Communication interfaces and protocols for advanced control of high-end prosthetic hands

Master of Science Thesis in Embedded Electronic System Design

SEBASTIAN KARLSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2015

Communication interfaces and protocols
for advanced control of high-end prosthetic hands

Sebastian Karlsson

Examiner: Per Larsson-Edefors

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2015

Abstract

State-of-the-art control of hand prostheses is conventionally done by using a simple threshold technique; one output from one input. Two input signals are necessary to control the two directions of one degrees of freedom (DoF), and to switch between different predefined functions either co-contraction or a physical switch is used. However, this serial operation is unnatural and slow. The purpose of this project was to develop advanced control systems for high-end prosthetic devices with the pattern recognition platform BioPatRec to allow more sophisticated control schemes. Two systems considering three different hands were developed: One system for the MyoHand VariPlus Speed or the SensorHand Speed from Ottobock, and another for the i-limb hand from Touch Bionics. In addition to these two systems, a previously proposed framework was further developed to specify a standard on how to interface a motor driven device with BioPatRec. This framework guide developers how to define their devices in BioPatRec and guide the development of the control firmware.

Keywords: Pattern recognition, Hand prosthesis, Control system.

Acknowledgements

I would like to offer special thanks to Max Ortiz Catalan, my supervisor at Integrum AB for giving me this opportunity to write a thesis in this interesting field and for his knowledge and technical support during the project. I also thank everyone else at Integrum AB who have participated with valuable support and feedback. Finally, I would like to thank Per Larsson-Edefors for being my examiner.

Sebastian Karlsson, Gothenburg, June 2015

Contents

1 Introduction

The hand is a powerful tool and its loss causes severe physical and psychological drawbacks. A hand prosthesis is an artificial device that is used as a replacement for the lost hand. A highly desirable feature for the control of artificial hand limbs is prediction of simultaneous limb motions.

In clinics around the world, the state-of-the-art technology for rehabilitation of amputees is commonly a dual-site controlled myoelectric prosthesis [1]. The method for activating such a device is simply controlled using a threshold detection technique, one output signal from one input signal which is the myoelectric signals (MES) of a group of muscles. Two input signals are necessary to control the two directions of one degree of freedom (DoF), and to switch between different predefined functions either co-contraction or a physical switch is used. However, such a serial operation is unnatural and slow. The lack of intuitive control is one of the reasons that these devices are not used regularly. Surveys reveal that 30% to 50% of upper limb amputees do not use their prosthetic devices regularly [2].

1.1 Background

Despite the impact of losing a hand and the functionalities that a hand prosthesis offer; the numbers of amputees requiring a prosthesis is too small to motivate rapid development. This is one reason why control interfaces and mechanisms barely have changed in the past 40 years [3]. Robotic hand prostheses with one or two DoF with the functionality to open/close or pronate/supinate the wrist have been commercially available since early 1970's [3].

Products with several DoF have been developed in relatively recent years and are today commercially available. The i-limb hand from Touch Bionics [4], the bebionic hand from RSL Steeper [5], the Michelangelo hand from Ottobock [6] and Vincent from Vincent systems [7] are the most used multifunctional prostheses with several DoF. These hands, as devices with only one DoF, are controlled using the standard control interface with two input control signals. Even though these hands offer sophisticated functionality they are currently controlled in an unnatural manner. This is where the research take place today; control methods to achieve intuitive control and feedback to the user. Instead of using the two input signals to only control one DoF at a time, a communication protocol could be applied to select functions individually or simultaneously.

The Azzura hand project [8] [3] is one example that aim to achieve such a system. This hand is in a prototype stage design to be controlled through serial communication. The embedded controllers in the hand are arranged in a hierarchical architecture consisting of one high level controller and five lower level motion controllers, one for each finger. Each low level motion controller controls the position, and current movement speed of its connected finger. A memory is integrated enabling memorization of grasps. Communication to the high level controller can then use the memory to launch automatic grasps.

A system with intuitive control of individual or simultaneous functions require pattern recognition from the user to be able to determine which function to select. Research lead by Dr. Ortiz Catalan at Chalmers University of Technology, Sahlgrenska University Hospital, and Integrum AB, has resulted in pattern recognition algorithms that can provide intuitive control over several degrees of freedom [1] [9].

1.2 Goals

The assignment of this project was to interface, high-end commercially available robotic hand prostheses, with the pattern recognition platform BioPatRec [9] to allow more sophisticated control schemes to access individually powered joints. Electronics, printed circuit board (PCB) design and communication protocols are the areas that are involved in the project. The goals of this project are the following:

- Implement hardware communication protocols to interface between the hand prosthesis device and pattern recognition platform.
- Implement PC interface using BioPatRec [9].
- Implement real-time control for individual and simultaneous movements within BioPatRec.

The i-limb hand from Touch Bionics [4], the VariPlus Speed [10] hand and the SensorHand Speed [11] from Ottobock are three hands that were evaluated in this project. It was expected that the hands differ in communication methods; different prototypes were therefore produced for each hand. The aforementioned goals are applied for each hand in an iterative manner, where all three points are first applied to one hand before the next hand is evaluated.

In the case of the i-limb hand, the interface to the pattern recognition platform was already implemented. The task involving this hand was to refine this interface by designing a common PCB for all the motor drivers and implement the interface on another microprocessor platform.

2
Development methods

In this chapter, the challenges are analyzed and solutions are proposed to address them. The chosen development tools are listed and the advantages of their utilization are described in addition to the testing methods.

2.1 Analysis of problems

The project comprises different disciplines ranging from PCB design and electronics, to software programing and communication protocols. Because of the high level of dependencies between different disciplines and modules, it is necessary to define clear interfaces and specifications. Limitations in hardware because of constraints in functionality and performance and, the other way around, functionality and performance demand certain support by the system. One of the main problems is to identify these dependent relations and to integrate all modules into a well-functioning system.

The project also concerns the end users, although the goal is to design prototypes that are not meant for commercial usage; each prototype has to provide sophisticated functionality as an amputee want to use it. This means that real-time control for individual and simultaneous movements should be implemented.

Providing low latency, reliable control and low power consumption are three specific technical challenges that are considered in this project. Different analyses of the system considering the different parts is necessary, identifying the sources of variance and make decisions for the design. Decisions concerning compromises between independent optimal solutions and also compromises between subsystems. An example of such a source in this project is the firmware that is a critical sub-system of the design that comprises of several components which are non-trivial to implemented.

2.2 Requirement specification

The goal of this project was to interface, high-end commercially available, robotic hand prostheses with the pattern recognition platform BioPatRec [9] to create more sophisticated control schemes. The design, implementation and development of the prototype systems need to consider medical requirements since the systems use, and are meant to be used, as medical devices. In this project the requirements are divided into four groups: Functionality, Development, System performance and

Prototype design.

Each prototype system that is built upon different robotic hand prostheses should be able to provide real-time control for individual and simultaneous movements within BioPatRec. Priority was given to implement individual movements and then simultaneous movements.

The development of medical devices containing software requires knowledge of what the software is intended to achieve and that the use of the software fulfill those intentions with acceptable risks. Integrum's implementation of the IEC 62304 standard is used in this project to provide a framework of the development process of software for medical devices.

Power consumption and system latency are the two parameters that are considered for performance. Since the pattern recognition platform BioPatRec [9] is not yet portable to be implemented on a stand-alone microprocessor, this sub-system is not considered for performance in this project. However, the control and interfacing firmware coupled with its respective robotic hand prosthesis is expected to be driven by a 7.4V 2200mAh lithium ion battery and be able to operate 24h before the battery need to be recharged. Approximately 30ms is considered as acceptable system latency for the system to send a control command and then receive feedback if the command is accepted or not.

The design of the prototype systems is considered as low priority in this project where design to achieve functionality is the main focus rather than cosmetic design. The systems should be easy to setup consisting of one connection to a PC and the control signals should be grouped to match the standard interface for robotic hand prostheses.

2.3 Ethical aspects

Since this project didn't use any human data nor did any experiments with patients, we don't consider any ethical aspects.

2.4 Development tools

This section describes the tools that has been used for development of the software and hardware in this project.

2.4.1 MATLAB

MathWorks MATLAB [12] is a useful tool for pattern recognition processing whereas BioPatRec [9] is implemented using this tool. In this project, MATLAB is chosen as the main tool for designing, testing and implementing the interface between BioPatRec and the robotic hand prostheses. MATLAB provides a direct and simple way to handle data and algorithms using a combination of functions, scripts and command-line control. The broad selection of built-in functions that are accessible at an abstract level makes it convenient to design without considering target-specific implementation details. The graphical user interface design environment (GUIDE) [13] is used in this project to create GUIs to interact with the various control parameters.

