The output of the operational amplifier is then forwarded to the reset-pin on a RS-switch, and to the motor controller. The output of the RS-switch further coupled to a transistor that controls the supply voltage to the motor drivers. So a current that exceeds the maximum limit, will result in the output from the operational amplifier resetting the SR-switch, and the supply voltage to the motordrivers are discontinued. The RS-switch will then have to be set by the motorcontroller to enable the motordrivers.

The resistor, R6, can be added to create a hysteresis on the output signal, but since the output will reset the SR-switch at the first passing of the reference voltage, and have to be re-enabled by the motor controller, that does not seem necessary.

The presented suggestion where due to some last minute changes, and the specific components to use have not been chosen in this thesis. Some aspects to have in mind when choosing components:

- The resistor R2 should be as small as possible, but big enough to make a detectable difference in voltage. Since the max permitted current is 96mA, and the measured voltage should be in the region of 100-200mV, a reasonable resistor value is 2Ω.

- The region of the measured voltage mentioned is also dependent of the operational amplifier to be chosen.

- The low-pass filter characteristics are dependent of how the PWM-signal is achieved, and the CPU-clock frequency chosen. Two ways to implement PWM is available in the AT90CAN128, fast PWM and Phase Correct PWM[3], which will differ the frequency of the current to filtrate. It is important that the time constant for the filter is less than the time constant in relation with the heat conduction in the motor, to prevent the motor from overheating before the circuitry reacts.

