

**Politechnika Warszawska**  
**Wydział Mechatroniki**  
**Instytut Automatyki i Robotyki**

Praca przejściowa nr 2

Temat pracy:

Przygotowanie stanowiska akwizycji danych  
graficznych na potrzeby systemu przestrzennej  
wizualizacji odmienności morfologicznych zębów  
(Akademia Medyczna w Warszawie)

Prowadzący pracę:

**Politechnika Warszawska**

*Instytut Automatyki i Robotyki*

dr hab. inż. Barbara Putz

**Akademia Medyczna**

*Zakład Stomatologii Zachowawczej*

*Zakład Informatyki Medycznej*

lek. stom. Adam Okoń

Wykonujący pracę:

Artur Kozubski R58

Warszawa, dnia .....

# Sprawozdanie z drugiej pracy przejściowej

Artur Kozubski

Grupa R58 - Instytut Metrologii i Systemów Pomiarowych

Specjalność: Sensory i systemy pomiarowe

Rok akademicki: 2005/2006

## Spis treści:

1. Ogólny opis projektu i cel pracy	3.
2. Założenia	4.
3. Szczegóły techniczne	4.
4. Podsumowanie	28.
5. Literatura	28.

## English description:

Preparation of station for graphical data acquisition based on CNC grinder controlled by free Enhanced Machine Controller software developed at NIST (USA), which works under Linux OS with realtime kernel patch ADEOS/RTAI. This machine will be used for collecting images of cutsets. These images are required by system for three – dimensional visualisation of toothes, which is built at Medical University of Warsaw.

# 1. Ogólny opis projektu i cel pracy

Celem pracy było przygotowanie stanowiska akwizycji danych graficznych na potrzeby systemu przestrzennej wizualizacji wewnętrznej struktury zębów, opracowywanego dla Akademii Medycznej w Warszawie. Sam system zostanie stworzony w ramach pracy dyplomowej. Stworzony system zostanie wykorzystany do budowy atlasu anatomicznego udostępnionego w sieci Internet.

W celu pozyskiwania obrazów przekrojów poprzecznych zębów, została zbudowana specjalna maszyna, będąca rodzajem sterowanej numerycznie szlifierki CNC połączonej ze skanerem komputerowym.

W ramach pracy przejściowej została wykonana instalacja elektryczna urządzenia, zostało wstępnie skonfigurowane, zainstalowane i zmodyfikowane oprogramowanie sterujące maszyną (system EMC) i skanerem (poprzez interfejs SANE – *Scanner Access Now Easy* – linuksowy odpowiednik TWAIN z Windows) oraz został zaprojektowany format pliku przechowującego wokselowe modele zeskanowanych zębów wraz z zarysem algorytmu przetwarzania obrazów przekrojów i dodawania ich do wspomnianego pliku.

Fragment oryginalnego opisu projektu:

*„(...) Celem pracy jest stworzenie narzędzia badawczego i dydaktycznego umożliwiającego pokazanie odmienności morfologicznych zębów – ilości i zmienności korzeni, kanałów zębowych i struktur tkankowych.*

*Komputerowe modele przestrzenne będą przygotowane w oparciu o usunięte zęby, które zostaną specjalnym aparatem pokrojone na plastry.*

*Aparat ten składać się będzie ze stołu ruchomego sterowanego komputerem, zaopatrzonego w tarczę skrawającą, układ odsysania pyłów i skaner. Każdy przekrój zostanie sfotografowany i umieszczony w bazie danych. Baza danych zdjęć wykonanych w jednej płaszczyźnie zostanie użyta do odtworzenia wszystkich pozostałych przekrojów za pomocą programu Xuntech Slicer. (...)*

*W ramach projektu zostanie przygotowany aparat do fotografowania warstw zębów oraz oprogramowanie do obróbki zdjęć. Zostanie poddanych badaniu co najmniej 100 zębów.*

*Efekty pracy w postaci przestrzennego atlasu morfologii zębów zostaną opublikowane w sieci Internet. Witryna internetowa posiadać będzie stronę wprowadzającą, stronę będącą instrukcją obsługi i właściwy atlas. Zawarte w nim będą trzy typy przekrojów (Axial, Sagitta, Coronal). Małe obrazy dla każdego z środkowych przekrojów zęba będą zawarte w panelu sterującym. Poruszanie linii cięcia na obrazie będzie dawało nowy przekrój w panelu środkowym. Dodatkowy panel parametrów pozwoli na określenie oczekiwanej rozdzielczości (mała 200x300, średnia 800x600, duża: ponad 1000x2000 pixeli). Możliwy będzie eksport wybranego przekroju do pliku typu \*.jpg (np. do użycia w slajdach). Na górze ekranu będzie dostępny przycisk do zmiany zębów. Do każdego zęba będzie, w miarę możliwości dokumentacja radiologiczna.*

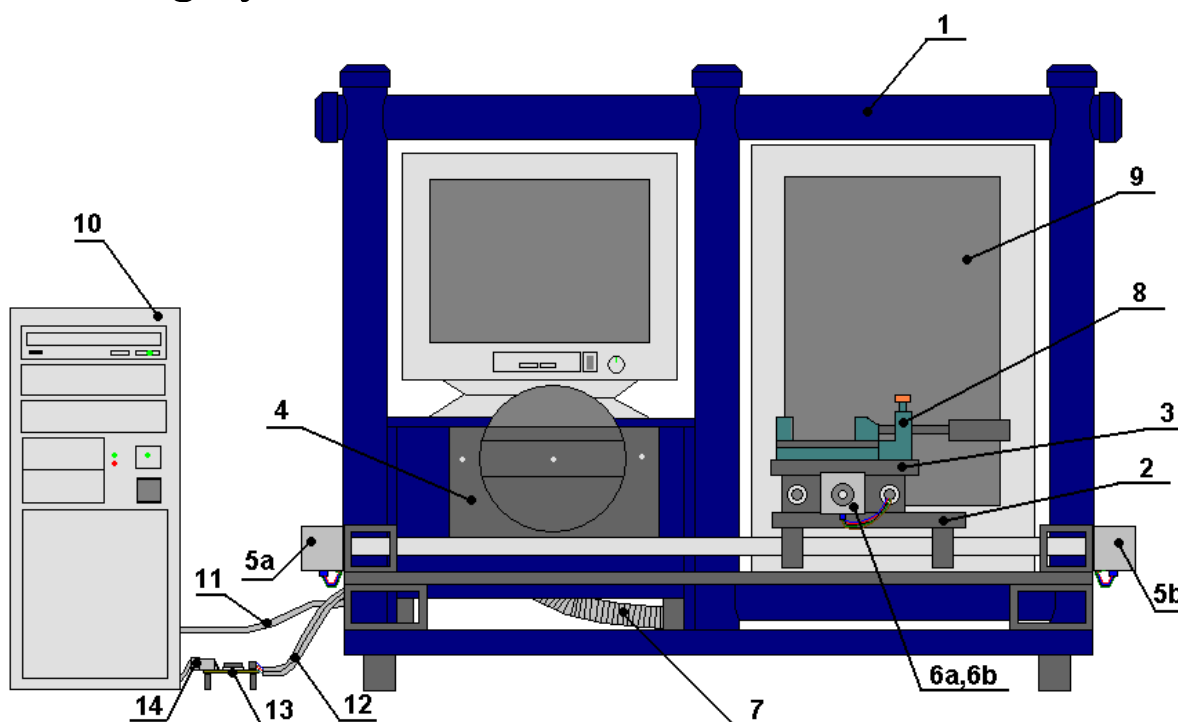
*Przygotowane przez nasz narzędzie badawcze będzie mogło mieć zastosowanie do wyszukiwania przyczyn niepowodzeń leczenia endodontycznego. Z punktu*

widzenia dydaktyki, stanowić będzie interesujące uzupełnienie zajęć z Anatomii Prawidłowej i zajęć przedklinicznych w zakresie stomatologii Zachowawczej i Protetyki.”