2.4.2 TIVA LaunchPad and CCS

The Tiva C Series LaunchPad Evaluation Kit [14] is a low-cost evaluation platform from Texas Instruments (TI). This platform, together with the integrated development environment (IDE) Code Composer Studio (CCS) [15] provides tools to develop and debug embedded applications with C/C++ programming. Configurable modules with pre-defined function libraries allow development at a high abstraction level that is easy to use. Considering the limited timeframe of this project, it is important to save development time. This platform is therefore useful to implement the control systems in this project with acceptable performance. Modules used for serial communication and signal generation are particularly useful in this project for the communication interfaces from the microprocessor to the PC interface and connected hand prosthesis.

2.4.3 EAGLE

EAGLE PCB design software [16] is a tool that is commonly used by engineers worldwide. The schematic editor, layout editor and library editor allows the user to customize components and design with precision to meet their individual requirements. The free version of EAGLE that allow two-layer designs is used in this project to create a BoosterPack [17] for the Tiva LaunchPad platform.

2.5 Design Methods

Each prototype system was created considering the requirements from its connected hand prosthesis. Interfacing the different sub-systems was a major part in the design. The basic building blocks in BioPatRec [18] for communication and control is used, where the interfaces to the systems in this project are designed to be as modular as possible. Reuse of functions and settings allow flexibility in the development, where changes in one part of a prototype system don't have a large impact on the rest of the system. MATLAB and CCS is used to create the communication interface between BioPatRec and the control firmware. Basic serial communication and experiences from the TI and MathWorks communities were combined to create high-level communication interfaces between the BioPatRec platform and the microprocessor. In the case when creating the interface between the hand and the microprocessor, the communication requirements of the hand control the configuration of the interface that is made using CCS.

2.5.1 Control firmware

The control firmware for each prototype system was designed, as for the interfaces, to be as modular as possible. The firmware translates the commonly used control commands from BioPatRec [18] to the corresponding specific commands to its connected hand. The communication requirements of the connected prosthesis hand and BioPatRec determine implementation of the firmware, since these are both standalone products. CCS was the tool used when created the firmware, where hardware modules and function libraries were used to design at a high hardware abstraction level. This tool also provides detailed debugging tools for the firmware to analyze system behavior.

2.5.2 PCB design

The free version of EAGLE PCB design software [16] was used in this project. Certain objectives were considered when designing with this tool, as shown in the following listing.

- No 90 degrees wires are used.
- 24mil wide wires are used for the input supply source and 12mil wide wired are used in the rest of the design.
- Wires are routed to utilize the available space.
- Using bottom and top ground layers.

2.5.3 Graphical user interface

The graphical user interface design environment (GUIDE) [13] in MATLAB was used to create user interfaces. This tool uses a drag and drop concept for the design and generate files where the functionalities of the GUI can be specified. Each GUI for each prototype system was designed using the same structure for connecting and disconnecting to the microprocessor. The functionalities of the connected hand determine what parameters that can be controlled and what feedback that can be captured and visualized to the user.

2.6 Testing

Verifying a system can be carried out in several ways and has to be done thoroughly, since it is the only way to guarantee quality. The most efficient option, both in software and hardware, is formal verification, but due to its complicated and time-consuming nature this project uses sets of test cases most of the time. If the design is verified against a good coverage of possible inputs, then it is a high probability that the system is functioning without a failure during its operation can be guaranteed. In this project, different sub-systems were first tested separately and then together as a whole system.

2.6.1 Embedded application

The embedded application consisting of the control firmware and interfaces implemented on the microprocessor are testing thoroughly using the CCS debugger. This tool was used to analyze the behavior of the application to verify that each hardware module and function was implemented and functioning as intended.

This method confirms that the application is configured correctly and observe the state of the implementation at specific stages. Although this is a powerful tool to find errors, it is still hard to capture timing issues. Timing issues are considered by execution the firmware in segments, verifying the behavior up to a specific point before continuing the execution.

2.6.2 Communication

The serial communication between BioPatRec and the microcontroller was tested using the MATLAB debugger and the CCS debugger. Verifying that the connection is configured as intended was done by comparing the sent and received value on both ends. On the other end, the serial communication between the hand and the microcontroller was tested using the CCS debugger. Before testing the connection to the hand, the microcontroller was tested using an oscilloscope to verify the signal levels and if the messages are constructed as intended according to the communication protocol requirements of the connected hand. The connection to the hand was then verified by observing how the hand behave according the command that is sent and, if applicable, analyzing the given feedback.

2.6.3 PCB

Two methods were used in this project when testing and verifying the functionality of the PCB design. First, the design was tested if it is manufacturable using the standard design rule check (DRC) in EAGLE. Second, when the board is manufactured and all the components are mounted, the design is connected to a Tiva LaunchPad [14] configured with the intended firmware where the functionality is tested by verifying the correct output to a given input.

2.6.4 Performance

System delay and power consumption are the two parameters that are considered for performance in this project. Timers in MATLAB are useful to measure the execution time of segments in the system. More specific, timers were used to measure the execution time of sending a command and to measure the response time from the microprocessor.

Power consumption was measured for the hand and the embedded application, separately and all together using a multimeter to get an indication how much power the system is consuming. A BoosterPack for the motor controllers was designed and manufactured for the prototype system with the i-limb hand from Touch Bionics [4].

In this case, the power consumption of the BoosterPack is included with the system all together.

3
Theory

3.1 BioPatRec: Processing and pattern recognition of myoelectric signals

The simple one-muscle one-function approach for ordinary prosthetic control is unsophisticated considering the complexities of Electromyography (EMG) crosstalk, muscle co-activation, and the contribution of deep muscles [19]. This has motivated the use of a pattern recognition approach for prosthetic control, by processing multiple myoelectric signals to achieve control of many more movements.

BioPatRec is introduced as an open source software as an effort to provide a common research platform for development and evaluation of algorithms in prosthetic control [9]. All the required functions for myoelectric control, ranging from data acquisition to real-time evaluation, are included in BioPatRec. In order to facilitate utilization, the functionalities are available through GUIs. These functions and GUIs are implemented and divided in the following modules:

- Signal Recordings
- Signal Treatment
- Signal Features
- Pattern Recognition
- Control

The modular architecture of BioPatRec provide flexibility for implementation of new algorithms, by allowing modification, enhancement, or replacement of any module without affecting the others. Fig 3.1 shows how these modules are linked and structured together. The following sections briefly describes these different modules, further details can be found in the online hosting platform [18].

3.1.1 Signal Recordings

Recording of bioelectric signals is the first step in the pipeline. Three different ways of signal acquisition can be performed, each serving different purposes [9]: (1) One-shot recordings are mainly used to verify the functionality of the hardware and inspecting the signal quality. (2) Recording session, the user is provided with

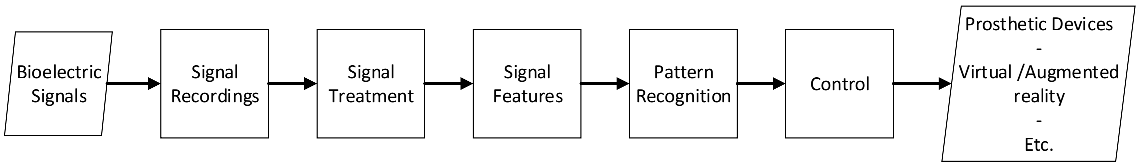


Figure 3.1: BioPatRec flow diagram. BioPatRec is organized in different modules that allows replacement or modifications without affecting each other, given that the structure is preserved.

visual instructions to perform pre selected movements and the signals from the user are recorded. (3) Recordings for real-time control, the settings from the recording session are stored in order to reproduce the setup in real-time control.

3.1.2 Signal Treatment

The recording session intend to capture as much information of the recorded signal as possible. The signal treatment module aim to optimize this information to be used for pattern recognition [9]. The BioPatRec wiki [18] list these optimizations and show how they are implemented in routines that allow: removing channels and movements that are of no interest, removing transient periods of the contraction, filtering the signals, and order the signal into segments.

3.1.3 Signal Features

The majority of pattern recognition algorithms require discrete values of the recorded signals, commonly known a signal features [9]. BioPatRec include 27 signal features that are extracted from the treated signal segmentation, handled by the signal treatment module. The result can then be used as input data to various pattern recognition algorithms.