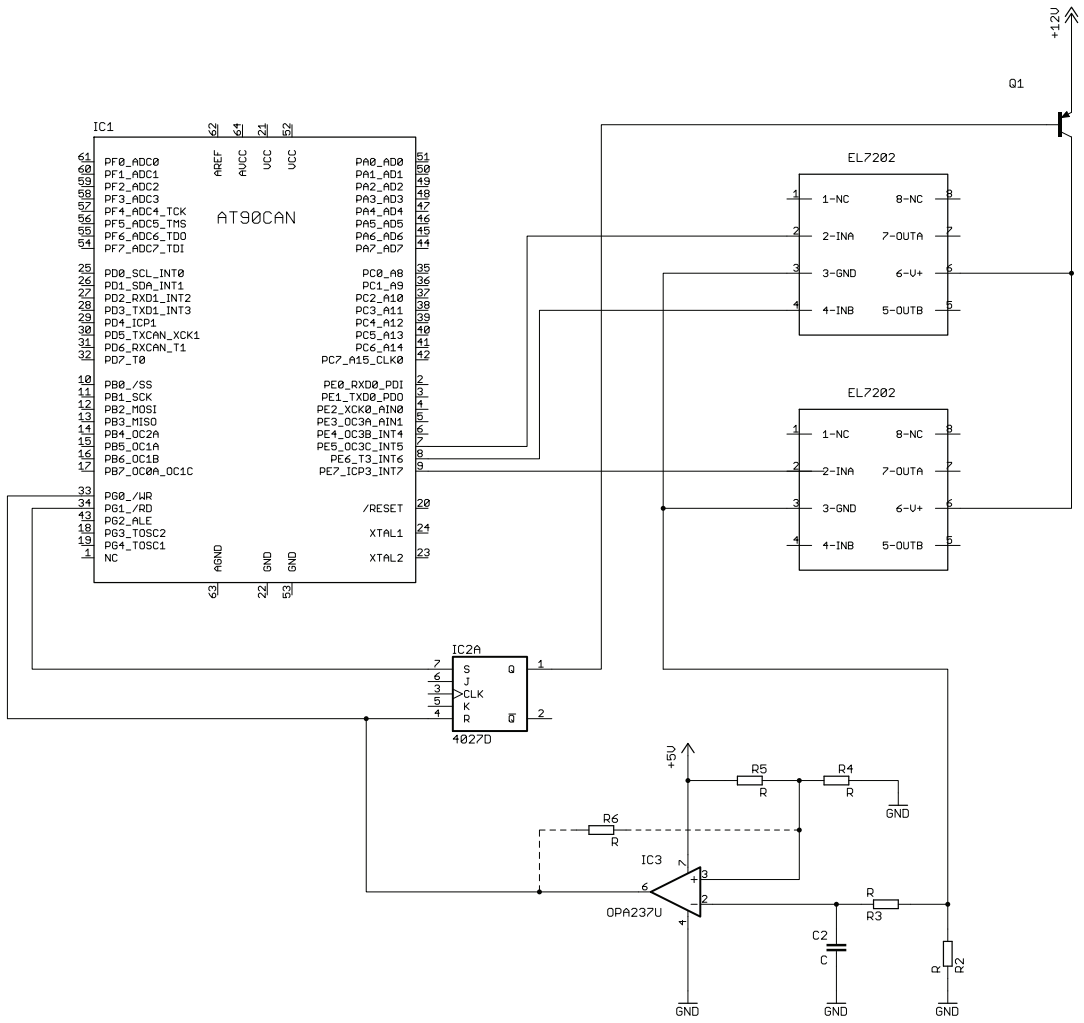


Figure 3.11: Motor Protection

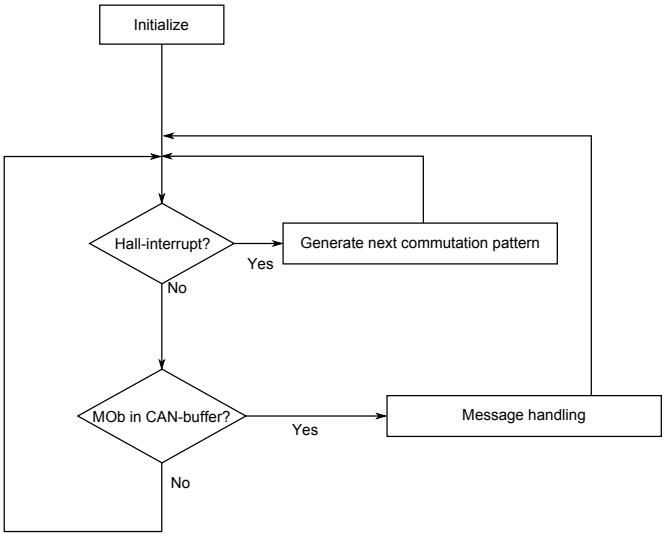


Figure 3.12: Main Flow Chart

3.6 Software

During design of the hardware some of the software to be implemented have been kept in mind. The main flow chart is seen in Fig.3.12. It needs to be implemented as a big while loop, where the higher priority assignments are at the top. The motor commutation has first priority, and then the message handling and calculation of setpoints follows. The priority is achieved by making the while-loop start at the top after a task is finished. The CAN-messages will mostly be handled by automatic functions in the CAN-controller, and the only processing related to the messages will be calculating new setpoints. However, by an effective way of composing messages containing setpoints, the setpoints could be used directly.

3.6.1 Motor Control

The software for the motor can to a great extent evolve from the mentioned application note from Atmel. Their solution also includes handling CAN-messages, so it should be easy to add the ones proposed in the UNB-protocol. Commutation of the motor is done in the following steps:

1. A change in logical level on one of the Hall-elements invokes an interrupt. The same interrupt service routine is invoked independently of which element caused the interrupt.

2. All Hall-elements are read to get a pattern to indicate the position of the

motor.

3. The pattern is masked with a commutation scheme to get the next commutation pattern.

4. The output port is set according to the achieved pattern.

The commutation scheme in the code provided uses six outputs and PWM of the high-side outputs. The motor driver chosen for the NRWD uses only three signals, discussed in Section 3.4.3, and the code needs to be changed. A solution can be to set the relevant pins, the ones with "X" in the tables 3.2 and 3.3, as inputs, or in some way enable the internal pull-up resistor.

3.7 Testing

Unfortunately not much time where left for testing, and will be due to further work. There will, however, be two circuit boards intended for testing available. One was made during the previous project by Brattbakken[4], that includes the AT90CAN128 and makes the pins available through soldering pads for connection with wires. This can be seen in the Fig.3.13. The board was intentionally made for testing on both the AT90CAN128 and the ATtiny2313, and therefore the ATtiny is also present on the circuit board. And the other was made during this thesis, and is the complete motor driver circuitry, and can be seen in Fig.3.14, with the belonging circuit diagram in Fig.3.15. Both boards are complete with soldered components.

