4. Materials & Methods

The university clinic's second Moduleaf miniMLC unit, currently not in clinical use, was taken as a development platform. Together with some spare parts and electronics obtained with the help of Siemens, a complete development environment was set up.

This chapter is split into several sections, which give an overview on how the development process was implemented and what tools are required. Furthermore, the test setup and data analysis tools are explained in detail.

4.1. Hardware

The Moduleaf, originally developed by MRC Systems GmbH/Germany consists of 80 leaves with a leaf width of 2.5 mm in the isocentre. It therefore matches conformation needs also for the majority of brain stereotactic treatments (Wu et al. 2009). When mounted on an Elekta (Elekta AB, Stockholm, Sweden) LinAc, it is rotated by 90° with regard to the integrated MLC. The maximum leaf travel speed is 30 mm/s in the isocentre (Schlegel et al. 2006).

By original design, only step and shoot IMRT treatments are possible. The collimator control system simply cuts power to the motors once a given position has been reached. Rotational treatments, VMAT/RapidArc or tracking are therefore impossible in the original implementation. Being a clear limitation, efforts were taken to re-engineer parts of the commercial system.

In its original configuration, the Moduleaf consists of a number of hardware parts that communicate with each other. This communication is outlined in figure 4.1.

The control computer is a Pentium III class standard desktop PC in a custom housing, equipped with 128 MB of random access memory (RAM). One of the ISA slots, however, is equipped with a SORCUS (SORCUS Computer GmbH, Heidelberg, Germany) ML-4 single board computer (SBC), including modules for the Controller Area Network (CAN) bus and digital input/output (DIO).

The CAN-bus is then connected via a multi-core cable, where also power and interlock signals are transmitted, to the collimator itself. Inside the carbonium casing, 6 C167CR (Infineon Technologies AG, Munich, Germany) microcontrollers are attached to the CAN bus. Each of them controls up to 16 leaves and has several interlock lines connected to the control system as well.

Given the age of the system, several components such as ISA bus single board computers are not produced anymore and do not provide the necessary data through-

A. Böhler, *Collimator-Based Tracking with an Add-On Multileaf Collimator,* BestMedDiss,
DOI 10.1007/978-3-658-10658-4_2, © Springer Fachmedien Wiesbaden 2015

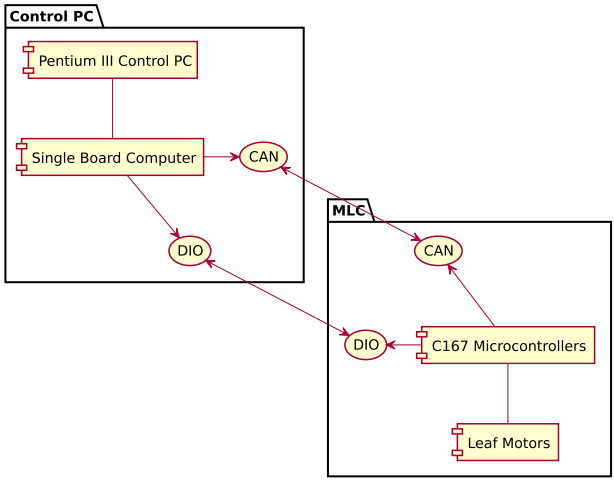

Figure 4.1: Original implementation; the Single Board Computer is connected to the microcontroller system and responsible for safety monitoring

put for realtime applications. Thus, those components were either removed or replaced by newer systems.

Figure 4.2 (Böhler et al. 2015) visualises the interplay of the updated components. The C167CR microcontrollers were left in place, but communicate directly via CAN with the control computer. The intermediate single board computer is completely removed and its functionality is now incorporated into other subsystems.