3.1.4 Pattern Recognition

The pattern recognition module is organized in two classifications; Offline classification and Real-time classification. This separation offer utilization and flexibility during implementation and testing of new algorithms, were recorded sessions can be used to ease the process flow [9].

The Offline pattern recognition is implemented in three phases: training, validation, and testing [9]. Independent data sets are created for each of these phases using pre-recorded recording sessions. The training and validation sets are used for the learning process [18] to specify a classifier. After the completion of this process, the testing set is used to evaluate the performance of the trained classifier [9].

The Real-time pattern recognition constantly delivers predictions of movements [9]. These movement predictions can then be coupled with a prosthetic device or a virtual reality. The routines that are used to achieve these predictions require a classifier that is specified from the offline phase [18].

3.1.5 Control

BioPatRec is initially released with two control algorithms: Majority voting and Buffer output [9]. The majority voting method provide stability by filtering sporadic misclassifications at the cost of delayed response. This method uses a buffer to determine which movement that should be controlled, by choosing the movement with the most predictions within the buffer window. The Buffer output method uses a similar strategy. A threshold is set to determine if a predicted movement should be performed. In comparison the Majority voting method, the Buffer output methods allow control of simultaneous movements. In addition to these two algorithms, other control strategies or post-processing methods can be applied in order to achieve further sophisticated control of the system [9].

3.2 Serial Communication and Control

Serial point-to-point communication is preferable to transfer data when communication speed is not of high concern for performance. A serial communication protocol that is commonly used between embedded systems is the UART protocol [20]. This protocol is, in most cases, simply configured by four parameters: baud rate, parity bit, the number of data bits included in one package and the number of stop bits. The baud rate specify the modulation rate of data transmission as bits per second. The parity bit is used as a form of error detecting code and is added at the end of the package. This bit indicate if the summation of the bits within a package is even or odd. The number of data bits in a package is usually set to eight, but can be configured by the user depending on what UART module that is used. Fig 3.2 shows the structure of a UART package.

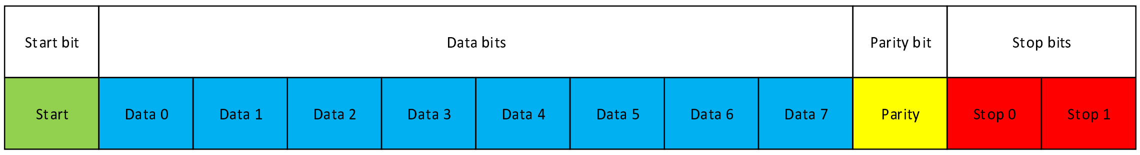


Figure 3.2: This figure show the general structure of a UART package. The number of Data bits may vary depending on how the connection is set up, as well as the parity bit that can be disabled if error checking is not needed.

3.3 Motor driven devices used in this work

Communication and control requirements as well as technical details of a robotic hand prosthesis determine the implementation of its connected control systems. Since digital control of such devices is relatively unexplored where different hands have different requirements, these control systems need to be tailored for each device.

Section 3.3.1-3.3.3 describes the technical details of the motor driven devices that are used in this project.

3.3.1 MyoHand VariPlus Speed and SensorHand Speed

The MyoHand VariPlus Speed from Ottobock [10] is one of the most commonly used prosthetic hand. This hand is design for robustness and performance rather than flexibility or several degrees of freedom. Its high gripping force (up to 100N) and speed (up to 300mm/s) make it possible to grip objects precisely and quickly [10]. Since the electronics of the hand doesn't readjust the gripping strength automatically, the targeted muscles for control have to hold and position the gripped item. Because of this feature, this hand is recommended for active patients with a low amputation level in particular.

The SensorHand Speed from Ottobock [11] feature the same features as the MyoHand VariPlus Speed. In addition as the name indicate, this hand feature a more sophisticated sensor system.

3.3.2 The i-limb hand

The i-limb hand from Touch Bionics [4] is a prosthetic hand with five individually powered and architectured fingers. The hand is programmed with 14 predefined grip patterns. These patterns are controlled and selected through the two input signals and by manually rotating the thumb to create different grasping options. The hand is design for proportional control; the stronger the input signal, the faster the fingers move. Auto grasp is another feature of this hand. This feature can sense when a gripped object is slipping and adjust the grip to secure it.

3.3.3 Servo motors

Two servo motors are integrated in a device that is used to tilt and rotate a connected hand. The angle of each motor is determined by the pulse width of its connected control signal. This signal modulation is called Pulse width Modulation (PWM). These two PWM signals are configured at 50Hz and can be generated with a pulse width ranging from 1ms(min) to 2ms(max), according to the specification of the motors, Fig 3.3. This device including the two motors was created at Integrum AB and is called the PanTilt.

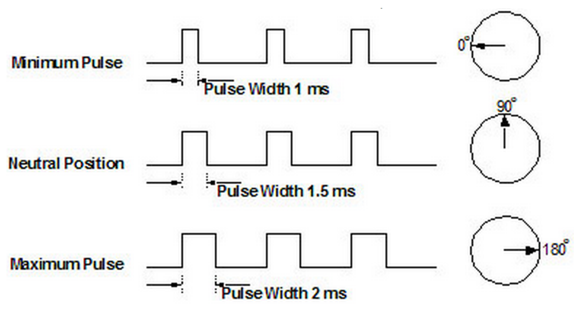


Figure 3.3: This figure show how the pulse width of the control signal determine the angle of the motor. [21]

4 Design and implementation

Two control systems have been developed in this work. One that is used for control of the MyoHand VariPlus Speed [10] or the SensorHand Speed [11], and the other is used for control of the i-limb hand [4]. Both these systems are built up by several parts where each part is necessary to achieve the desired functionality and fulfill the requirements. Fig 4.1 shows a general overview of the structure and major components of these systems. A description of the design and implementation of each part within the two systems is given in this chapter.

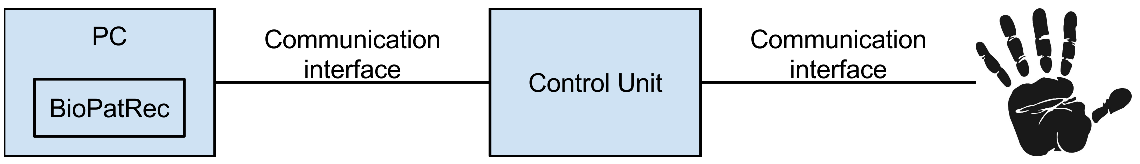


Figure 4.1: This figure show a system overview of the control system, including the three major parts: Pattern recognition, control firmware, and the connected prosthetic device.

4.1 The system for the MyoHand VariPlus Speed and the SensorHand Speed

The purpose of this system was to create a control interface between BioPatRec and a MyoHand VariPlus Speed [10] or a SensorHand Speed [11]. The data flow starts with predicting a movement using BioPatRec. Control commands of this movement is then sent to a control firmware which translate the information to send the corresponding control signals of the connected hand. These signals are in the end sent to the hand to perform the intended movement. Fig 4.1 shows the structure and components of this data path.

This system also feature a feedback data path. Both the MyoHand VariPlus Speed and the SensorHand Speed include sensors. The information from these sensors are captured by the firmware and then used to generate a feedback signal and visual feedback to the user. Fig 4.2 shows the structure and components of this feedback data path.

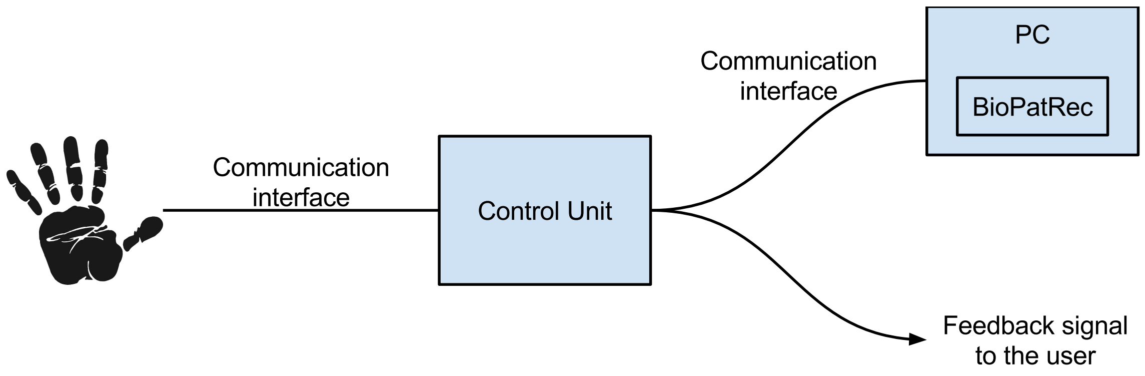


Figure 4.2: This figure show the data path of the sensor information. The feedback data can be used to generate feedback directly to the user and it can be used as input back to BioPatRec.