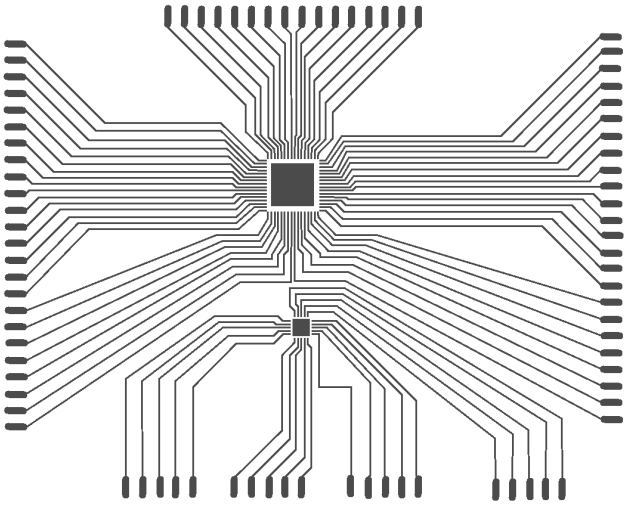


Figure 3.13: AT90CAN128 Board

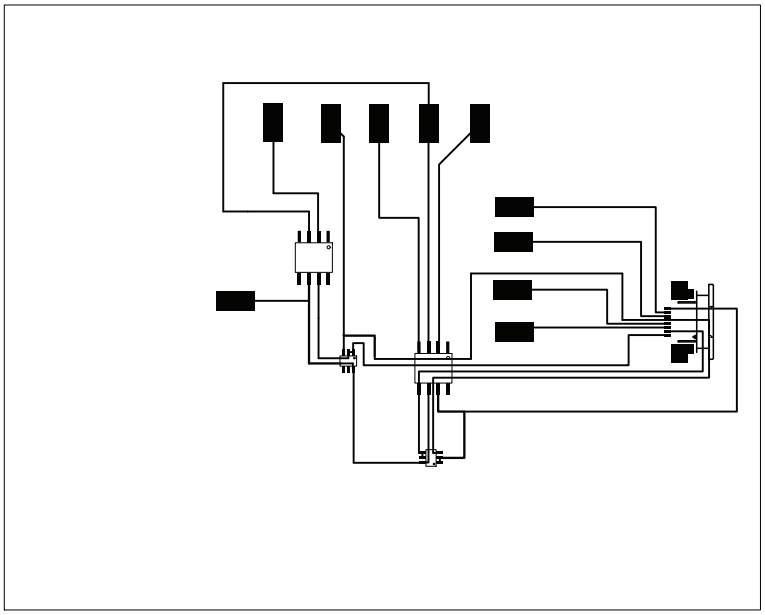


Figure 3.14: Test Driver Circuit Board

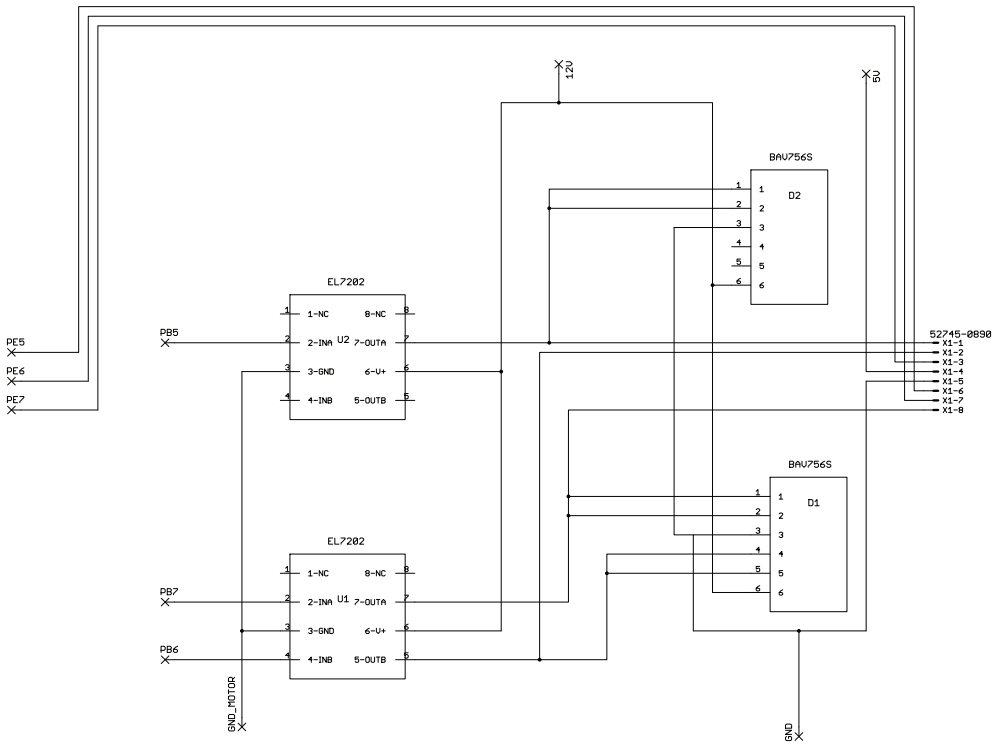


Figure 3.15: Test Driver Circuit Diagram

Chapter 4

Discussion

4.1 Communication

Expansion of the UNB-protocol

The way the profile message has been composed will demand a certain degree of memory in the main controller. One could implement a simpler version of the message, by using fewer of the available bytes in a message, but then at the cost of the number of nodes that can be recognized. The proposed functionality will have to be compared to the new revised UNB-protocol that is to be released any time soon. The message added was intended to be a part of a handshake during initialization of a new node. However, the resulting message can be requested by the master module at any time.

Device Profiles

Adding device profiles to the communication protocol should be a requirement, and it is likely to believe that the final protocol will contain this functionality. As long as the protocol becomes a guideline for all electric prosthesis manufacturers the possibilities are endless, and each prosthesis can more easily be custom-made for each user.

The down-sides of adding device profiles must be the complexity by adding another function to the protocol, and as always when adding more functionality, bring in more application code that could be error-prone.

One or two buses

After the release of the first revision of the UNB-protocol there have been some discussions regarding the number of buses to implement in a prosthesis that is

to communicate according to the UNB-protocol. The first revision of the UNB-protocol does not mention anything about more than one bus, but discussion indicates that two buses can be standardized, one for the actuators and one for the sensors. For the proposed message this will not be a problem. The message can easily adapt into the new setting and can be used unchanged, even though the second byte can seem superfluous it is still needed if other profiles is to be connected, e.g., equipment for analysis and development.

4.2 NRWD Hardware

Configurations

The choice of removing the analog interfaces for communication was based on the fact that the wrist intentionally was thought to be completely digital complying with the UNB-protocol, and the UNB-protocol only. This seemed a little strict, that every sensor to be used with the NRWD had to include an CAN-controller, and the choice of including an analog interface on the proximal side was made, but the distal analog interface still left out. The final specification will make the wrist able to communicate according to the UNB-protocol, but also in a stand-alone setting with an EMG-sensor, without the need of a CAN-controller in the sensor. However, if the wrist should be compatible with older systems the distal analog interface should possibly be added.

Architecture

The communication interfaces chosen are done with the UNB-protocol in mind, and adding another analog interface should be considered to make the wrist compatible with older systems. This did not seem relevant during the beginning of the project and were therefore left out.

The use of jumpers to select witch interface to use makes the wrist very modular. Another jumper could be added to signalize the controller, letting it know what interface is used, and making it able to go into a different state space. Some caution should be taken. The electronics must be able to tolerate a lot of movement and shaking, and the jumpers must be installed in such a manner that they are not able to fall off.

Motor Control

The choice of implementing the motor control in the same controller as communication control is doubtlessly risky. But by the calculations done in Section 3.4.1, it seems to be a good chance of making it work. In addition there is the possibility

of setting the CPU-clock to 16MHz which is possible with the 5V supply voltage. This would, however, increase the power consumption by the controller and it is desirable to maintain the default 8MHz.

Motordriver

The solution with one signal to the drivers for each phase adds some complexity to the motorcontrol software. In order to be able to control the speed, but maintain the momentum, the signal obviously have to be PWM, and then when the signal should be low according to the commutation scheme PWM must be disabled. Any undetermined state (states that are not supposed to be high or low) needs to be set to high-impedance.

Programming Interface

It may seem unnatural to keep the same interface that previously led to the malfunction of the wrist. This problem has now been completely removed, and there is no possibility that the same error could arise. The problem resulted from the communication between two controllers, but now there is only one present.