SORCUS's real-time operating system on the ML-4 acted for one part as a bridge between the control computer and the CAN bus. Being an intelligent component, it also supervised the information flow on the CAN message level, but also on the Interlock line level. Thus, only if all microcontrollers reported a healthy state, the LinAc's interlock was released. The last functionality of this SBC is therefore the communication with the LinAc, depending on the configuration using only Interlock lines or more advanced systems like the CAN bus for tighter integration.

The supervision of the CAN message flow is now incorporated into the control application on the control computer itself, while the interlock supervision requires an external system. This additional safety controller, which also communicates with the LinAc, is explained in more detail in section 4.1.2.

Some of the important components along with their characteristics are presented

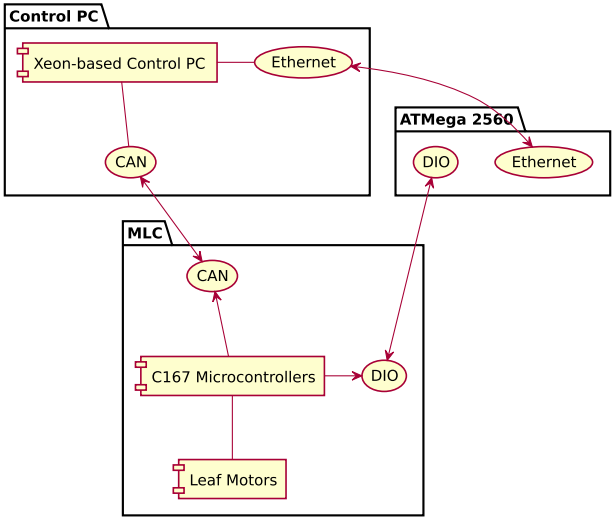

Figure 4.2: New implementation; the control computer is directly attached to the microcontroller system and an additional microcontroller is responsible for safety monitoring in the following subsections.

4.1.1. C167CR microcontroller

The Infineon C167CR microcontroller units (MCUs), already present in the commercial design, were not modified in terms of hardware components. However, they play a key role in both old and new software designs as they directly control the movement of the leaves. The C167 family is binary-compatible to the older C166 MCU. This particular variant, the C167CR-LM 16-bit MCU, is clocked at 20 MHz, features 2 kB internal RAM for variables, register banks, system stack and code as well as another 2 kB of XRAM, which can be used for variables, user stack and code, too. A key feature of this microcontroller is the interrupt system that provides 56 interrupt nodes which can be configured in 16 different priority levels. Directly on the chip are two 16-channel capture/compare units, four 16-bit timers/counters and a 4-channel pulse width modulation (PWM) unit. Furthermore, there are five additional general-purpose timers/counters, an asynchronous/synchronous serial channel unit (USART), a 10-bit 16-channel analogue/digital (AD) converter and a CAN bus module. 111 input/output (IO) lines with individual bit addressability are available for additional input and output tasks.

An external CAN transceiver is connected to the CAN bus controller that provides access to the physical layer of the CAN bus. This device is configured so that it receives only messages that match the microcontroller's CAN ID (message mask). It raises an interrupt whenever a message is ready for retrieval, eliminating the need to constantly poll for new messages.

To control the motors, up to 16 motor drivers are connected to the IO lines of the microcontrollers. The drivers require a pulse width modulated (PWM) signal that directly corresponds to the speed of the motor. The PWM signal is not generated by the on-board PWM units, but rather by using the interrupt subsystem for practical reasons.

Since each leaf has two high-precision potentiometers attached that can be used to read the current position, AD converters are necessary. Because of the high precision requirements, the onboard AD converter is unused. Instead, an external 16-bit AD converter is connected to the IO lines. As there are up to 16 leaves, two 16-channel multiplexers and one two-channel multiplexer are attached as well. This way, up to 32 potentiometers can be read one after the other with a single AD converter.

A number of light emitting diodes (LEDs) that report system states and two interlock lines are attached to the digital IOs as well.