4.1.1 Control unit

The control unit was implemented using the TIVA LaunchPad [14] platform. As mentioned in section 2.4.2, the TIVA LaunchPad platform is a useful tool with configurable modules and pre-defined function libraries. In this system, the system clock is configured at 1MHz.

Communication interface between a PC and the firmware

In this system the UART module is used to create the communication interface between a PC and the firmware. Section 3.2 describes what parameters that need to be considered when using the UART protocol. Table 4.1 shows the setup of these parameters for the UART module in this system. These values were chosen as they are setup in BioPatRec to be able to establish a connection.

Parameter	**Value**
Baud rate	115200
Number of data bits	8
Number of stop bits	None
Parity bit	None

Table 4.1: This table shows how the parameters used for serial communication is configured.

The TIVA LaunchPad platform feature several UART modules that can be used for multiple connections. In this system, the module called UART0 is used for the communication interface between a PC and the microprocessor. This module use the debugging port to send and receive data. The debugging port uses a micro-USB connector, which can easy be connected to a PC with a micro-USB to USB cable.

Communication interface between the firmware and its connected hand

A communication interface between the firmware and its connected hand is implemented. Due to confidential information of the control of the hands I'm not allowed to describe the design of this interface, nor the communication protocol that is applied.

A PCB was created for the necessary circuitry combined with the connectors for the control signals to the hand. This PCB was designed as a BoosterPack to be attached to the TIVA LaunchPad platform. It includes few components and have space to be combined with other designs.

Sensor feedback

The feedback signal that is generated using the information from the sensors in the hands is modulated as a PWM signal. The frequency of this signal reflects the value that is capture from the sensors; the frequency range up to 10Hz. Two timer modules are used to generate this signal: one determine the duty cycle, and the other determine the period. Fig 4.3 shows how these timers are used to generate the signal. The intended usage of this signal is discussed in section 6. If selected, the feedback data is also sent to the connected PC.

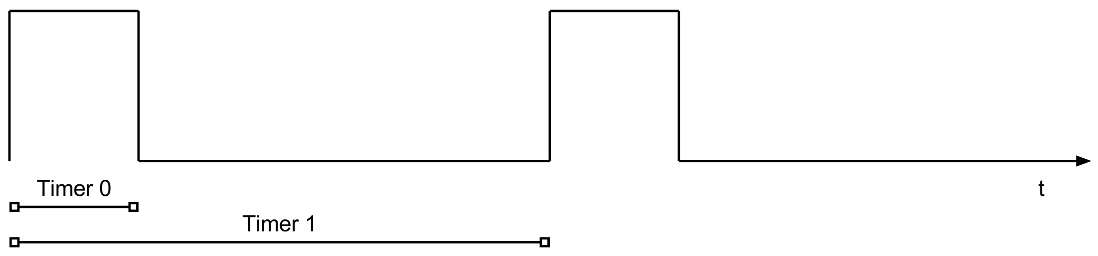


Figure 4.3: This figure shows how the timers are used to generate the PWM signal. Timer0 is used to determine the pulse width of the signal and Timer1 is used to determine the period if the signal.

Software

The firmware is divided into the following functions:

- main(): The main function.
- DeviceInit(): This function initialize the modules that are used in this system.
- UARTIntHandler(): This function handle interrupts from the UART module when data is available.
- OBHand_sendCommand(): This function send commands to the connected hand.
- TimerIntHandler(): Each timer has its own interrupt function. These functions are used to generate the feedback signal.

Fig 4.4 shows how the functions are structured and connected to each other.

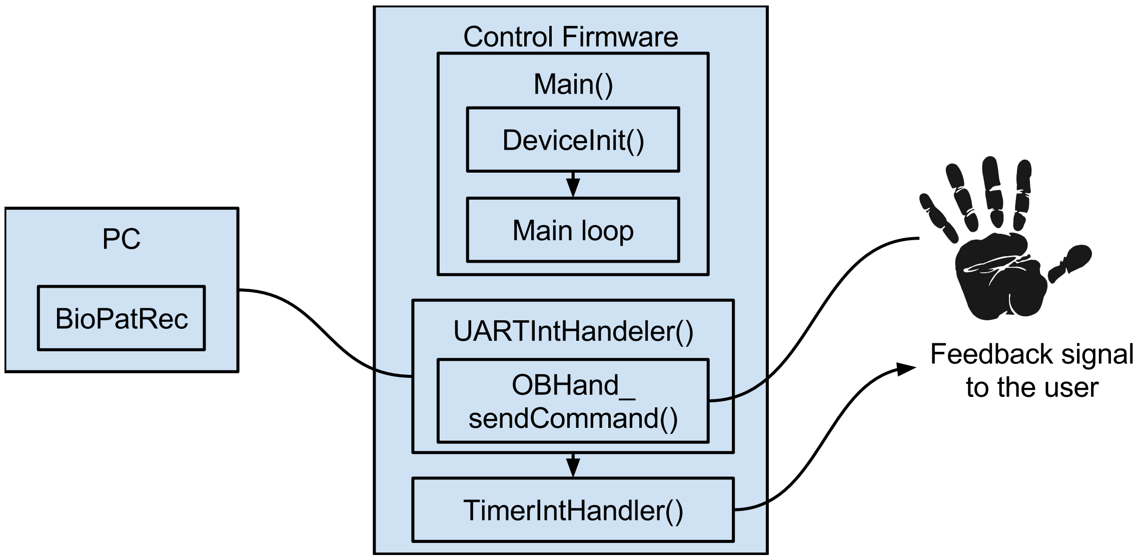


Figure 4.4: This figure show how each function of the firmware is used and how they depend on each other. The function main() initialize the modules by calling the function DeviceInit(). The function UARTIntHandles() handles the data received from the UART. This function also configure the timers and call the function OBHand_sendCommand() according to the received data.

Communication protocol between BioPatRec and the Firmware

When a UART connection is established between a PC and the firmware; this system communicate using five different types of messages. Each message start with an identifier to determine what function that should be performed. Depending on the type of the message, more parameters may follow after the id.

- **'C':** The message with this id is used to test the connection. The PC send the character 'C' and the firmware, upon successful connection, acknowledge the connection by responding with 0x1.
- **'D':** The motor control of the connected hand is performed by sending a message with this id from the PC. In addition to the id, this message include three more parameters: motor index, speed value 1, and speed value 2. The motor index is used to determine which motor to control. The two speed variables are used, as the name indicate, to set the speed of the selected motor. Two speed variables are used to determine the direction ot the motor. The firmware acknowledge receiving this message by responding with 0x1.
- **'s':** The firmware will start to send the sensor data when a message with this id is received.
- **'z':** The firmware will stop to send the sensor data when a message with this id is received.
- **'f':** After the PC have processed the data from the sensors a message with this id is sent to generate the feedback signal to the user.

Test GUI

A MATLAB GUI was made for this system which is used for simple control of the connected hand using buttons to send the control messages. This GUI was designed specifically for the the MyoHand VariPlus Speed and the SensorHand Speed which include confidential information of the hands. Therefore, no further detailed description can be presented in this report.

4.2 The i-limb system

An interface between BioPatRec and the i-limb hand have already been developed at Integrum AB. This interface was implemented on the Picollo platform [22] and use external motor driver boards, and a separate breadboard to connect the components. The i-limb hand was re-engineered to enable control of the motors by these external drivers.

The task involving the i-limb hand was to refine this interface by developing a common PCB for all the motor drivers and implement the control firmware on the TIVA LaunchPad platform.

Similar as in the system for the Ottobock hands, the data flow starts with predicting a movement using BioPatRec. Control commands of this movement is then sent to a control firmware which translate the information to the corresponding control signals. These signals are sent to the PCB which include the drivers for the motors of the hand. Fig 4.5 shows the structure and components of this data path.