## 2. Założenia

Praca urządzenia będzie polegać na cyklicznym zeszlifowywaniu warstw próbek i skanowaniu odsłanianych kolejno przekrojów. Zeskanowane obrazy będą zapisywane w nieskompresowanym formacie TIFF aby uniknąć utraty ważnych szczegółów obrazu. Obrazy będą gromadzone tymczasowo na lokalnym dysku (pojemność 10 GB, około 8 GB na obrazy). Następnie będą przesyłane przez sieć do właściwego komputera – serwera. Zgromadzone obrazy zostaną połączone w całość w plikach modeli wokselowych VOX. Każdy ząb będzie miał swój oddzielny plik VOX. Oddzielne „tify” zostaną wykorzystane do atlasu Internetowego obsługiwane przez aplet Javy. Natomiast pliki VOX umożliwią szybką eksplorację zgromadzonych obrazów i interaktywne wykonywanie przekrojów dowolnie nachyloną płaszczyzną definiowaną w 2D lub 3D (podgląd przestrzenny z wykorzystaniem OpenGL). Jednym z głównych założeń była jak największa minimalizacja kosztów urządzenia, wiążąca się niestety z pewnymi ustępstwami m. in. w konstrukcji napędu posuwu, obniżającymi dokładność i szybkość obróbki.

## 3. Szczegóły techniczne



Rys 3.1 – Schemat stanowiska akwizycji danych

Na rysunku 3.1 jest przedstawiony ogólny schemat stanowiska. W finalnej wersji

zastosowano komputer w obudowie typu desktop, umieszczony i przymocowany w tylnej części maszyny, za skanerem.

Oznaczenia głównych elementów:

1. Główna rama maszyny
2. Ruchomy stolik posuwu wzdłużnego X
3. Ruchomy stolik posuwu poprzecznego Y
4. Wbudowana szlifierka
5. Silniki napędu osi X (a, b)
6. Silniki napędu osi Y (a, b)
7. Rura do odsysania pyłu (podłączona do wodnego odkurzacza z regulacją siły ssania)
8. Zaciskany uchwyt na kostki gipsowe z próbkami
9. Skaner HP
10. Komputer sterujący
11. Przewód USB do skanera
12. Przewody sterujące – zasilające silników
13. Płytki z układami mostków mocy (ostatecznie została umieszczona wewnątrz obudowy komputera sterującego)
14. Przewód LPT – doprowadza sygnały sterujące silnikami

Cechy konstrukcji:

Gabaryty (przybliżone):

Podstawa: 760 x 500 mm

Rama: 800 mm

Użyte materiały:

Płyta wiórowa – podstawa

Aluminium – pozostałe elementy

Napęd:

Cztery unipolarne, 4 - fazowe silniki skokowe Mikroma FA – 15

Liczba pasm uzwojenia: 4

Skok znamionowy:  $1,8^\circ \pm 5\%$

Napięcie: 12 V

Prąd pasma: 160 mA

Rezystancja pasma:  $74 \Omega \pm 10\%$

Indukcyjność pasma:  $48 \text{ mH} \pm 20\%$

Moment trzymający: 50 mNm (min.)

Moment rozruchowy: 25 mNm (min.)

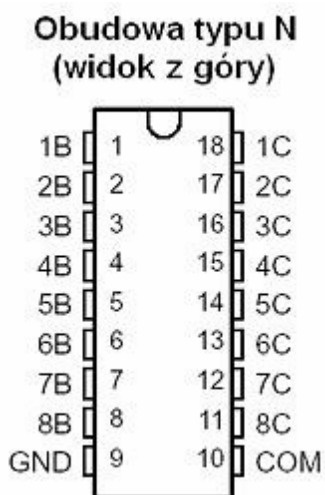
Częstotliwość start – stopowa: 450 Hz (min.)

Mechanizm ruchu liniowego:

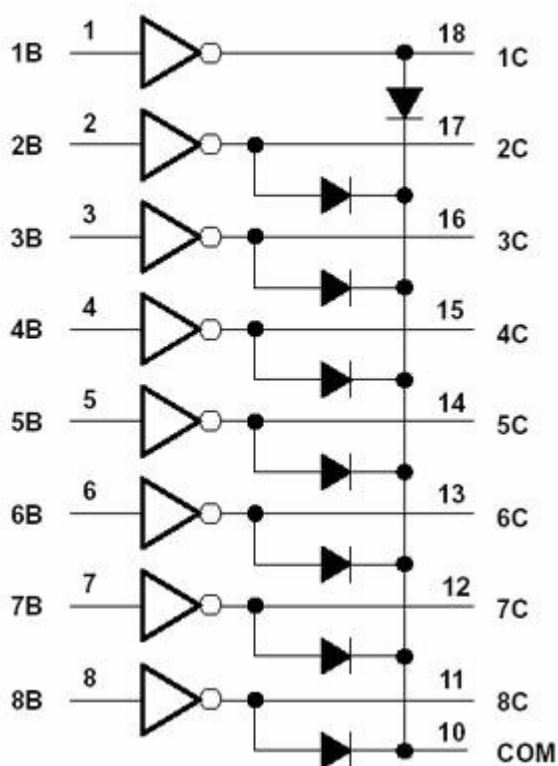
Śruba pociągowa o skoku 0,7 mm (tolerancja nieznana) z gwintem o zarysie trójkątnym

## Sterowanie:

Komputer PC wyposażony w specjalistyczne oprogramowanie sterujące działające pod kontrolą zmodyfikowanej wersji systemu operacyjnego Linux, opartej na dystrybucji Debian GNU/Linux. Sterowanie odbywa się poprzez linie danych D0 – D7 portu równoległego (LPT), sterujące bezpośrednio zasilaniem kolejnych faz uzwojeń silników. Komputer jest połączony z silnikami poprzez separujące mostki mocy TOSHIBA ULN2803A wyposażone w osiem par Darlingtona NPN (rysunki 3.2 i 3.3) zasilanych przez jedno z wyprowadzeń zasilacza komputera, dostarczającego napięcie +5V i +12V.



**diagram logiczny (logika dodatnia)**



Rys 3.2 – Opis wyprowadzeń układu ULN2803A

Rys. 3.3 – Schemat logiczny ULN2803A

\*\*\*

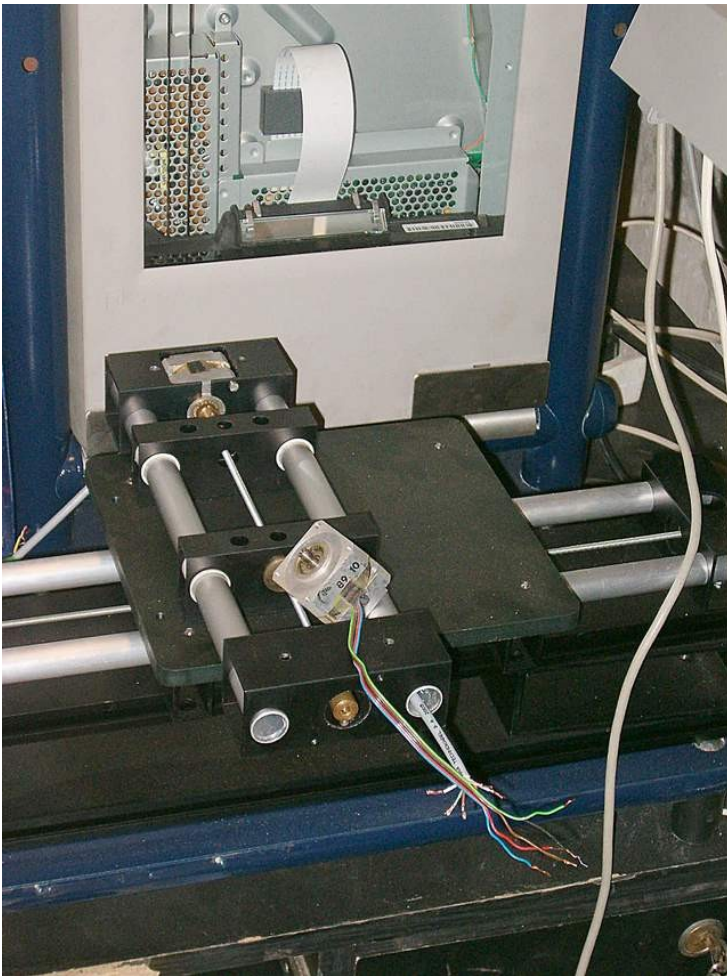
Poniżej znajdują się zdjęcia maszyny zrobione w fazie montażu i testowania:



Rys 3.4 – Ogólny widok maszyny



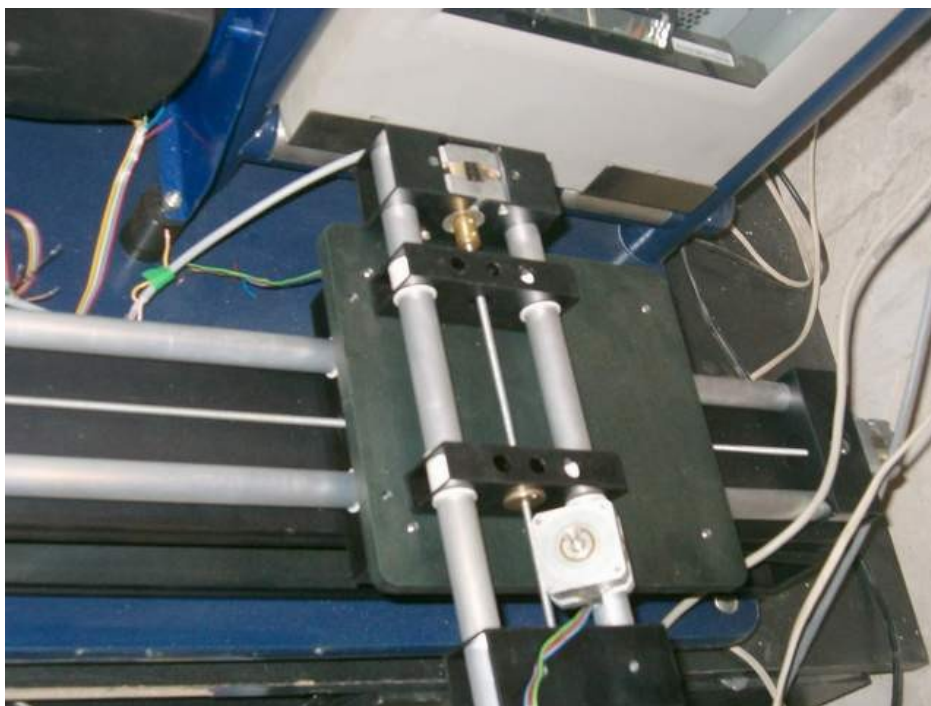
Rys 3.5 – Regulacja prowadnicy Y



Rys 3.6 – Napęd posuwu poprzecznego po zdjęciu blatu stolika Y i zacisku



Rys 3.7 – Komputer sterujący używany do testów i płytka z mostkami mocy



Rys 3.8 – Prowadnica poprzeczna Y



Rys 3.9 – Widok z góry

\*\*\*

Moim zadaniem było uruchomienie gotowej maszyny: odpowiednie podłączenie silników do układu sterującego oraz przygotowanie komputera sterującego wraz ze sterownikami SANE do skanera. Aby tego dokonać, musiałem zapoznać się z bogatą dokumentacją EMC, budową i sposobami

sterowania różnych silników skokowych, z biblioteką SANE oraz z zagadnieniami systemów czasu rzeczywistego (ang. *RTOS – Real Time Operating Systems*) używanymi m. in. w przemyśle do sterowania maszynami jako tzw. systemy wbudowane (ang. *embedded systems*) o krytycznych wymaganiach czasowych i deterministycznym zachowaniu.

Ponieważ silniki nie posiadały dokumentacji wyprowadzeń, identyfikacji przewodów dokonałem przy pomocy pomiarów rezystancji pozwalających upewnić się, który przewód jest przewodem wspólnym dla wszystkich pasm. Pozostałe przewody odpowiadające kolejnym fazom zostały zidentyfikowane metodą prób i błędów poprzez podłączanie ich kolejno do wyprowadzeń sterownika (do testów wykorzystałem dodatkowy, nieużywany silnik) i obserwacji pracy silnika. Silniki osi X i Y zostały parami podłączone do wyjść na płycie z mostkami tranzystorowymi. Zostały użyte dwa wyjścia (4 przewody fazowe z potencjałem ujemnym i jeden z potencjałem +12 V na wyjściu). Silniki dla danej osi zostały podłączone tak, aby wykonywały obroty w przeciwnych kierunkach (zamiana kolejności podłączenia faz), ze względu na pracę na przeciwnych końcach śruby.

Jak już wspomniałem, komputer działa pod kontrola systemu Linux ze zmodyfikowanym jądrem umożliwiającym obsługę zadań czasu rzeczywistego. Zainstalowany system to tzw. BDI (*Brain Dead Install*), zawierający gotowe skompilowane jądro z łąką czasu rzeczywistego, darmowym oprogramowaniem CAD i CAM. Główną część systemu BDI stanowi oprogramowanie EMC (*Enhanced Machine Controller*), będące darmowym, uniwersalnym systemem przeznaczonym do sterowania maszynami technologicznymi, robotami, itp. Projekt EMC jest projektem otwartym (ang. *open source*) i w związku z tym jego kod źródłowy jest powszechnie dostępny dla każdego, kto chciałby dodać samodzielnie jakieś rozszerzenia czy usprawnienia. EMC został napisany w języku C. EMC2 jest rozwijany w C i C++. Wysokopoziomowe funkcje są pisane w C++, a niskopoziomowe podprogramy kontaktujące się bezpośrednio ze sprzętem i API czasu rzeczywistego w C. Do sterowania maszyną została wykorzystana wersja 2 EMC, ze względu na bardziej elastyczną konfigurację poprzez HAL (*Hardware Abstraction Layer*). Wersja 2 jest nadal w ciągłym rozwoju, więc aby jej użyć, trzeba ściągnąć źródła i samodzielnie je skompilować.

# Welcome to LinuxCNC.org

Home for users of the Enhanced Machine Controller -- EMC

a free and powerful PC-based CNC machinery Controller

[EMC - an Overview](#)

[FAQ - frequently asked questions](#)

[EMCKnowledgeBase - a Wiki for EMC](#)

[Get Started With the Brain Dead Install](#)

[EMC - History, News, and Events](#)

[Documentation, Mailing Lists, Links, Archives, & Other Stuff](#)

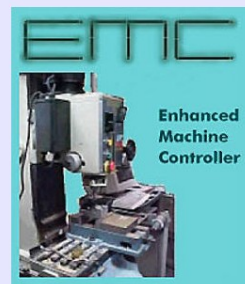
[EMC2 - New Development](#)

[Bug Reports and Feature Requests](#)

[Board of Directors](#)

[About LinuxCNC.org](#)

[EMC and LinuxCNC Supporters](#)



[EMC is in use around the world](#)