Motor Protection

In the first attempt for the motor protection circuitry the signal from the motor protection circuitry was meant to be connected to an AND-gate together with the control signal from the motor. Because of the approach used for the motordriver, where some of the signals need to be able to be set to high-impedance, the logical computation in the AND-gates would possibly make the circuitry fail. If the output-pin on the motorcontroller is set to high-impedance, the AND-gate output will be 0V (ground) and the voltage would split over two motor coils instead of one. The second solution uses a transistor to control the supply voltage of the motor driver.

Chapter 5

Further Work

Device Profiles

According to e-mail correspondence with the developer of the UNB-protocol, some functionality close to device profiles will be added to the next revision of the protocol. The proposed expansion should be compared with their solution for connecting devices without the need of updating of the system.

CAN-interface

Except a simple through connector, no physical interface for the CAN- communication is chosen. A stable and versatile connector should be standardized and added to the circuitry.

In addition, a choice should be made to decide whether the wrist should be compliant with the use of two buses. This choice may be obvious according to the revised protocol to be released. If two buses are to be implemented one CAN-controller must be added.

Motor Protection

Specific components needs to be chosen for the motor circuitry. Max permissible current are calculated in Chapter 3.3, and the low-pass filter values will be dependent of what PWM-solution is implemented and CPU-clock frequency of the motor controller.

Testing

Unfortunately no time were left for testing and this should be done thoroughly, before constructing the complete circuitry.

Designing Circuit Board

A circuit board that fulfills the requirements regarding size should be constructed from the presented circuit diagram.

Software development

The complete control code must be written. The hardware design demands efficient and high quality code.

Chapter 6

Conclusion

The assignment has led to a complete circuit diagram that to a great extent is ready for a circuit board. The subcircuits should obviously first be tested individually, and some minor adjustments are probably necessary, but overall, a circuit that fulfills the requirements is developed. The components chosen are all compliant with the small circuit board that is necessary to fit in the NRWD housing. Much of the circuitry from previous projects have been reused after careful review and combined with the new aspects added in this thesis. The adding of the motor protection circuitry will contribute to make the NRWD a robust and fail-safe device.

It is chosen an area efficient solution for the complete control of the wrist, that excludes the possibility of the previous error to rise again since the communication between two controllers no longer exists. The solution demands strict and careful development of the software to be implemented, but when the appropriate code is written, this is doubtlessly the most efficient way of controlling the NRWD.

The making of the expansion for the UNB-protocol has also been interesting and can hopefully contribute to the final standardized protocol. This proposal may not be implemented directly, but thoughts and ideas concerning device profiles may contribute. As discussed the solution demands some memory in the main controller, but the benefits of being able to connect different devices like they where built at the same factory, are much greater than the cost of implementing memory. There is no doubt that device profiles are highly relevant, and should be implemented in the world of electrical prosthesis

Appendices

Appendix A

Functional Specifications

The following pages includes the complete revised functional specifications, which can also be found in the enclosed zip-file.

NTNU
Norwegian University of Science and Technology

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Engineering Cybernetics

REPORT

Report number	
Classification	
ISBN	

Address: **NTNU**
Department of Engineering Cybernetics
O.S. Bragstads plass 2D
NO-7491 TRONDHEIM, NORWAY

Switchboard: +47 73 59 40 00
Telephone: +47 73 59 43 76
Fax: +47 73 59 43 99

Report title Functional Requirements Specification for the NTNU Revolute Wrist Device (NRWD) v0.3	Date Number of pages 10 Project supervisor Øyvind Stavdahl
Author (s) Øyvind Stavdahl, Inge Brattbakken	Project no. TBC; Spør Tove
Sponsoring Organization NTNU	Client's reference

Abstract

This document specifies the functional requirements for the initial version of the NTNU Revolute Wrist Device (NRWD).
The specification is intended to provide the necessary basis for developing a technical requirements specifications with respect to mechanical, electrical and algorithmic properties.

TBC

Keywords

TBC		

Contents

List of Figures

1. Background

The functionally optimal 1-DOF wrist prosthesis kinematics was theoretically derived in (Stavdahl, 2002). Now one aims at implementing a physical device based on these principles. The current document constitutes a functional specification for this device, which covers not only the kinematics but even mechanical, electrical, electronic and algorithmic aspects.

2. Revision History

When	Who	What
2005.02.08	Ø.Stavdahl	Brief revisions for FPGA-based version. Several requirements and commentsclarified and typos fixed. This relates to: GEN-08, WJF-01, WJF-02, WJF-03, WSF-01-03, WSF-03, WCF-02, WCF-03, WCF-04, WCF-04-01, WCF-05, WCF-09, WCF-10, WPF-02, WPF-03, Circuit board geometry included.
Spring 2010	Inge Brattbakken	Revised specifications regarding communication. This relates to WCF-02 and WCF-03. Also removed requirements no longer relevant. Those beeing: WCF-04, WCF-04-01, WCF-04-02, WCF-06, WCF-08 and WCF-09.

3. Conventions

3.1 Abbreviations etc

The following conventions apply to this document:

NA	Not Applicable; irrelevant
HW	Hardware
SW	Software
TBC	To Be Completed; an aspect that has to be filled in.
TBD	To Be Defined; an aspect that is still not completely defined.

Use of the word *shall* denotes requirements that must be met by the system, while the word *should* denotes requirements that are desirable and must be met unless justification is provided for an alternative.