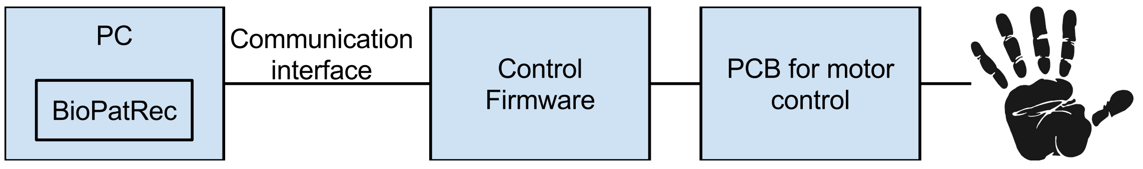


Figure 4.5: This figure show a system overview of the control system, including the four major parts: Pattern recognition, control firmware, the motor driver, and the connected prosthetic device.

4.2.1 Control unit

The control unit was implemented using the TIVA LaunchPad [14] platform as it is a useful tool with configurable modules and pre-defined function libraries. The system clock is configured at 1MHz in this system.

Communication interface between a PC and the firmware

Similar as in the system for the Ottobock hands, this system also use the UART module to create the communication interface between a PC and the firmware. This module is configured using the parameters shown in Table 4.1. These values were chosen as they are setup in BioPatRec to be able to establish a connection.

The UART module called UART0 is used for the communication interface in this system. This module is simple to connect to a PC since it use the debugging port to send and receive data.

Communication interface between the firmware and its connected hand

A communication interface between the firmware and its connected hand is implemented using the PWM modules of the TIVA LaunchPad and the external motor driver PCB, which will be described in section 4.2.2. The system include seven motors: five DC motors for each finger of the hand, and two Servo motors for the PanTilt device (described in section 3.3.3).

Each DC motor require two PWM signals to set the speed of the motor and determine the direction. The direction of a DC motor is chosen by activating one of the signals, and the speed is controlled by the duty cycle of the signal where higher duty cycle result in higher speed. These 10 PWM signals for the five motors of the hand are configured to operate at 1kHz.

In contrast to the motors of the hand, the two servo motors of the PanTilt device interpret the duty cycle as a position instead of a speed. The two motors of this device is used to rotate and tilt the hand. These two PWM signals are configured and control according to the specification described in section 3.3.3. Table 4.2 list the 12 PWM signals and show how they are connected to the motors.

Motor	**TIVA LaunchPad name**	**TIVA LaunchPad pin**
Thumb	M1PWM2	PA6
	M1PWM3	PA7
Index finger	M0PWM3	PB5
	M0PWM4	PE4
Middle finger	M0PWM5	PE5
	M0PWM2	PB4
Ring finger	M1PWM0	PD0
	M1PWM1	PD1
Pinky finger	M0PWM6	PC4
	M0PWM7	PC5
Rotate	M1PWM6	PF2
Tilt	M1PWM7	PF3

Table 4.2: This table lists the PWM signals and show how they are connected to the motors.

Software

The firmware is divided into the following functions:

- main(): The main function.
- initPWM(): This function initialize the PWM modules.
- initConsole(): This function initialize the UART module that is used for communication with a test GUI or BioPatRec.
- UARTIntHandler(): This function handles the interrupts that are triggered by receiving data on the UART.
- handleData(): This function handles the data from the UART to control the PWM signals that are connected to the motors.

Fig 4.6 shows how the functions are structured and connected to each other.

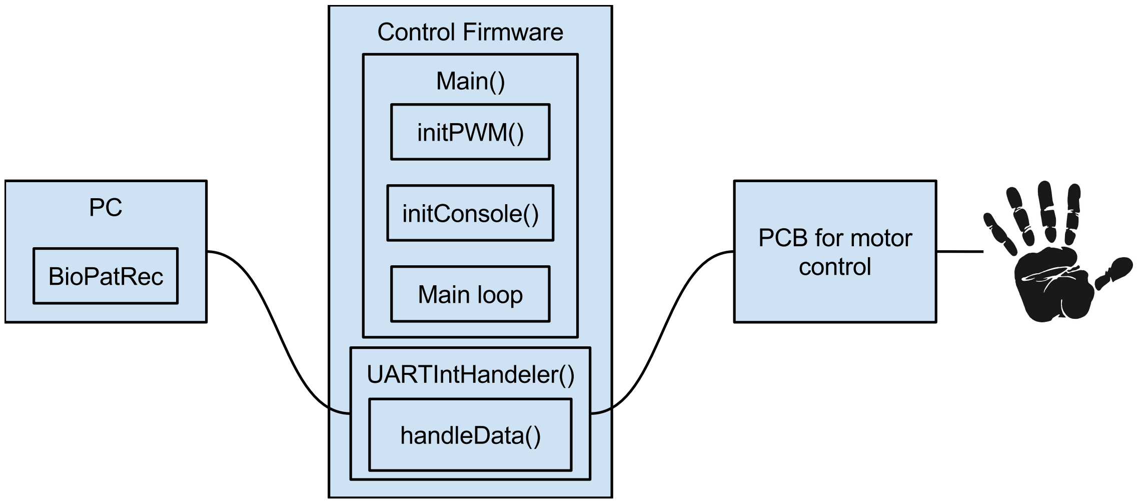


Figure 4.6: This figure show how each function of the firmware is used and how they depend on each other. The function main() initialize the modules by calling the functions initPWM() and initConsole(). The function UARTIntHandles() capture the data that is received from the UARt and send it to the function called handleData().

4.2.2 PCB for motor control

A PCB was made to combine all the external circuitry that is used to drive the motors into a common board. This PCB was designed to be connected on top if the TIVA LaunchPad as a TIVA BoosterPack [17]. It consists of three motor drivers, two voltage regulators and connectors to the hand. Fig 4.7 show the layout of this PCB where the red figure is the top layer and the blue is the bottom layer.

The components to the right named OUTA,OUTB,OUTC in the red figure are the connectors for the DC motors of the hand. The components in the upper right in the same figure with the label S1 and S2 are the connectors for the servo motors. Table 4.3 show the connections between the PCB and the motors.

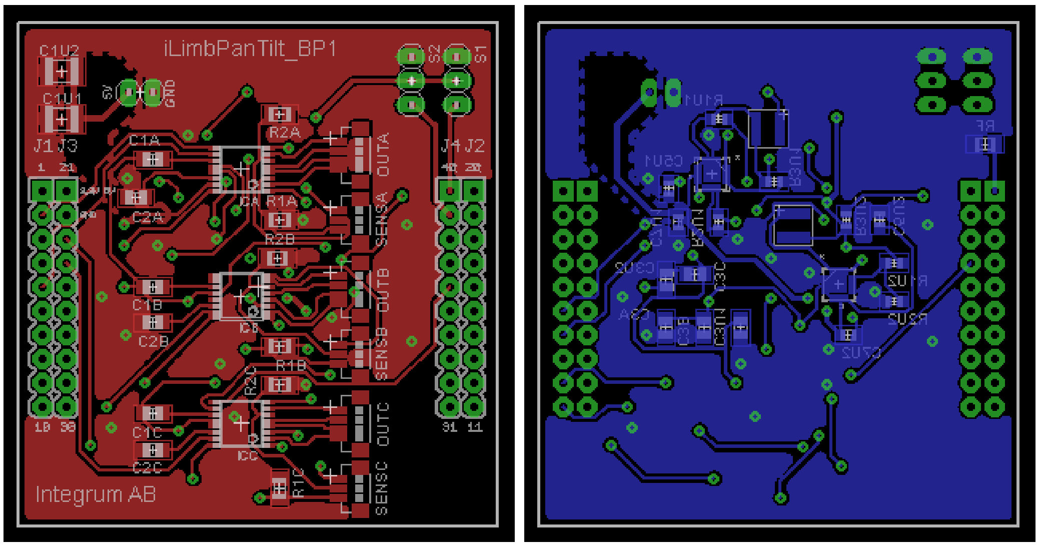


Figure 4.7: These figures shows the layout of the PCB design. The red layout is the top layer and the blue layout is the bottom layer.

Motor	**BoosterPack connector**
Thumb	OUTB
Index finger	OUTA
Middle finger	OUTA
Ring finger	OUTC
Pinky finger	OUTC
Rotate	S1
Tilt	S2

Table 4.3: This table lists the the connections between the motors and the PCB connectors.

As in the original setup, the DRV8833 [23] motor driver chip was chosen. Each of these drivers are able to drive up to two DC motors.

The DC motors of the hand have different voltage requirements: the motors for the thumb, index finger and middle finger require 4.5V, while the other two fingers require 3V. Instead of using two separate power supplies as in the original setup, two voltage regulators was implemented to regulate a common external voltage source.