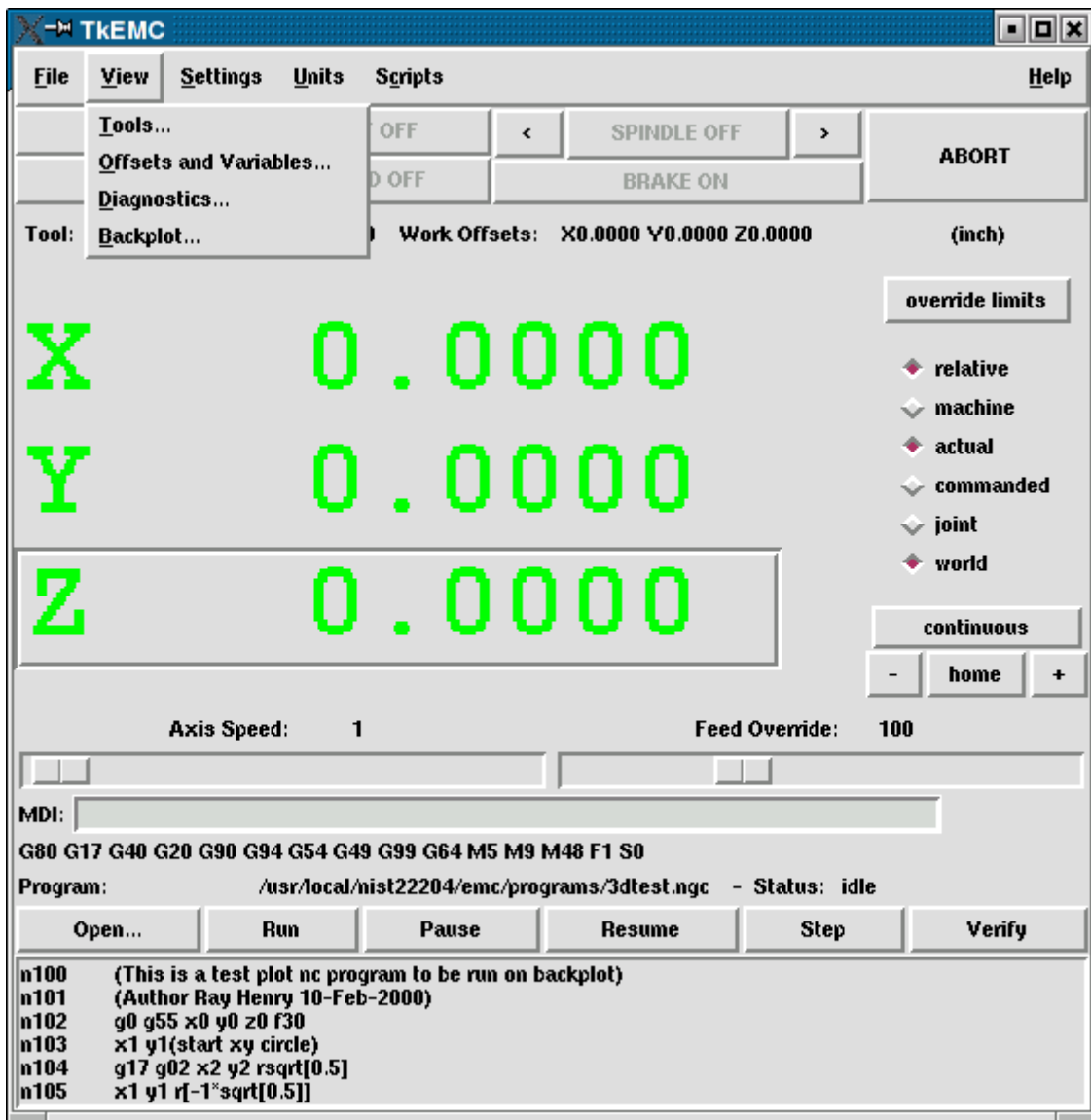


For questions about EMC and integration with Linux, please see our [reading lists](#)

For comments or questions about this web site, please see our [contact info](#)

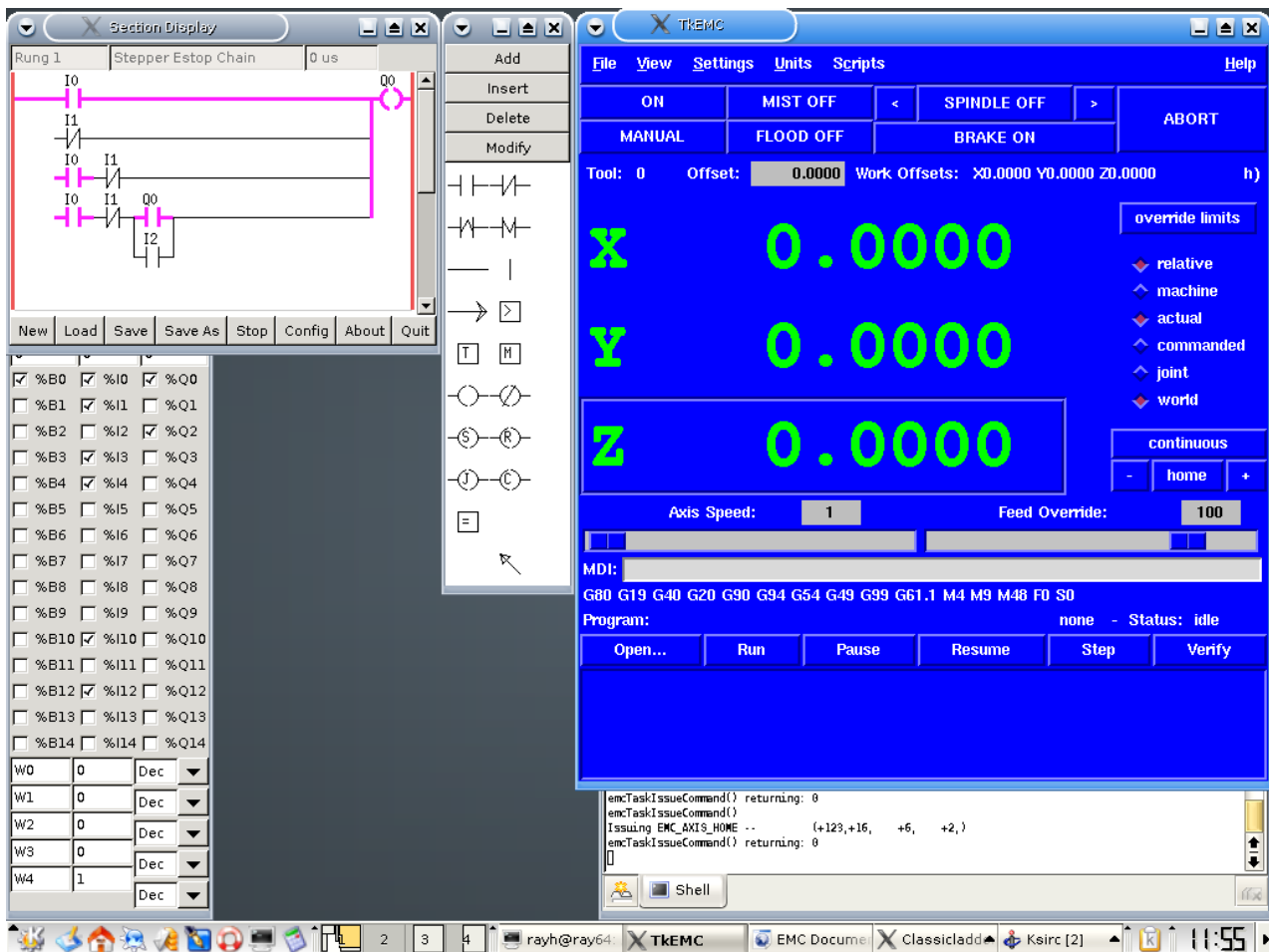
Rys 3.10 – Główna strona projektu EMC [7]

System EMC został opracowany w NIST (*National Institute of Standards and Technology*) w USA ([8]). EMC umożliwia sterowanie poprzez zwykły port równoległy komputera (LPT), z wykorzystaniem różnych algorytmów sterowania uzależnionych od posiadanego sterownika silników skokowych. Obsługuje też karty PCI przeznaczone do sterowania serwonapędami. System składa się z: interpretera kodu numerycznego zgodnego ze standardem RS274, interpolatora, regulatora PID i interfejsów umożliwiających komunikację z różnego rodzaju sprzętem (enkoderami, głowicami narzędziowymi, silnikami skokowymi, serwonapędami). Wszystko to jest realizowane na drodze programowej w zwykłym komputerze PC. Dzięki temu nie trzeba stosować kosztownych, specjalizowanych sterowników CNC. Dodatkowo programowa realizacja wszystkich komponentów sterowania NC jest elastyczna i umożliwia modyfikacje, poprzez dodanie np. własnych funkcji wywoływanych w programie technologicznym. EMC2 współpracuje też z komputerowymi symulacjami sterowników PLC (rysunek 3.12), które można programować w klasycznym języku drabinkowym (*Ladder Diagram*).



Rys 3.11 – Standardowy graficzny interfejs użytkownika TkEMC

Interpreter G – kodu zaimplementowany w EMC nie obsługuje pętli i podprogramów, więc w celu uzyskania cyklicznej obróbki (zeszlifowanie warstwy i dojechanie do skanera) trzeba będzie zastosować tzw. rozwinięcie pętli. Polega to na wygenerowaniu sekwencyjnej listy rozkazów sterujących całym procesem szlifowania i skanowania. Oczywiście ręczne napisanie kodu nie wchodzi w grę. Zostanie napisany pomocniczy program tworzący plik z kodem numerycznym dla zadanych wymiarów kostki i przemieszczenia jej w przestrzeni. Po dojechaniu do tarczy szlifierskiej, uchwyt z kostką powinien się wycofać, aby nie uderzyć w ramę szlifierki, przesunąć naprzeciw skanera i dojechać do jego szyby w celu umożliwienia zeskanowania odsłoniętej warstwy. Następnie powinien znowu się wycofać, aby ominąć ramę, dojechać do tarczy i dosunąć się do niej, aby zeszlifować kolejną warstwę. Opisany powyżej cykl musi zostać powtórzony wymaganą liczbę razy zależną od ilości warstw (ich grubości i wymiarów zęba).