3.2 Document Structure

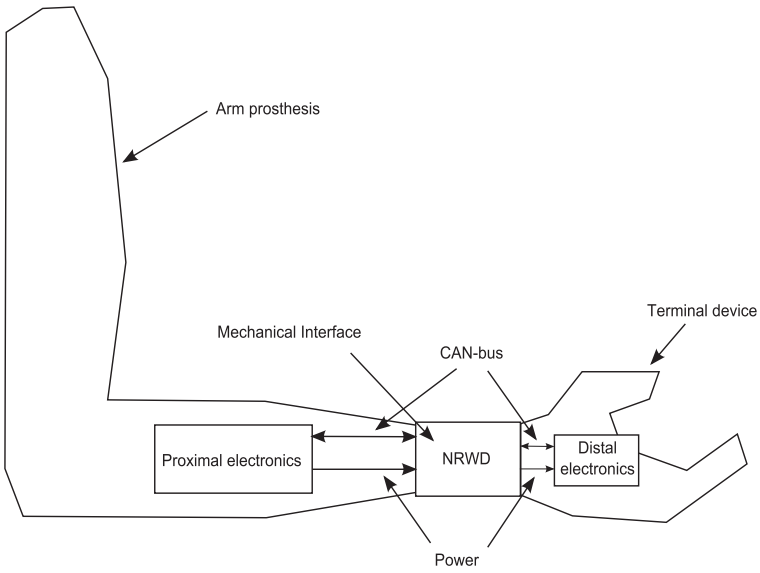
TBC.

4. System Description

4.1 Context and Purpose

The “system” referred to here is the entire wrist prosthesis, including both mechanical, electrical/electronic and

software components. The primary purpose of the system is to rotate/orient the terminal device with respect to the forearm according to user input. Figur 1gives an overview of the system and its context.



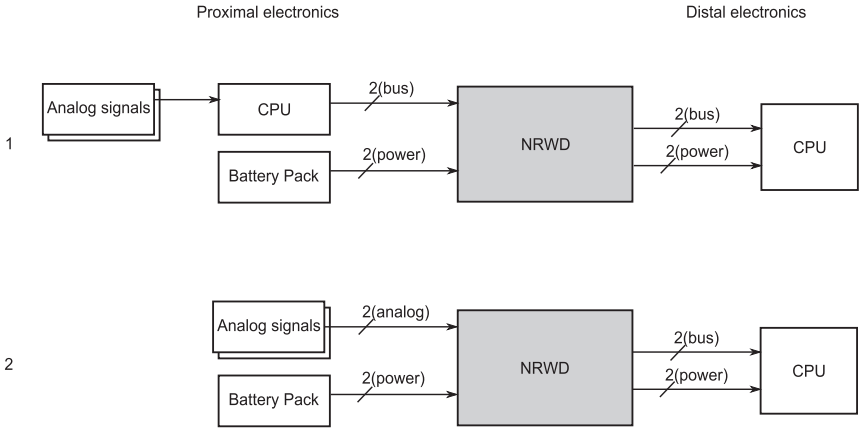
Figur 1 System context

The proximal electronics includes a battery pack or another electric power source, as well as digital circuits with sensors for reading the user's motor intent (typically EMG electrodes, switches, "pressure pads" or the like) and communication of this intent to the joint controllers in the prosthesis. While the figure suggests that the proximal electronics is situated in the forearm, this needs not be the case; in the case of a total arm replacement the prosthesis may comprise motorized shoulder, elbow, wrist and/or finger functions, and the arm electronics may correspondingly be distributed in the different parts of the arm. The "Proximal electronics" box Figur 1 thus represents all the electronics proximal to the wrist, while the "Distal electronics" represents any or all electronic components distal to the wrist, such as a motor controller responsible for opening and closing of the hand.

4.2 Configurations

The system is applicable to several different equipment configurations, as indicated in Figure 2. These configurations
include the following, listed in the order of relevance and only the first two being absolutely necessary to implement:

1. A completely digital mode where all intercomponent communication is based on a data bus. This mode is for use with other novel systems that support the same communication protocol.
2. A hybrid mode where the proximal communication (i.e. between the wrist and proximal electronics) is analog while the distal communication is digital. This is for using a novel hand in systems based on analog electrodes and the like.



Figur 2 The two different equipment configurations

5. Functional Requirements

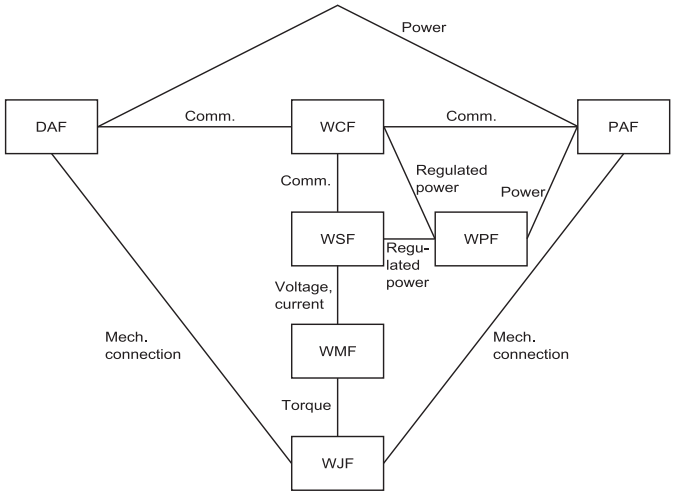
5.1 General requirements

The NRWD system shall implement the following functions and satisfy the following requirements.

Req. no.	Description	Comments
GEN-01	A joint which implements the optimal kinematics described in (Stavdahl, 2002) – Wrist Joint Function, WJF.	
GEN-02	An electric motor which drives the movement of the joint via a gear train – Wrist Motor Function, WMF.	
GEN-03	A motor control interface and regulation process to control the movements of the WJF. – Wrist Servo Function, WSF.	
GEN-04	A communication function that interfaces with WSF and other connected systems – Wrist Communication Function, WCF.	"Other connected systems" typically include proximal and distal joint controllers, as well as units for system diagnosis and configuration.
GEN-05	A power adaptation function that enables the system to run from a variety of voltage levels – Wrist Power Function, WPF.	Compliance with commercial and research systems.