4.1.2. Safety microcontroller

As mentioned before, another way of supervising the interlock lines and communicating with the LinAc had to be found after the removal of the SBC. Being aware of rapid prototyping platforms, the Arduino open source platform was chosen for this purpose. On the selected model, the Arduino Mega2560, an Atmel (Atmel Corporation, San Jose, USA) ATMega2560 8-bit microcontroller is present. In this case, the MCU operates at 16 MHz, provides 54 IO lines, 8 kB of RAM and 256 kB of Flash ROM for storing user code. A 10-bit AD converter is provided, which is multiplexed on 16 channels. Four hardware USART units can be used for input and output, as well as the serial peripheral interface (SPI) bus for communication with external peripherals.

For communication with the control computer, the Arduino Ethernet Shield was put in place. It uses a WIZnet (WIZNET, Hwangsaeul-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, South Korea) W5100 ethernet controller, connected by SPI to the ATMega MCU. The WIZnet ethernet interface is a 100 Mbit ethernet controller that incorporates the TCP/IP stack on-chip, offloading the TCP/IP protocol handling from the MCU to this dedicated chip. Up to four simultaneous socket connections are supported, an interrupt signals the ATMega the arrival of new data.

The Arduino's IO lines were connected to the MLC hardware and the LinAc, respectively. It is in charge of resetting the MLC MCUs, supervising the interlock lines, controlling power to the DC motors and signalling the LinAc when to release and when to inhibit the beam.

4.1.3. Control Computer

The new design relocates a lot of the processing power required to control the movements of the DC motors to the control application. Furthermore, adaptations to the treatment field are done on-line, while the beam is on. Therefore, a powerful machine is necessary that can handle these tasks without being at its limit. A current Intel Quad-Core Xeon machine equipped with 4 GB of RAM and two Gigabit ethernet cards was installed instead of the old control machine.

Using the first ethernet interface, communication with the safety MCU is performed, while the second ethernet interface provides an uplink to the hospital network.

Communication with the MLC MCU subsystem is realised via a Peripheral Component Interconnect (PCI) CAN controller, as explained in the following section 4.1.4.

4.1.4. Softing PCI-CAN interface

In order to give the control computer access to and direct control over the CAN bus, a dedicated PCI CAN controller was installed. This Softing (Softing AG, Haar, Germany) CAN-AC2 controller features two CAN channels and is fully compliant with the CAN bus specification. Drivers are provided for a wide range of Windows operating systems, as well as for Linux.

4.2. Software

All these new hardware and electronical components cannot operate without software. The main development work was put into the software part, as a lot of hardware components were readily available and seemed to be sufficient to reach the goal. This section therefore details the development environment, MCU software, control software and control loops.

4.2.1. Development Environment

The Moduleaf's original software was implemented using a number of different tools. The C167 MCU firmware was written using the Keil (Keil, Munich, Germany) development tools for C166. In order to programme the SBC, Borland (Borland Software Corporation, Austin, USA) C Compiler 3.1 for DOS was used. The control computer's software is implemented using an ActiveX component in C++ as well as a graphical user interface (GUI) written in VisualBasic. Several external libraries are necessary to provide treatment plan import/export capabilites and communication interfaces to the LinAc.

The changed hardware requires a different software environment and was updated to current technologies. Therefore, the C167 MCU firmware was reimplemented using the Altium Tasking VX-toolset (Altium, Belrose, Australia) for C166 [1], after initial trials with the free and now defunct HighTec GNU compiler toolchain did not succeed.

Development for the Arduino platform, despite the availability of the Arduino integrated development environment (IDE), was done using the Qt Creator IDE [2] and the avr-gcc[3] toolchain.

Qt Creator was also used for the development of the control computer's software,

[1] http://www.tasking.com/products/c166/, accessed Jan 2014
[2] http://qt-project.org, accessed Jan 2014
[3] http://gcc.gnu.org/wiki/avr-gcc, accessed Jan 2014

with the GCC toolchain[4] under the hood.