Rys 3.12 – Współpraca EMC z oprogramowaniem emulującym PLC programowanym w LD

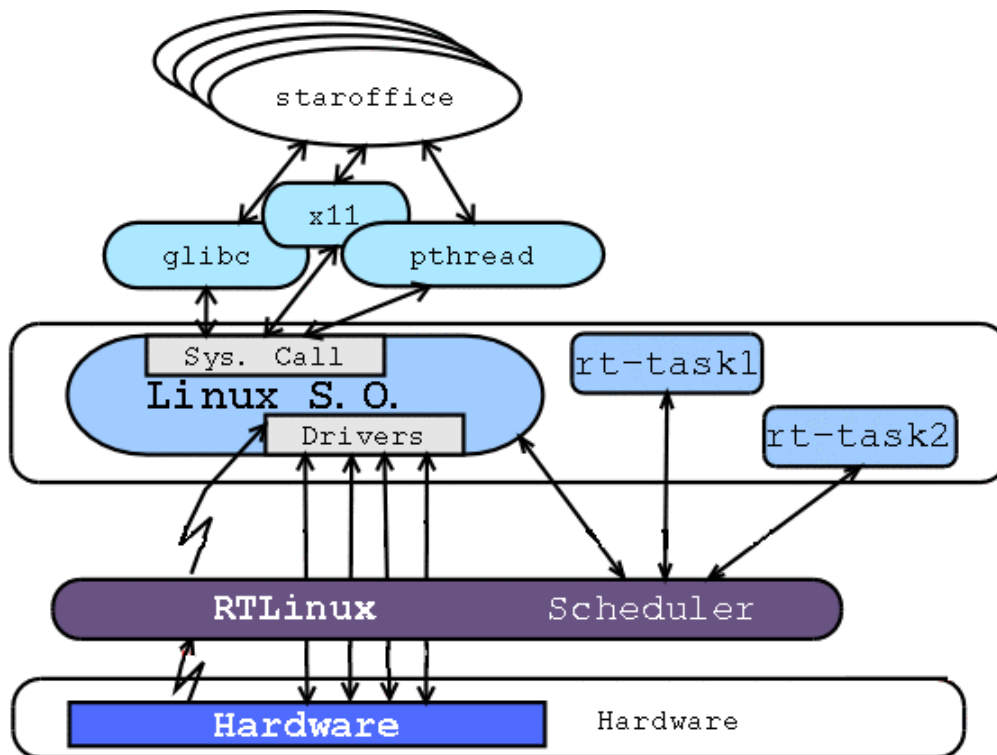
Dzięki koncepcji HAL można tworzyć wirtualne połączenia programowych generatorów sygnałów o dowolnym kształcie i parametrach czasowych z dowolnymi pinami portu LPT lub innymi urządzeniami traktowanymi jako czarne pudełka (ang. *black boxes*), bez znajomości ich wewnętrznej struktury. Pierwsza wersja EMC nie umożliwiała takiej rekonfiguracji połączeń. Właśnie takie podejście do konstrukcji maszyn sterowanych komputerowo, przyświecało twórcom EMC2. Jest to typowa koncepcja mechatronicznego projektowania. W dokumentacji EMC zamiast pojęcia mechatronik użyto określenia integrator, co w pełni odzwierciedla rolę interdyscyplinarnego konstruktora nowoczesnych maszyn, jako osoby łączącej w działającą całość, komponenty wykonane przy wykorzystaniu różnych technik, bez dokładnej znajomości ich wewnętrznej struktury i wykorzystywanych w nich zjawisk fizycznych.

Krytyczny kod EMC, odpowiedzialny za bezpośrednią komunikację z napędami i czujnikami oraz innymi urządzeniami, jest wykonywany jako zadania czasu rzeczywistego pracujące pod kontrolą wbudowanego nanojądra ADEOS (ang. *nanokernel*). Komunikacja z ADEOS-em i korzystanie z mechanizmów synchronizacji i wymiany informacji pomiędzy zadaniami czasu rzeczywistego i resztą zwykłego systemu (semaforami, muteksami, skrzynkami pocztowymi, buforami FIFO) odbywa się przez interfejs RTAI (*Real Time Application Interface*) stworzony na Politechnice Milano ([9], [10]). Bezpośrednia kontrola napędów w

obrabiarkach numerycznych i robotach oraz odczytywanie danych z czujników nie byłyby możliwe w zwykłym systemie operacyjnym (Linux, Windows, Unix) ze względu na optymalizację średniego czasu odpowiedzi i duży rozrzut czasu reakcji na te same żądania. Jest to niedopuszczalne w odpowiedzialnych systemach gdzie czas reakcji musi być nie tylko jak najkrótszy (liczy się jego stosunek względem szybkości zmian stanu kontrolowanego obiektu), ale przede wszystkim przewidywalny. Oznacza to, że twardy system czasu rzeczywistego zapewnia określony maksymalny i przewidywalny czas reakcji na żądania i zewnętrzne zdarzenia oraz może wywoływać periodyczne zadania ze znaną dokładnie częstotliwością. Oparty o 64-bitowy licznik procesora TSC (Time Stamp Counter – dostępny w procesorach począwszy od pierwszego modelu Pentium), uaktualniany z częstotliwością pracy zegara CPU, timer w RTAI mierzy czasy rzędu mikrosekund w przeciwieństwie do zwykłego timera Linuksa gdzie można teoretycznie odmierzać czas z dokładnością do 1 milisekundy i w dodatku nie mamy pewności czy rzeczywiście za każdym razem będzie to milisekunda czy kilka milisekund. Łagodne systemy czasu rzeczywistego stosowane powszechnie w komputerach PC nie zapewniają stałości i przewidywalności. Mają inne algorytmy szeregowania zadań i dopuszczają opóźnienia jednych wątków kosztem innych łącznie z wyłączeniem przerw. Wyróżniane są jeszcze tzw. miękkie systemy czasu rzeczywistego, w których małe opóźnienia reakcji są dopuszczalne, chociaż niezbyt pożądane. Stosowane są głównie w urządzeniach audio-video (nowoczesne kamery cyfrowe, aparaty cyfrowe, dekodery telewizji kablowej i cyfrowej, telewizory, odtwarzacze DVD, itp.) oraz w sprzęcie sieciowym (modemy, routery, firewalle sprzętowe). ADEOS wraz z RTAI nie jest jednak typowym samodzielnym systemem czasu rzeczywistego (jak np: QNX, VxWorks, Nucleus PLUS, Windows CE), a jedynie nakładką umożliwiającą uruchamianie uprzywilejowanych zadań czasu rzeczywistego w pełnym systemie Linux.

Zostało tu zastosowane sprytne rozwiązanie opracowane przez prof. Victora Yodaikena z New Mexico Institute of Technology, twórcę systemu RTLinux ([1], [11]), polegające na wprowadzeniu programowej emulacji kontrolera przerw APIC, która przechwytuje żądania przerw i ich wyłączenie/włączenie wysyłane przez jądro Linuksa oraz używa własnego schedulera. Zasada działania tego systemu jest przedstawiona na rysunku 3.13.

Wymaga to tylko niewielkich modyfikacji kodu źródłowego oryginalnego jądra Linuksa. Po uruchomieniu rozszerzenia RTAI, ADEOS przejmuje kontrolę nad procesorem i udostępnia własnego planistę czasu rzeczywistego (ang. *scheduler*), a główne jądro Linuksa jest traktowane jako zadanie tła o najniższym priorytecie (ang. *idle task*). Dzięki temu wszystkie ważne zadania są wykonywane w zadanym czasie i z wymaganą częstotliwością, a gdy wystarczy wolnego czasu procesora, jest on przydzielany jądro Linuksa. Na wolnych komputerach może to powodować widoczne spowolnienie działania systemu, jednak mamy gwarancję, że priorytetowe zadania są wykonywane bez przeszkód.



Rys 3.13 – Koncepcja działania systemów RTLinux i RTAI

System EMC2 jest konfigurowany poprzez pliki tekstowe zawierające wszystkie ustawienia dotyczące rodzaju sterowania, parametrów geometrycznych napędów posuwu, wyglądu interfejsu graficznego, itp. W celu dostosowania go do zbudowanej dla Akademii Medycznej maszyny musiałem zmodyfikować istniejące lub dodać nowe pliki konfiguracyjne oraz zmodyfikować kod źródłowy EMC2, w celu umożliwienia korzystania z funkcji użytkownika wywoływanych w programie technologicznym. Funkcje te będą potrzebne do wywoływania programu skanującego po zeszlifowaniu każdej warstwy próbki.