GEN-06	A mechanism for attachment of the wrist to the forearm socket. – Proximal Attachment Function, PAF.	
GEN-06-01	The PAF **shall** be disconnectable and, when disconnected, allow entry to the space proximal to the wrist.	For maintenance and replacement of forearm electronics.
GEN-07	A mechanism for attachment of the wrist to the terminal device – Distal Attachment Function, DAF.	
GEN-08	An outer geometry that crudely resembles that of an adult wrist – Wrist Geometry Function, WGF.	No component shall extend beyond the envelope of a normal wrist, which has an elliptic crosssectional area of approx. 5 cm x 4 cm. All parts and connectors must be kept small.
GEN-08-01	The longitudinal dimension (along the forearm) of the entire NRWD **should** be as short as possible, and **shall** not exceed 65 mm.	Allows longer residual limbs to use it, i.e. “larger market”. 65 mm is Bock spec.
GEN-09	The system **should** consume a minimum of energy, and the current conspumtion **shall** be kept below 2A at all times.	When motor is active, its output dictates a lower bound for the power consumption. This requirement just implies that unnecessary circuits should be turned off, and active use should be made of the controller’s various sleep modes.
GEN-10	The system shall be modular with respect to both HW and SW.	
GEN-11	The weight of the entire NRWD **should** not exceed 100 g.	Industrial components: Otto Bock: 96 g, VASI:100 g.

Figur 3 depicts the functions and their dependencies.



Figur 3 Functional block diagram

5.2 Wrist Joint Function, WJF

The following requirements are derived from GEN-01.

Req. no.	Description	Comments
WJF-01	The NRWD joint **shall** be a single, simple revolute joint which axis of rotation can be placed at an attitude with respect to the forearm and terminal device as spedcified in (Stavdahl, 2002) Equations (7.4) and (7.5).	Derived from (specific version of) GEN-01. Note a typographical error in Eqn. (7.5); last element of vector should read “-0.23” instead of “0.23”.
WJF-02	The joint axis **should** be manually adjustable to other attitudes than that specified in WJF-01.	In order to test other axis alignments. Note that adjustability wrt. forearm AND hand requires TWO adjustable functions!
WJF-03	The joint **shall** enable an angular excursion of at least 180 degrees. The excursion **should** be unlimited.	180 deg is crudely that of a healthy limb.

5.3 Wrist Motor Function, WMF

The following requirements are derived from GEN-02.

Req. no.	Description	Comments
WMF-01	The wrist joint (output of the gear train) **shall** have a maximum angular	Otto Bock spec. No-load speed

	velocity of at least 1.4 rad/s (81 deg/s). The maximum velocity s**hould** be as high as possible.	
WMF-02	The wrist joint **should** have a maximum torque of at least 34,3 mNm.	VASI spec. Stall torque
WMF-03	The motor **shall** have a maximum mechanical output power of at least TBC W.	
WMF-04	The motor **shall** be protected from overheating. The protection **should** be implemented by hardware.	

5.4 Wrist Servo Function, WSF

The following requirements are derived from GEN-03.

Req. no.	Description	Comments
WSF-01	The movements of the wrist joint **shall** be controllable according to the follwing modes: 1. On/Off-mode 2. Position mode. 3. Velocity mode	
WSF-01-01	In On/Off-mode the motor shall be at rest or run at maximum speed (open-loop) in one direction according to a given setpoint.	Requires only transistor bridge.
WSF-01-02	In position mode the joint angle **shall** be proportional to a given angular setpoint.	
WSF-01-02-01	The WSF **shall** include an absolute position sensor. This sensor **shall** provide no less than 10 bits resolution per revolution.	
WSF-01-03	In velocity mode the joint angular velocity **shall** be crudely proportional to a given velocity setpoint.	"Crudely proportional" implies that there is no explicit need for velocity feedback, the speed may be controlled in open-loop.
WSF-01-03-01	The WSF **shall** include a velocity sensor or estimator.	If brushless motor: use Hall elements for speed estimates.
WSF-02	The movements of the wrist joint **should** be controllable according to the follwing modes: 4. Torque mode 5. Impedance mode	
WSF-02-04	In torque mode the motor torque **shall** be proportional to a given torque setpoint.	Torque crudely proportional with motor current, so current can be used for feedback.

WSF-02-04-01	The WSF **should** include means for monitoring motor current.	Also follows from GEN-09.
WSF-02-05	In impedance mode the mechanical impedance of the joint **shall** be determined by a given impedance setpoint.	Mech. Impedance = torque/velocity. May require strain gauge measurements etc. to “bypass” friction.
WSF-03	WSF **shall** provide an interface to WCF through which the modes (described in WFS-01 to WSF-02) can be selected and relevant parameters can be set, and through which WSF can report relevant state variables TBD.	Typically: Setpoints in, process values out. The precise content of this communication will be defined by a protocol sepcification (SCIP) TBD.

5.5 Wrist Communication Function, WCF

The following requirements are derived from GEN-04.