The primary develoment platform was based on Ubuntu[5]/GNU Linux 10.04 and 12.04, respectively. The Tasking toolchain requires Microsoft Windows, a dedicated virtual Windows XP environment was used for this purpose.

The development environment is built on free software wherever possible. Unfortunately, there is no free (free as in "free speech", not free as in "free beer") toolchain for the C167 family of microcontrollers available. Thus, the next best alternative was the use of HighTec's C166 compiler, which is at least freely available upon request from the developer (free as in "free beer"). As this compiler led to random crashes of the MCU, the same code was recompiled using Altium's Tasking compiler and now works reliably.

4.2.2. Microcontroller Firmware

The re-engineering of the MLC MCU firmware was one of the main development tasks in order to reach the goal of a tracking capable control system. Initially, it was investigated to what extent the commercial firmware could be used and if a rewrite was necessary. The first idea was thus to use the existing commandset for the SBC to MLC MCU communication and implement a pseudo-tracking system. For this purpose, up to 100 treatment fields were pre-calculated on the control computer an then transferred to the MCU system. Using a Python evaluation software, it was found that the latency of $>$300 ms led to uncertainties in the current positions. Furthermore, the leaves were not speed controlled, thus differences of up to 10 mm were found for a travel range of 100 mm (all values in the isocentre). The transfer of a single field to the MLC MCU system took 1.1 s, for 100 fields this is more than 1 min. The way positional information is transferred and the requirement for polling the MCU system for information were identified as limiting factors.

Based on these initial trials, it was decided to re-engineer the complete firmware and the way positional information is transferred. After initialising the system, consisting of the MCU system initialisation and the configuration via the CAN bus, the MCU system uses a hardware timer (Timer 3) unit to read the values from the potentiometers. Every 20 ms, this timer fires and the following sequence is run:

1. Set variable axis number to 1 and start a loop from axis 1 to axis 16

2. Set the first mulitplexer to the given axis

[4] http://gcc.gnu.org, accessed Jan 2014
[5] http://www.ubuntu.com, accessed Jan 2014

3. Read from the external ADC and store it as Potentiometer 1 value for the given axis
4. Set the second multiplexer to the second potentiometer system
5. Read from the external ADC and store it as Potentiometer 2 value for the given axis
6. Run one iteration of the Control Loop for the given axis
7. Set the second multiplexer to the first potentiometer system
8. Increase the axis number by one and start over

In order to be compatible with the original control software, the configuration can be changed, so that the MCU system must be polled for positional information. The control loop that is run in step 6 differentiates between the available modes. In Host Loop Mode, the newly developed operating mode, no actual control loop is run on the MLC MCU system. Rather, every 4 axes one CAN message is sent to the control computer containing the raw position values from the ADC system. Contrary to the commercial implementation, no conversion of ADC units of any kind is performed on the MCU level. This work is offloaded to the control application. Also, no filtering, smoothing or other data processing is performed on the MCU in this mode.

Upon arrival of a new target position via the CAN bus, an interrupt one level lower than the interrupt used for AD conversion is raised. This interrupt causes the CAN message handler to set a new PWM output value to the PWM generation system.

The PWM generation system itself does not use the hardware PWM units, but a software implementation for practical reasons. The C167CR features special capture/compare (CAPCOM) units, that can toggle an output when the CAPCOM unit overflows. Therefore, only the correct CAPCOM value for overflow has to be written to a register, the remaining output toggling is handled by the C167 hardware system. This makes software-based PWM generation very efficient.

When the MCU is idle, verification and plausibility checks are performed. For this purpose, the values of the two potentiometer systems are compared. If they are not within a certain, configurable range, the MCU's interlock is raised, causing a reaction by the safety controller.

In standard PID loop mode, when the commercial control software is used, the workflow differs. In this case, there is an actual PID control loop run on the MLC MCU system. Raw ADC values are also smoothed using a simple finite impulse

response (FIR) filter, averaging the last three values. This implementation ensures that the control loop is run at fixed intervals, with a period length of 20 ms.