W celu wygodnego uruchamiania i kompilacji EMC2, stworzyłem dwa pliki ze skryptami powłoki:

```
run_emc2.sh
```

```
#!/bin/bash
```

```
# Shell script for automatic running of EMC2 with proper initialization file.
```

```
# Created by Artur Kozubski - 17.02.2006
```

```
clear
```

```
scripts/emc xuntech_stepper/xun_stepper.ini
```

Powyższy skrypt czyści konsolę i uruchamia EMC z argumentem wskazującym na mój własny plik z konfiguracją *xun\_stepper.ini*. Plik ten zawiera podstawowe ustawienia dotyczące używanego sterownika napędów, informacje o używanych osiach i ich ilości (w tym przypadku są to dwie osie X i Y), skali (parametr związany

ze skokiem śruby pociągowej i skokiem znamionowym silników), jednostkach, itp.

\*\*\*

*make\_emc.sh*

```
#!/bin/bash
# Shell script for automatic EMC2 compilation.
# Created by Artur Kozubski - 17.02.2006
clear
cd src
make clean
make KDIR=/usr/src/linux-headers-2.6.12.6-rtai/
sudo make setuid
cd ..
```

Skrypt wykonujący kompilację EMC2.

\*\*\*

*xun\_stepper.ini* (fragmenty)

```
# Medical University of Warsaw
# EMC2 configuration file for stepper motors - Xuntech Medical Systems
#-----
# Warsaw University of Technology - Faculty of Mechatronics
# Created by Artur Kozubski (17.02.2006)
#-----
```

```
# EMC controller parameters for generic controller. Make these what you need
# for your system.
```

```
# General note: Comments can either be preceded with a # or ; - either is
# acceptable, although # is in keeping with most linux config files.
```

```
# Settings with a + at the front of the comment are likely needed to get
# changed by the user.
```

```
# Settings with a - at the front are highly unneeded to be changed
```

```
#####
```

```
# General section
```

```
#####
```

```
[EMC]
```

```
##- Version of this INI file
```

```
VERSION =          $Revision: 1.3 $
```

```

(...)
#####
# Motion control section
#####
[EMCMOT]

#- Name of the motion controller to use (only one exists for nontrivkins)
EMCMOT =          motmod

#- Key for real OS shared memory, e.g., for simulated motion
SHMEM_KEY =       111

#- Timeout for comm to emcmot, in seconds
COMM_TIMEOUT =    1.0

#- Interval between tries to emcmot, in seconds
COMM_WAIT =       0.010

#+ Base task period, in secs - this is the fastest thread in the machine
BASE_PERIOD =     0.000050
#- Servo task period, in secs - will be rounded to an int multiple of
BASE_PERIOD
SERVO_PERIOD =    0.001000
#- Trajectory Planner task period, in secs - will be rounded to an
# integer multiple of SERVO_PERIOD
TRAJ_PERIOD =     0.010000

#####
# Hardware Abstraction Layer section
#####
[HAL]

# list of hal config files to run through halcmd
#+ files are executed in the order in which they appear
HALFILE =         xun_stepper.hal
HALFILE =         xun_pinout.hal

#- list of halcmd commands to execute
# commands are executed in the order in which they appear
#HALCMD =         save neta

#####
# Trajectory planner section
#####

```

**[TRAJ]**

**#+ machine specific settings**

**AXES = 2**  
**COORDINATES = X Y**  
**HOME = 0 0**  
**LINEAR\_UNITS = 1.0**  
**ANGULAR\_UNITS = 1.0**  
**CYCLE\_TIME = 0.010**  
**DEFAULT\_VELOCITY = 0.424**  
**MAX\_VELOCITY = 30.48**  
**DEFAULT\_ACCELERATION = 300.0**  
**MAX\_ACCELERATION = 500.0**  
**PROBE\_INDEX = 0**  
**PROBE\_POLARITY = 1**

#####

**# Axes sections**

#####

**#+ First axis**

**[AXIS\_0]**

**TYPE = LINEAR**  
**UNITS = 1.0**  
**HOME = 0.000**  
**MAX\_VELOCITY = 30.48**  
**MAX\_ACCELERATION = 500.0**  
**STEPGEN\_MAXVEL = 35.00**  
**STEPGEN\_MAXACCEL = 520.0**  
**BACKLASH = 0.000**  
**CYCLE\_TIME = 0.001000**  
**INPUT\_SCALE = 200 0**  
**OUTPUT\_SCALE = 1.000 0.000**  
**MIN\_LIMIT = -1000.0**  
**MAX\_LIMIT = 1000.0**  
**FERROR = 1.270**  
**MIN\_FERROR = 0.254**  
**HOME\_OFFSET = 0.0**  
**HOME\_SEARCH\_VEL = 0.0**  
**HOME\_LATCH\_VEL = 0.0**  
**HOME\_USE\_INDEX = NO**  
**HOME\_IGNORE\_LIMITS = NO**

**#+ Second axis**

**[AXIS\_1]**

```

TYPE =                LINEAR
UNITS =               1.0
HOME =                0.000
MAX_VELOCITY =       30.48
MAX_ACCELERATION =   500.0
STEPGEN_MAXVEL =     35.00
STEPGEN_MAXACCEL =   520.0
BACKLASH = 0.000
CYCLE_TIME =         0.001000
INPUT_SCALE =        200 0
OUTPUT_SCALE = 1.000 0.000
MIN_LIMIT =          -1000.0
MAX_LIMIT =          1000.0
FERROR = 1.270
MIN_FERROR = 0.254
HOME_OFFSET =        0.0
HOME_SEARCH_VEL =    0.0
HOME_LATCH_VEL =     0.0
HOME_USE_INDEX =     NO
HOME_IGNORE_LIMITS = NO

```

(...)

Plik z głównymi ustawieniami EMC.

\*\*\*

*xun\_stepper.hal*

```

# Medical University of Warsaw
# EMC2 configuration file for stepper motors - Xuntech Medical Systems
#-----
# Warsaw University of Technology - Faculty of Mechatronics
# Created by Artur Kozubski (17.02.2006)
#-----

# core HAL config file for steppers

# first load the stepper module
loadrt stepgen step_type=6,6

# hook its functions to realtime threads
addf stepgen.capture-position servo-thread 1
addf stepgen.update-freq servo-thread -1

```

**addf stepgen.make-pulses base-thread -1**

**# connect position commands from motion module to step generator**

**newsig Xpos-cmd float**

**linksp Xpos-cmd <= axis.0.motor-pos-cmd**

**linksp Xpos-cmd => stepgen.0.position-cmd**

**newsig Ypos-cmd float**

**linksp Ypos-cmd <= axis.1.motor-pos-cmd**

**linksp Ypos-cmd => stepgen.1.position-cmd**

**# connect position feedback from step generators**

**# to motion module**

**newsig Xpos-fb float**

**linksp Xpos-fb <= stepgen.0.position-fb**

**linksp Xpos-fb => axis.0.motor-pos-fb**

**newsig Ypos-fb float**

**linksp Ypos-fb <= stepgen.1.position-fb**

**linksp Ypos-fb => axis.1.motor-pos-fb**

**# connect enable signals for step generators**

**newsig Xen bit**

**linksp Xen <= axis.0.amp-enable-out**

**linksp Xen => stepgen.0.enable**

**newsig Yen bit**

**linksp Yen <= axis.1.amp-enable-out**

**linksp Yen => stepgen.1.enable**

**# connect signals to step pulse generator outputs**

**newsig XstepA bit**

**newsig XstepB bit**

**newsig XstepC bit**

**newsig XstepD bit**

**newsig YstepA bit**

**newsig YstepB bit**

**newsig YstepC bit**

**newsig YstepD bit**

**linkps stepgen.0.phase-A XstepA**

**linkps stepgen.0.phase-B XstepB**

**linkps stepgen.0.phase-C XstepC**

**linkps stepgen.0.phase-D XstepD**

**linkps stepgen.1.phase-A YstepA**

**linkps stepgen.1.phase-B YstepB**

**linkps stepgen.1.phase-C YstepC**

**linkps stepgen.1.phase-D YstepD**

```

# set stepgen module scaling - get values from ini file
setp stepgen.0.position-scale [AXIS_0]INPUT_SCALE
setp stepgen.1.position-scale [AXIS_1]INPUT_SCALE

# set stepgen module velocity limits - get values from ini file
setp stepgen.0.maxvel [AXIS_0]STEPGEN_MAXVEL
setp stepgen.1.maxvel [AXIS_1]STEPGEN_MAXVEL

# set stepgen module accel limits - get values from ini file
setp stepgen.0.maxaccel [AXIS_0]STEPGEN_MAXACCEL
setp stepgen.1.maxaccel [AXIS_1]STEPGEN_MAXACCEL

# load realtime portion of scope, just to have it handy
loadrt scope_rt