Req. no.	Description	Comments
WCF-01	The NRWD **shall** have a two-wire Proximal Communication Interface (PCI) and a twowire Distal Communication Interface (DCI).	
WCF-02	The PCI **shall** be configurable so that it implements two (0V, 7,2V) analog input lines or a bidirectional two-wire CAN interface with the UNB protocol(Losier,2009).	The 7.2V spec is approximate. For protocol, see comment re. WSF-03.
WCF-02-01	The analog input lines **shall** be able to sample both lines at a rate of 1 kHz. The sampling rate **should** be as high as 2 kHz.	EMG signals have a bandwidth of approx. 500Hz.
WCF-03	The DCI **shall** be configurable so that it implements two (0 V, 7.2 V) analog output lines or a bidirectional two-wire CAN interface with the UNB-protocol.	The 7.2V spec is approximate; it may be acceptable to reduce this to 5.0V.
WCF-05	The WCF **shall** be configurable to an alldigital mode, with the PCI and the DCI acting as bidirectional CAN bus interfaces.	Only a single CAN interface is necessary, as both PCI and DCI can be internally connected to this single interface.
WCF-07	The WCF **should** be configurable to a hybrid mode in which the PCI acts as two analog input lines while DCI acts as a bidirectional CAN-interface (cf. WCF-01).	
WCF-10	The WCF **shall** include a serial interface for downloading software and for debugging/diagnostic purposes.	Same fashion as AVR Butterfly serial programming. This requires a bootloader. Might include RS-232 and/or other proper interfaces.
WCF-08	WCF **shall** implement an interface to	

	WSF according to WSF-03.	

5.6 Wrist Power Function, WPF

The following requirements are derived from GEN-05.

Req. no.	Description	Comments
WPF-01	The WPF **shall** accept external power in the form of an unregulated two-wire DC supply.	Implies on-board voltage regulation.
WPF-02	The NRWD **shall** tolerate and run normally when powered with a voltage in the range (6 V, 12 V). The range of usable voltages **should** be as wide as (5 V, 18 V).	This corresponds to Otto Bock, Motion Control and other systems. Upper limit possibly to be relaxed (lowered).
WPF-03	The NRWD shall tolerate supply voltage in the range (0 V, 12 V) without exhibiting unpredictable behaviour and without getting damaged.	E.g. shutting down the controller before the voltage gets dangerously low, may otherwise damage Flash and EEPROM content etc. See comments to WPF-02.
WPF-04	The NRWD **should** automatically limit its motor current to a level that does not reduce the supply voltage below the interval given in WPF-02.	Low batteries => careful motor control to avoid power-down.

5.7 Proximal Attachment Function, PAF

The following requirements are derived from GEN-06 and more.

Req. no.	Description	Comments
PAF-01	The PAF **shall** comprise two parts, the proximal of which is adapted to be permanently attached to the forearm socket and the distal permanently attached to the wrist unit. The parts must "mate" to form a mechanically stable connection while also being detachable.	A commercially available "quick disconnect" unit may be used, but the entire mechanism must be kept as short as possible.
PAF-01-01	The proximal part of the PAF **shall** be hollow to allow access to the space within the socket proximally to the wrist.	Batteries and electrodes etc. is mounted here, so an opening must be present to allow maintenance and replacement of these units.
PAF-02	PAF disconnection **should** be possible with hand or a simple tool, e.g. a screwdriver.	
PAF-03	The PAF **shall** include a four-wire electric coupling, preferably mechanically integrated with the PAF itself.	GEN-04 and GEN-05. Preferably a "quick disconnect" type, optionally loose wires and a manually detachable coupling/plug.
PAF-03-01	The PAF electrical coupling **shall** include at least two power supply wires/contacts capable of transferring a constant current of 4A per wire.	This is the current for the wrist AND the terminal device.
PAF-03-02	The PAF electrical coupling **should** be rotatable without twisting the wires.	Brush rings etc. This requirement and DAF-02-02 are mutually exclusive; both are not needed!

5.8 Distal Attachment Function, DAF

The following requirements are derived from GEN-07 and more.

Req. no.	Description	Comments
DAF-01	The DAF **should** comprise two parts, the proximal of which is permanently attached to the wrist and the distal permanently attached to the terminal device. The parts must "mate" to form a mechanically stable connection while also being detachable.	The distal part may be a function of the terminal device, e.g. a Bock hand. No "quick disconnect" required here!
DAF-02	The DAF **shall** include a four-wire electric coupling, preferably mechanically integrated with the	GEN-04 and GEN-05. Preferably a "quick disconnect" type, optionally loose wires and a manually

	DAF itself.	detachable coupling/plug.
DAF-02-01	The DAF electrical coupling **shall** include at least two wires/contacts capable of transferring a constant current of 2 A per wire, and these wires **shall** be connected to the power supply wires from PAF.	This is the current that drives the terminal device. Check with Otto Bock (=1 A?); higher currents needed for more advanced/multifunction hands.
DAF-02-02	The DAF electrical coupling **should** be rotatable without twisting the wires.	Brush rings etc. This requirement and PAF-03-02 are mutually exclusive; both are not needed!

Bibliography

Losier, Yves. 2009. *Communication protocol for the AIF UNB hand project.* University of New Brunswick. 2009.