4.2.3. Control Software

The control software on the control computer has several functionalities and is thus split into libraries, threads and logical blocks.

If the signal flow from the MCU unit is followed, new positional information arrives via the CAN bus on the PCI CAN interface. There is an open source interface driver provided by Softing, which gives the programmer access to the CAN bus. Using this API, a software library, libCOSMICrtcan, was written that provides a hardware abstraction layer (HAL) for the higher level control system.

4.2.3.1. libCOSMICrtcan

This HAL is based on the API provided by Softing and the Boost [6] C++ library. The latter is used to ease thread programming and cross-thread communication using Mutexes and Queues. Being a HAL, the library can be forced into simulation mode or, if no actual CAN controller is found, turns on simulation mode automatically. In simulation mode, the MLC MCU system can be talked to as usual, but the communication is simulated at the library level. Every CAN message sent to the MLC MCU system in this mode is reacted upon by the library. This method provides easy debugging and testing scenarios without the need to access the MLC hardware. However, no physical simulation is performed. This means that the positional feedback is based on timers only and no characteristics such as acceleration/deceleration effects, drive systems or signal runtimes are taken into account.

The use of this library is also important for portability reasons. In order to run the control system under Microsoft Windows, mainly this library needs to be adapted to the Windows driver provided by Softing. The reamining parts of the control application only use cross-platform toolkits and thus should be already compatible with Windows, too.

4.2.3.2. COSMICrt

Being the main control application, COSMICrt has great responsibility and a number of parallel tasks to accomplish. Its name, however, is based on the commercial application's name, COSMIC, which is an acronym for "Control System for Multileaf Collimators". The added suffix "rt" signals its realtime capabilities.

To better understand and explain the information flow within the application,

[6] http://www.boost.org, accessed Jan 2014

a flowchart is shown in figure 4.3. Upon startup of the application, the software itself is initialised and the configuration database is parsed. This SQLite[7] database contains information about the various tracking features, calibration data for the leaves and for the 4D phantom, etc.

Initially, this configuration can be imported from the commercial implementation's calibration file, mlc70.ini. As calibrating the system involves manual measurements using a slide gauge, it is a very tedious task that can be shared with the clinical application very easily.

Afterwards, 5 threads are initialised, each having a very specific duty.

Safety Thread It is responsible for communicating with the safety MCU. If all leaf positions are within tolerance, the LinAc is released.

Phantom Thread Its duty is the reading of the current 4D phantom positions and the calculation of the new target positions.

CAN Thread This is the most important thread as it communicates with the MLC MCUs, reads the current positions, runs the control loops and sets new PWM output values to the MLC hardware.

Governor Thread Also in communication with the safety thread, the Governor Thread constantly checks the current and target position deviation and signals the safety thread when and if the system is in or out of tolerance.

GUI Thread It is responsible for visualising the state of the software and for accepting user input.

When an irradiation is to be performed, the workflow is as follows: The record & verify system (PRVS), for instance open radART (medPhoton GmbH, Salzburg, Austria), sends a request including the field definition to the COSMICrt software. The treatment field is loaded, visualised and the leaves are brought into the initial position. When performing a tracked treatment, the tracking system must be initialised and COSMICrt told where to receive the inputs from. From now on, the treatment field is constantly adapted to the new positions of the tumour as reported by the external system.

The main interface of COSMICrt also displays a realtime graph showing the tracking target positional variations over time. The radiotherapist can use this display to verify the movements of the tumour.

[7] SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine; http://www.sqlite.org, accessed Jan 2014

http://www.springer.com/978-3-658-10657-7

Collimator-Based Tracking with an Add-On Multileaf Collimator

Modification of a commercial collimator system for realtime applications

Böhler, A.

2016, XIV, 53 p. 18 illus., 14 illus. in color., Softcover
ISBN: 978-3-658-10657-7