```

Plik zawierający konfigurację sterownika silników skokowych. Wykorzystuje mechanizmy HAL. W sekcji **stepgen** na początku pliku są zdefiniowane dwa generatory sygnału prostokątnego typu 6, ładowane jako moduły jądra pracujące w czasie rzeczywistym (*kernel modules*). Odpowiada to sterowaniu pełnokrokowemu, kiedy zasilane są jednocześnie dwie fazy silnika, a wirnik przyjmuje położenia pośrednie pomiędzy zębami aktywnych faz. Skok jest taki sam jak przy zasilaniu jednej fazy. Zęby zatrzymują się pomiędzy zębami stojana zamiast naprzeciwko ich jak przy sterowaniu falowym (przy zasilaniu tylko jednej fazy na krok). Sterowanie pełnokrokowe zapewnia większy moment, gdyż dla silników unipolarnych jest wykorzystywane 50% uzwojenia przy każdym kroku, natomiast przy sterowaniu falowym tylko 25% uzwojenia. Silniki bipolarne mają odpowiednio wyższe użycie uzwojenia i w związku z tym większy moment, ale ich sterowanie jest bardziej złożone ze względu na większą ilość przewodów i konieczność przełączania kierunku przepływu prądu przez uzwojenia. Po drugie nie miałem wpływu na wybór użytych silników, ponieważ dostałem już zmontowaną maszynę. Początkowo były użyte generatory typu 5 (sterowanie falowe), co objawiało się nierówną pracą silników i dużym ograniczeniem ich prędkości obrotowej oraz blokowaniem się napędów wraz z „gubieniem” kroków. Zmiana sterowania na pełnokrokowe znacznie poprawiła parametry napędu. Dalszą poprawę, bez zmiany silników, można by było uzyskać przez zastosowanie forsowania prądu w uzwojeniach za pomocą kluczowania prądowego. Jednak wiązałoby się to ze zmianą elektroniki sterującej. W obecnej wersji są to proste wzmacniacze powielające pięciowoltowe sygnały z portu LPT na sygnały dwunastowoltowe służące doysterowania silników.

W dalszej części pliku są zdefiniowane połączenia poszczególnych faz (A, B, C, D - sterowanie czterofazowe) do odpowiednich sygnałów z generatorów.

\*\*\*

*xun\_pinout.hal*

```

# Medical University of Warsaw
# EMC2 configuration file for LPT parallel port - Xuntech Medical Systems
#-----
# Warsaw University of Technology - Faculty of Mechatronics
# Created by Artur Kozubski (17.02.2006)
#-----

# standard pinout config file for 2-axis steppers
# using a parport for I/O
#
# first load the parport driver
loadrt hal_parport cfg="0x0378"
#
# next connect the parport functions to threads
addf parport.0.read base-thread 1
addf parport.0.write base-thread -1
#
# finally connect physical pins to the signals
linksp XstepA parport.0.pin-02-out
linksp XstepB parport.0.pin-03-out
linksp XstepC parport.0.pin-04-out
linksp XstepD parport.0.pin-05-out
linksp YstepA parport.0.pin-06-out
linksp YstepB parport.0.pin-07-out
linksp YstepC parport.0.pin-08-out
linksp YstepD parport.0.pin-09-out

# create a signal for the estop loopback
linkpp iocontrol.0.user-enable-out iocontrol.0.emc-enable-in

```

Powyższy plik zawiera opis połączeń sygnałów sterujących poszczególnymi fazami z fizycznymi pinami portu równoległego LPT. Zawiera też adres portu I/O pod jakim ma się odbywać komunikacja ze złączem równoległym (378h).

\*\*\*

```

(...)
/*! \todo Check that user defined custom M codes are supported.
      These are M100-199 with optional P & Q parameters.
*/
/*! \todo Another #if 0 */
#if 1
/* *** Code modified by Artur Kozubski - 17.02.2006 *** */
/*! \todo FIX-ME Impliment these at a later stage... */
/* user-defined M codes */
for (index = 100; index < 200; index++) {
  if (block->m_modes[index] == index) {
    if (USER_DEFINED_FUNCTION[index - 100] != 0) {
      (*(USER_DEFINED_FUNCTION[index - 100])) (index - 100,
                                                block->p_number,
                                                block->q_number);
    } else {
      CHK(1, NCE_UNKNOWN_M_CODE_USED);
    }
  }
}
}
#endif
return INTERP_OK;
}
(...)

```

Fragment pliku kodu źródłowego interpretera kodu numerycznego odpowiedzialny za wywoływanie funkcji użytkownika. Kod był nieaktywny, co nie było nigdzie udokumentowane, więc musiałem zapoznać się ze źródłami EMC2 i znaleźć fragment odpowiedzialny za przetwarzanie funkcji specjalnych M. W dokumentacji jest informacja, że funkcje specjalne o numerach M100 – M199 są zarezerwowane dla funkcji użytkownika. Oficjalny standard przemysłowy nie wykorzystuje tak wysokich numerów dla funkcji M, więc można bezpiecznie użyć tych numerów bez naruszania zgodności ze standardem. Gdy interpreter napotka wywołanie funkcji M100 – M199 to szuka programu lub skryptu o takiej nazwie w katalogu zdefiniowanym w pliku konfiguracyjnym (tu *xuntech\_stepper.ini*) i uruchamia go. Do zewnętrznego programu lub skryptu można przekazać dwa parametry liczbowe. Z poziomu programu technologicznego są to zmienne P i Q. Przykładowo rozkaz: M101 P123.5 Q454.1 wywoła skrypt lub program o nazwie M101 z parametrami 123.5 i 454.1. Rozszerzenie jest ignorowane, dlatego pliki wykonywalne mogą mieć dowolne rozszerzenie lub go nie posiadać.

\*\*\*

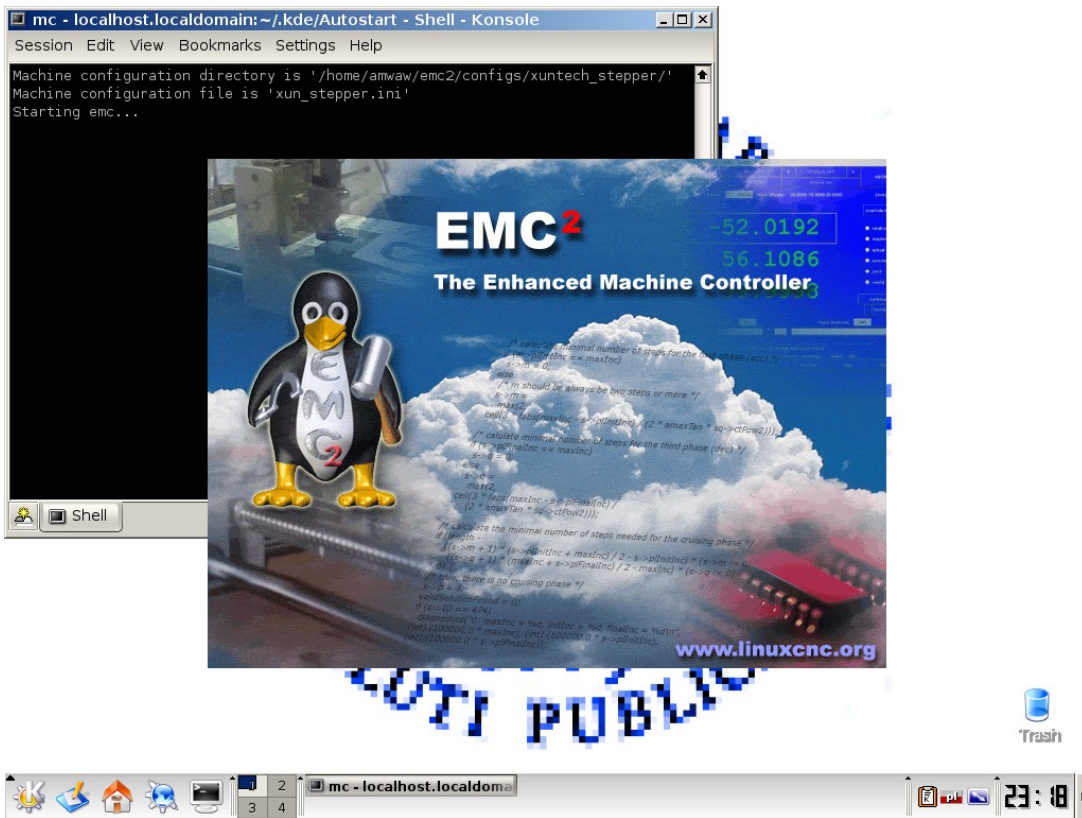
*start\_emc2.sh*

```
#!/bin/bash
# Shell script for KDE autostart of EMC2.
# Created by Artur Kozubski - 17.02.2006
cd ~/emc2
~/emc2/run_emc2.sh
```