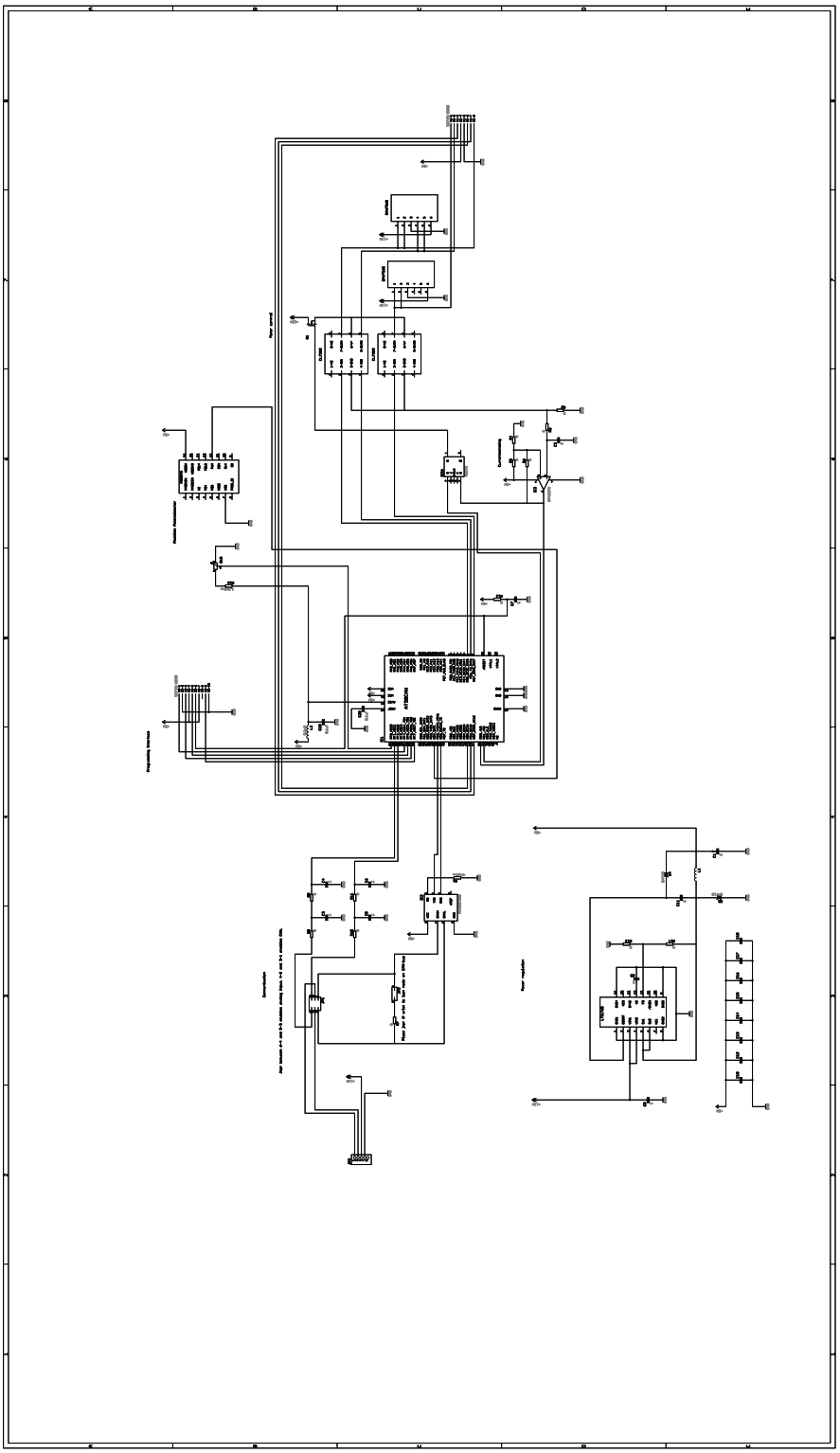
Stavdahl, Øivind. 2002. *Optimal wrist prosthesis kinematics: Three-dimensional rotation statistics and parameter estimation.* Department of Engineering Cybernetics, Norwegian University of Science and Technology. 2002. PhD Thesis.

Stavdahl, Øyvind. 2002. *Functional Requirements Specification.* 2002.

Appendix B

Circuit Diagram

This is the complete circuit diagram for the NRWD. The format is A3, and so the fonts become unreadable when down-sized to fit here, but a pdf-version of the circuitry will be added to the enclosed zip-file.



Bibliography

[1] Atmel. AVR452: Sensor-based Control of Three Phase Brushless DC Motors Using AT90CAN128/64/32. Application Note, 2005.

[2] Atmel. Datasheet: JTAG ICE User guide. `http://www.atmel.com/dyn/resources/prod_documents/doc2475.pdf`, Accessed Sept2009.

[3] Atmel. Datasheet: AT90CAN128. `http://www.atmel.com/dyn/resources/prod_documents/doc7679.pdf`, Accessed: September09.

[4] Inge Brattbakken. Innvevd maskinvare for kybernetisk håndledd. Specialization Project, 2009.

[5] Faulhaber. Datasheet: 0620K012B Brushless DC-Servomotor. `http://www.faulhaber.com/uploadpk/EN_0620B_MIN.pdf`, Accessed March 2010.

[6] Faulhaber. Brushless DC-motor Technical Information. `http://www.faulhaber.com/servlet/com.itmr.waw.servlet.FileViewer?dokmanid=590213&kdid=40929&spachid=1`, Accessed May2010.

[7] Torgrim Gjelsvik. NRWD Circuit Diagram, 2006.

[8] Intersil. Datasheet: EL7202 High Speed, Dual Channel Power MOSFET Drivers. `http://www.farnell.com/datasheets/32539.pdf`, Accessed: February2010.

[9] Yves Losier. Prosthetic Device Communication Protocol for the AIF UNB Hand Project. `http://groups.google.ca/group/scip-forum/`, 2009.

[10] NXP Semiconductors. Datasheet: PCA82C250. http://www.farnell.com/datasheets/17255.pdf, Accessed 30.05.10.

[11] Robert Bosch GmbH. CAN specification, 1991.

[12] Siemens Microelectronics. Controller Area Network introduction, 1998.

[13] Håkon Skjelten. Innvevd maskinvare for kybernetisk håndledd. Master's thesis, Norwegian University of Science and Technology, 2005.

[14] Wikipedia. OSI-model. `http://en.wikipedia.org/wiki/OSI_model`, Accessed May2010.

[15] Øyvind Stavdahl. *Optimal wrist prosthesis kinematics: Three-dimensional rotation statistics and parameter estimation.* PhD thesis, Department of Engineering Cybernetics, Norwegian University of Science and Technology, 2002.