4.2.3 Communication interface between BioPatRec and the Firmware

When a UART connection is established between a PC and the firmware; this system communicate with three different types of messages. As in the system for the Ottobock hands, each message start with an identifier to determine what function that should be performed. Depending on the type of the message, more parameters may follow after the id.

- **‘C’:** The message with this id is used to test the connection. The PC send the character ‘C’ and the firmware, upon successful connection, acknowledge the connection by responding with 0x1.
- **‘D’:** The motor control of the DC motors of the hand are performed by sending a message with this id from the PC. In addition to the id, this message include three more parameters: motor index, speed value 1, and speed value 2. The motor index is used to determine which motor to control. The two speed variables are used to, as the name indicate, set the speed of the selected motor. Two speed variables are used to determine the direction of the motor. The firmware acknowledge receiving this message by responding with 0x1.
- **‘S’:** The motor control of the Servo motors of the PanTilt device are performed by sending a message with this id from the PC. In addition to the id, this message include two more parameters: motor index and position. The firmware acknowledge receiving this message by responding with 0x1.

4.2.4 Test GUIs

The system was design to be compatible with the two test MATLAB GUIs from the original interface. One GUI is used for the DC motors of the hand, and the other is used for the servo motors of the PanTilt device. They both share the same functions to set up a connection to the firmware. Then each motor is controlled individually using the buttons to sent different control commands. Fig 4.8 show these GUIs.

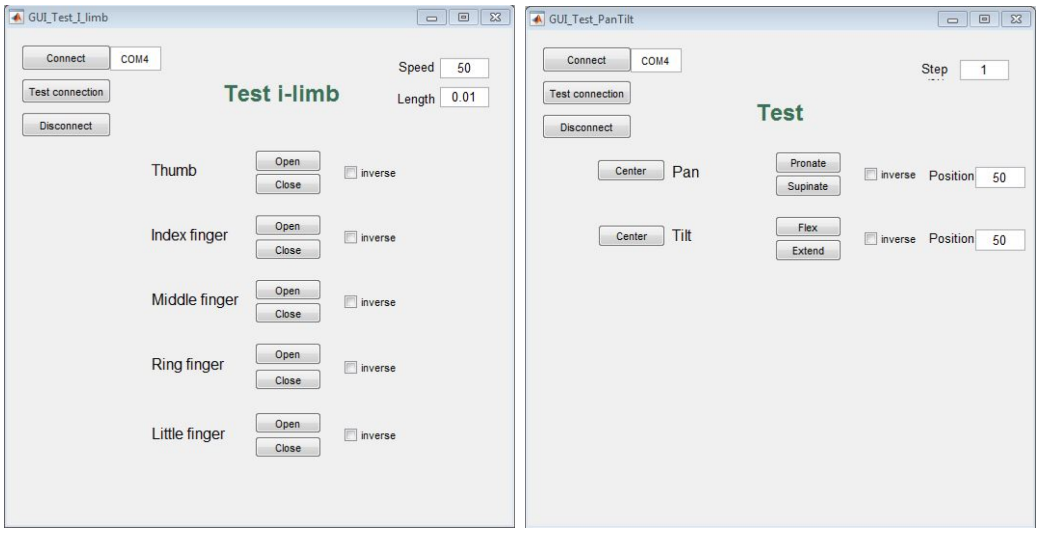


Figure 4.8: These figures shows the two GUIs that are used to control the motors. The GUI to the left control the DC motors and the GUI to the right control the servo motors.

The GUI for the DC motors of the hand have two parameters that is used to determine the speed and duration of the selected movement. The parameter “Speed” determine the speed of the motor and the parameter “Length” determine the duration the selected motor will be active.

The GUI for the Servo motors can control the angle of the motor step-wise or directly. The parameter “Step” is used to determine the distance each step result in. The boxes with the label “Position” change the motors directly according to the entered value. The buttons with the label “Center” return the motor to its starting position.

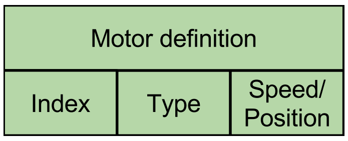
4.3 A proposed standardised interfacing framework

The two previously described systems, that were developed in this work, differ in design due to different requirements from the hand considered for each system. It is expected that the control signals of the firmware differ since the systems are implemented considering its connected hand. However, the communication interface between BioPatRec and the firmware can be improved with a common protocol.
With this is mind, we created a framework which specify a standard on how to interface a motor driven device with BioPatRec. This framework guide developers how to define their devices in BioPatRec and guide the development of the control firmware.

4.3.1 Defining a new device

Movements and how each motor is used in each movement need to be defined when interfacing a new device. This is achieved using two ".def" files in BioPatRec: one that lists the motors ("motors.def") of the device, and another that lists the movements ("movements.def") that the device can perform. A third ".def" file ("sensors.def") is used if the device is equipped with sensors.

The ".def" file for the motors is specified considering three parameters for each motor: a motor index, the motor type, and the motors speed or position. Depending if the type of the motor is specified as a DC motor or Servo motor the third parameter will be interpreted as speed or position, respectively. It is expected that the firmware is implemented using the same indices for each motor since these are used in the control commands from BioPatRec. The format of this definition is shown in Fig 4.9.



Motor definition		
Index	Type	Speed/ Position

Figure 4.9: Parameters and structure of a motor definition.

The movements that the device can perform is defined in a separate ".def" file ("movements.def"). Each movement is defined considering five parameters: a movement index, the name of the movement, a virtual reality (VRE) operation connected to this movement, the direction of the VRE operation, and a list of each motor with its direction that is used to perform this movement. The format of this definition is shown in Fig 4.10.

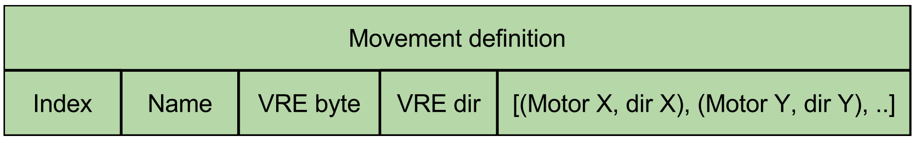


Figure 4.10: Parameters and structure of a movement definition.

Sensors are defined in a similar way in a separate ".def" file ("sensors.def"). Each sensor is defined considering two parameters: a sensor index, and the unit of the sensor data. It is expected that the firmware is implemented using the same indices for each sensor since these are used in the control commands from BioPatRec. The format of this definition is shown in Fig 4.11.

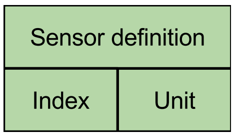
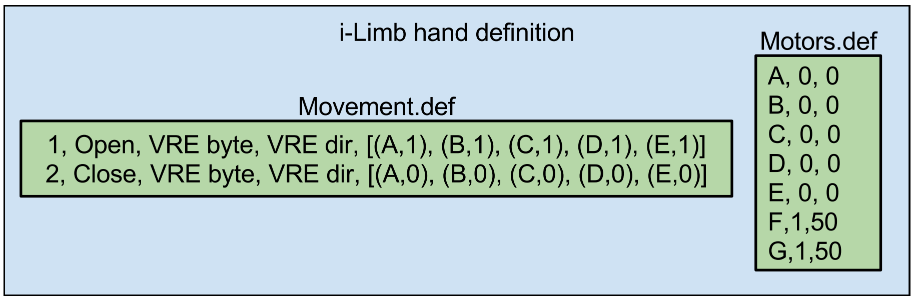


Figure 4.11: Parameters and structure of a sensor definition.

Fig 4.12 shows an example how to define the i-limb hand following this framework with the movements to open and close the hand. In this pseudo code the direction to open a motor is specified with ’1’ and close is specified with ’0’. In the motor definition, a DC motor is specified with ’0’ and a servo motor with ’1’. The third parameter that is used for speed/position in the motor definition can range from 0 to 100. For a DC motor, this parameter is used as the percentage of the motor’s speed, where 100 means 100% speed.



```
1, Open, VRE byte, VRE dir, [(A,1), (B,1), (C,1), (D,1), (E,1)]
2, Close, VRE byte, VRE dir, [(A,0), (B,0), (C,0), (D,0), (E,0)]
```

```
A, 0, 0
B, 0, 0
C, 0, 0
D, 0, 0
E, 0, 0
F,1,50
G,1,50
```

Figure 4.12: This figure show an example how to define the i-Limb hand and the PanTilt device following this framework.

4.3.2 Control structure and communication protocol

The features of BioPatRec is accessible to a new device which is defined according to the definition standards of this framework. Movements are predicted from the pattern recognition of BioPatRec. These movements are sent to a control algorithm which controls the speed of the movements and select what sensors that should be red. When the motors and sensors are selected, according to the predicted movements, messages are sent to the control firmware using the serial communication; one message for each motor or sensor. This data flow is shown in Fig 4.13.

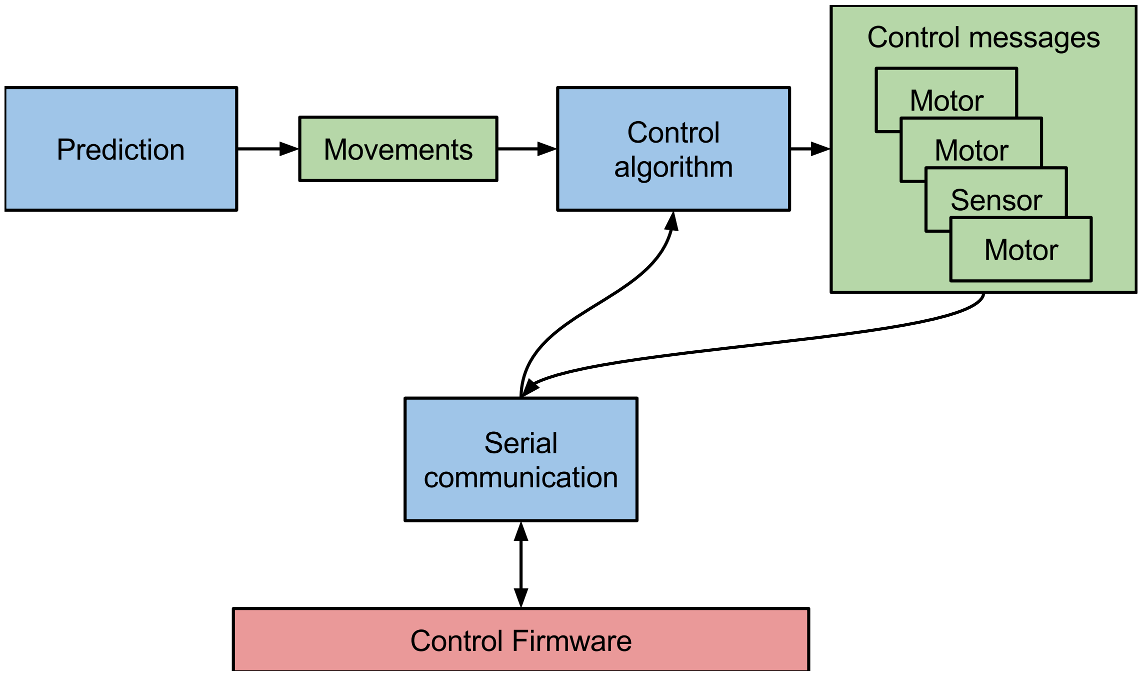


Figure 4.13: This figure shows the data flow of the framework.

There are two different types of messages: one for motor control, and another for sensor reading. The motor control message is five bytes long where each byte is one parameter. The firmware is expected to acknowledge with the received motor index when this message is received. Fig 4.14 shows the structure of this motor control message and its response.

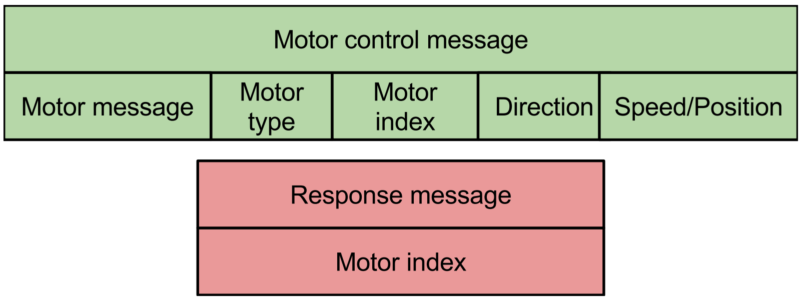


Figure 4.14: Parameters and structure of the motor message and the corresponding response message.

The sensor control message is two bytes long: the first byte is the message type, and the second is the sensor index. The firmware is expected to reply by sending the received sensor index and the current value of that sensor. Fig 4.15 shows the structure of this sensor message and the expected response.

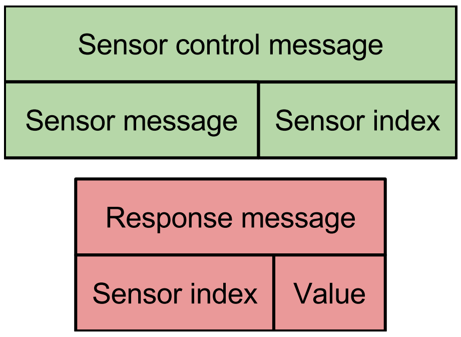


Figure 4.15: Parameters and structure of the sensor message and the corresponding response message.

5 Results

All the goals, presented in section 1.2, for each of the two systems are achieved. Three parameters were considered when evaluating each system: execution time, power consumption, and functionality. This chapter present these parameters of each of the two systems.

5.1 The system for the MyoHand VariPlus Speed and the SensorHand Speed

The control interface for the MyoHand VariPlus Speed and the SensorHand Speed achieve real-time control of the connected hand using either BioPatRec or a test GUI. The system is implemented to control the hand using 100 different speeds to open or close the hand. The feedback signal, that is intended to be connected to the user, can be generated at 90 different frequencies up to 10Hz. As described in section 3.3, these hands only feature one DoF. Therefore, simultaneous movements cannot be evaluated.

The execution time from sending a control command to receiving the response was measured from 10 samples resulting in a mean value of 6.40ms. Changing the speed and duration parameters had no effect on this execution time.

The method described in section 2.6.4, was used to measure the power consumption of the control unit and its connected hand. Two states of the system was tested: idle and operating. The current drawn by the system in the idle state was measured to 51mA. The current drawn was measured to approximately 400mA when operating the hand by closing or opening it. A 7.4V 2200mAh battery was used as the power source. This results in the system consuming 0.38W and 2.96W in the two states respectively. Table 5.1 shows these results.

System latency	6.40ms
Power consumption: idle state	0.38W
Power consumption: operating state	2.96W

Table 5.1: This table shows the results of the interface system for the Ottobock hands.

5.2 The i-limb system

The refined control interface for the i-limb hand achieve the same functionality as the original setup. Real-time control of the connected hand is done by either using BioPatRec or the test GUIs. Each motor is controlled one at a time when using the test GUIs. Simultaneous movements of the hand is handled by the control algorithms within BioPatRec. As described in section 4.3, these algorithms control each motor according the predicted movements. Each PWM signal that is connected to the DC and Servo motors are implemented to be generated with 100 different duty cycles. As described in section 4.2, the duty cycle of the signal set the speed or position a motor. This result in 100 different speeds for the DC motors and 100 different positions for the servo motors.

The same method, as in the system for the Ottobock hands, was used to measure execution time. From 10 samples a mean value of 19.87ms was measured. Changing the speed and duration parameters had no effect on this execution time.

The same method, as in the system for the Ottobock hands, was used to measure power consumption of the control unit and its connected hand. 37mA was measured as the drawn current when the system was in the idle state. Approximately 200mA was measured when operating one of the motors by closing or opening one finger. An external power supply providing the system 5V was used as the power source. This results in the system consuming 0.19W and 1W in the two states respectively. Table 5.2 shows these results.

System latency	19.87ms
Power consumption: idle state	0.19W
Power consumption: operating state	1W

Table 5.2: This table shows the results of the interface system for the i-limb hand.

6 Discussion

In this chapter, improvements, trade-offs, and the results of the two systems will be discussed and evaluated with the previous sections as reference. In addition, the proposed framework and its importance for future work is also evaluated.

6.1 System latency and Power consumption of the MyoHand VariPlus Speed and the SensorHand Speed

The priority when we implemented both control system was to achieve the intended functionality using a digital control platform. Although we manage to achieve this functionality we did not optimize for power efficiency. There are several parameters that affect both power consumption and performance of the systems. The question is: where it is beneficial to increase performance and what impact on the power consumption would the change have? For instance, the clock frequency of the TIVA LaunchPad was reduced from the maximum value of 80MHz, which is the standard configuration, to 1MHz. This was done to simplify calculations of the other modules and be able to use functions that may give strange results at high frequencies. This parameter may attract to gain more performance. However, is the performance of calculations on the TIVA LaunchPad platform the bottleneck of the system? I would say probably not; since only a handful of arithmetic operations are needed to translate the received control command to change the control signal of each motor. It is more likely that the communication interface and prediction of the movements dominate the performance of the system.

Although the presented system latency of 6.40ms and 19.87ms for the system of the Ottobock hands and the system for the i-limb hand respectively seem promising; there are other parts that need to be considered in an actual field test. Pattern recognition and other more advanced components are expected to affect the performance.

The measured idle drawn current of each system is acceptable. Robotic hand prostheses are usually driven by a battery with a capacity of 2200mAh. Considering such battery as the power source for theses systems it would result in an estimated idle life time of 30h and 40h for the system of the Ottobock hands and the system for the i-limb hand respectively.

The PCB including the motor drivers for the i-limb system was first design to use the 5V power source directly from the TIVA LaunchPad. This source turned out not to be able to provide enough current to certain motors of the hand. Therefore, the PCB was redesigned to use an external source instead. As mentioned in the description of the design and implementation, the motors of the i-limb hand require two different voltage levels, 4.5V and 3V, where two regulators are used to regulate the input voltage to these levels. The latest design still use two regulators and require the input voltage to be greater or equal to the highest output voltage, which is 4.5V in this design. This can be further improved; one of the regulators can be removed, since the PCB is not limited of the voltage levels from the connected TIVA LaunchPad anymore. This improvement would reduce approximately ¼ of the components of the PCB which would result in less power consumption. However, this improvement was discovered late in the project and is therefore not implemented because of limited time.

6.2 The proposed standardised interfacing framework

After developing two control interfaces with BioPatRec we realised that a standardized method would be useful. The purpose of the interface framework is to guide developers to interface with BioPatRec and also guide the implementation of the corresponding firmware. A new device is simply defined by following the movement, motor and sensor structures; instead of implementing separate control functions as a definition is made in the current version.

Separate control functions implemented in a high level language, with a tool such as MATLAB, may simplify the control logic of the firmware. This is often the case preferable when developing a system. However, since BioPatRec is meant to be used for many different kind of motor devices this structure would grow more and more complex as more devices are added to the platform. Therefore, we decided to create this framework to standardise the interface to BioPatRec as one common structure that only consider the motors and sensors of the connected device. All control functionality is instead guided to be implemented in the firmware using the data from this interface.

Another benefit of using the framework is that functionality of individual and simultaneous movements are ensured. The prediction mechanisms and control algorithms of BioPatRec determine which movements to perform and is able to activate the motors of the connected device accordingly by using the motor definition of the selected movements.

6.3 Future development

The two prototype systems developed in this project are both implemented for testing purposes to elaborate digital control of robotic hand prostheses. The first thing to improve, if continuing with this project, would be to modify the two systems according to the proposed framework. In the current versions, this is not implemented because the framework was developed based on the complexity that was discovered after designing the two control systems.

Further future development of similar systems might be interested in integrating the control system into a single product. To achieve this, BioPatRec would need to be portable so it can be implemented on a stand-alone device. In the current version, BioPatRec has many configurable options which are setup using the various GUIs. I think it would be useful to use a similar structure to configure a hardware platform where each function of the platform have a corresponding pre-defined hardware block. The configuration of the platform can then be programmed to some external hardware, for example an FPGA, using these blocks. In addition, the firmware for the device to be controlled would need to be implemented on the same hardware. At last, this hardware consisting of the BioPatRec configuration and the control firmware would need to be integrated with the device.

7
Conclusion

A hand prosthesis is an artificial device that is used as an replacement for a lost hand. A highly desirable feature for the control of artificial hand limbs is prediction of simultaneous motions. State-of-the-art control of such devices is done by using a simple threshold technique; one output from one input. Two input signals are necessary to control the two directions of one DoF, and to switch between different predefined functions either co-contraction or a physical switch is used. However, this serial operation is unnatural and slow.

The purpose of this project was to develop advanced control systems for high-end prosthetic devices with the pattern recognition platform BioPatRec to allow more sophisticated control schemes. Electronics, PCB design and communication protocols are areas involved when developing these systems.

Two systems considering three different hands was developed. One system for the MyoHand VariPlus Speed [10] or the SensorHand Speed [11] from Ottobock, and another for the i-limb [4] hand from Touch Bionics. Both systems achieve acceptable performance with a system latency of 6.40ms for the system considering the Ottobock hands and 19.87ms for the system considering the i-limb hand. A 7.4V 2200mAh battery is commonly used as the power source for robotic hand prostheses. If a source with this capacity is used for each of these control systems; they are both estimated to be operational longer than a day.

Creating an interface between BioPatRec and a control unit require specific functions on both sides to achieve a control system. After developing the two systems in this project we discovered that interfacing more devices became more and more complex on the software side. To address this, a proposed framework was developed to specify a standard on how to interface a motor driven device with BioPatRec. This framework guide developers how to define their devices in BioPatRec and guide the development of the control firmware.

Bibliography

[1] M. Ortiz-Catalan, B. Hakansson, and R. Branemark, "Real-time and simultaneous control of artificial limbs based on pattern recognition algorithms," *Neural Systems and Rehabilitation Engineering*, vol. 22, no. 4, pp. 756–764, 2014.

[2] D. J. Atkins, D. C. Heard, and W. H. Donovan, "Epidemiologic overview of individuals with upper-limb loss and their reported research priorities." *JPO: Journal of Prosthetics and Orthotics*, vol. 8, no. 1, pp. 2–11, 1996.

[3] C. Cipriani, M. Controzzi, and M. C. Carrozza, "The smarthand transradial prosthesis," *Journal of neuroengineering and rehabilitation*, vol. 8, no. 1, p. 29, 2011.

[4] Touch Bionics. Technology that touches lives. [Online]. Available: http://www.touchbionics.com/products/active-prostheses/i-limb-ultra

[5] RSL Steeper. The BeBionic hand - independence through technology. [Online]. Available: http://bebionic.com/the_hand

[6] Ottobock. The Michelangelo hand. [Online]. Available: http://professionals.ottobockus.com/cps/rde/xchg/ob_us_en/hs.xsl/49490.html

[7] Vincent systems. The touch sensing hand prosthesis of the next generation. [Online]. Available: http://vincentsystems.de/en/prosthetics/vincent-evolution-2/

[8] Prensilia. Self-contained robotic hands. [Online]. Available: http://www.prensilia.com/index.php?q=en/node/40

[9] M. Ortiz-Catalan, R. Brånemark, and B. Håkansson, "Biopatrec: A modular research platform for the control of artificial limbs based on pattern recognition algorithms." *Source code for biology and medicine*, vol. 8, no. 11, 2013.

[10] Ottobock. Myohand variplus speed. [Online]. Available: http://www.ottobock.com/cps/rde/xchg/ob_com_en/hs.xsl/19992.html

[11] ——. Sensorhand speed. [Online]. Available: http://www.ottobock.com/cps/rde/xchg/ob_com_en/hs.xsl/3652.html

[12] MathWorks. The language of technical computing. [Online]. Available: http://se.mathworks.com/products/matlab/whatsnew.html?s_tid=tb_14b

[13] ——. Matlab gui. [Online]. Available: http://se.mathworks.com/discovery/matlab-gui.html

[14] Texas Instruments. Tiva c series launchpad evaluation kit. [Online]. Available: http://www.ti.com/tool/ek-tm4c123gxl

[15] ——. Code composer studio (CCS) integrated development environment (IDE). [Online]. Available: http://www.ti.com/tool/ccstudio

[16] CadSoft. Eagle pcb design software. [Online]. Available: http://www.cadsoftusa.com/

[17] Texas Instruments. Boosterpacks - plug-in modules. [Online]. Available: http://www.ti.com/ww/en/launchpad/boosterpacks.html

[18] M. Ortiz-Catalan. Biopatrec. [Online]. Available: https://code.google.com/p/biopatrec/wiki/BioPatRec

[19] E. Scheme and K. Englehart, "Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use," *Journal of Rehabilitation Research & Development*, vol. 48, no. 6, p. 643, 2011.

[20] A. Osborne, "An introduction to microcomputers," 1976.

[21] SERVOCITY. Hs-5955tg servo. [Online]. Available: https://www.servocity.com/html/hs7955tg_servo.html#.VWHMOE_tlBf

[22] Texas Instruments. F28069 Piccolo controlCARD. [Online]. Available: http://www.ti.com/tool/tmdscncd28069

[23] ——. Drv8833: 2A low voltage dual brushed dc or single bipolar stepper motor driver (PWM ctrl). [Online]. Available: http://www.ti.com/product/drv8833