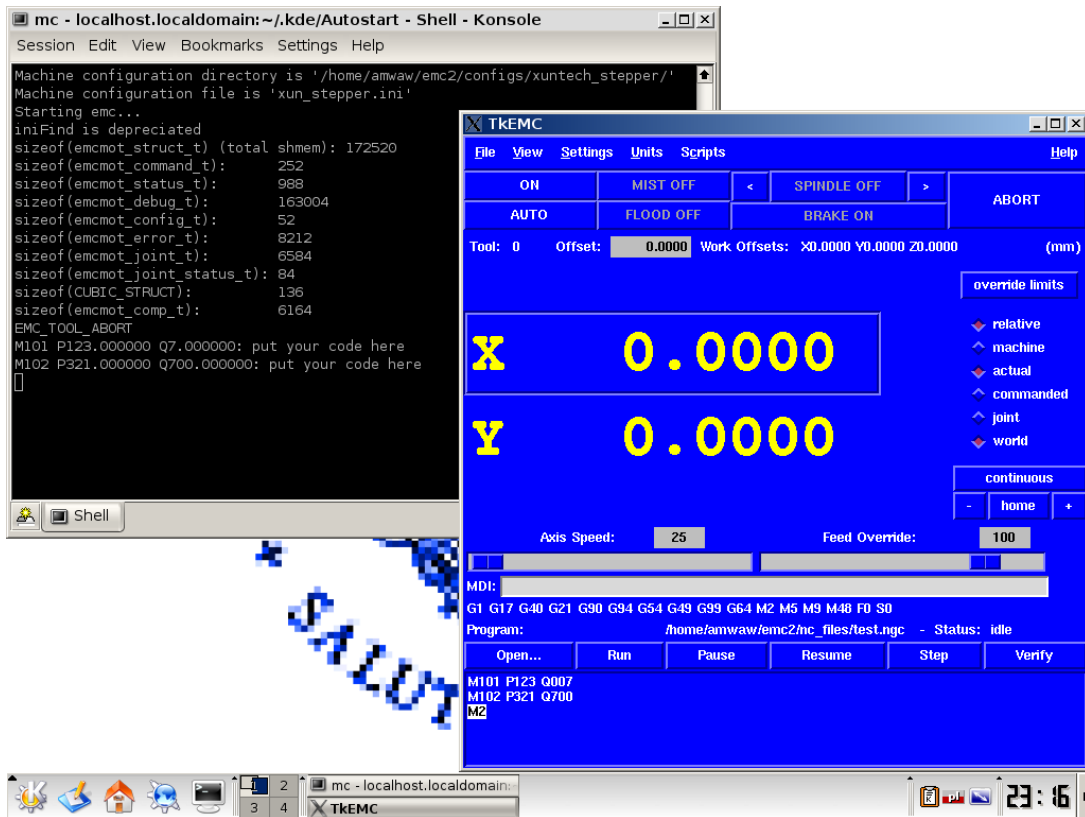
Plik ze skryptem automatycznie uruchamiającym interfejs EMC. Został on umieszczony w katalogu autostartu środowiska KDE. Dzięki temu bezpośrednio po włączeniu komputera uruchamia się na pełnym ekranie GUI EMC bez ingerencji użytkownika i komputer zachowuje się jak zwykły panel operatorski maszyny CNC. Upraszcza to uruchomienie oprogramowania sterującego. Wystarczy włączyć komputer i wszystko uruchomi się automatycznie.



Rys 3.14 – Gotowy system przystosowany do pracy na Akademii Medycznej



Rys 3.15 – Automatyczne uruchamianie EMC2 (ekran powitalny)



Rys 3.16 – Testowanie funkcji użytkownika

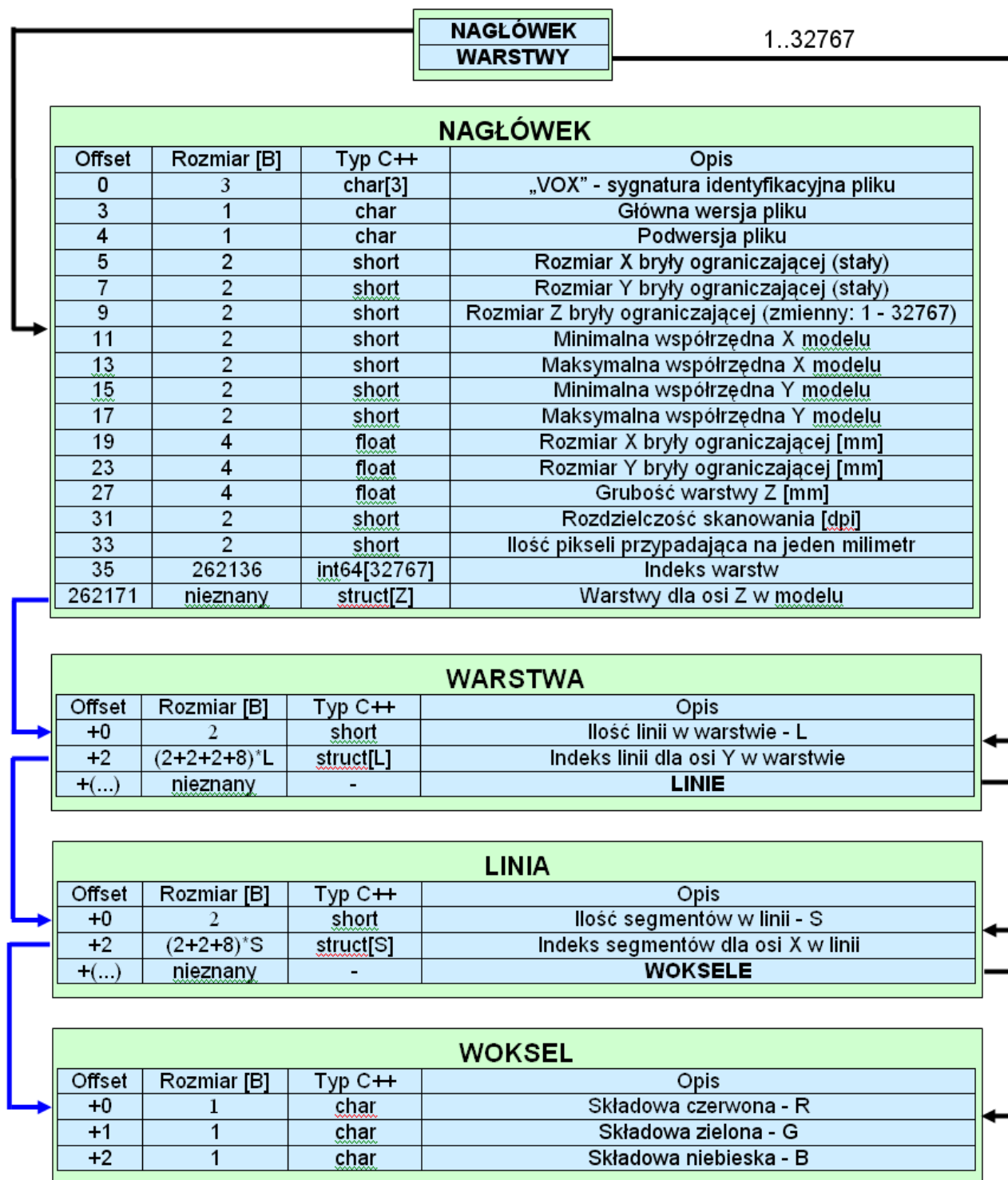
Maszyna została przewieziona do Zakładu Stomatologii Zachowawczej AM przy ul.

Miodowej 18 w poniedziałek 20.02.2006. Będzie tam przeze mnie testowana i przygotowywana do akwizycji danych graficznych.

W międzyczasie opracowałem format pliku VOX (od *VOXel* – *Volumetric pixEL*) przedstawiony na poniższym schemacie:

## STRUKTURA PLIKU W FORMACIE VOX

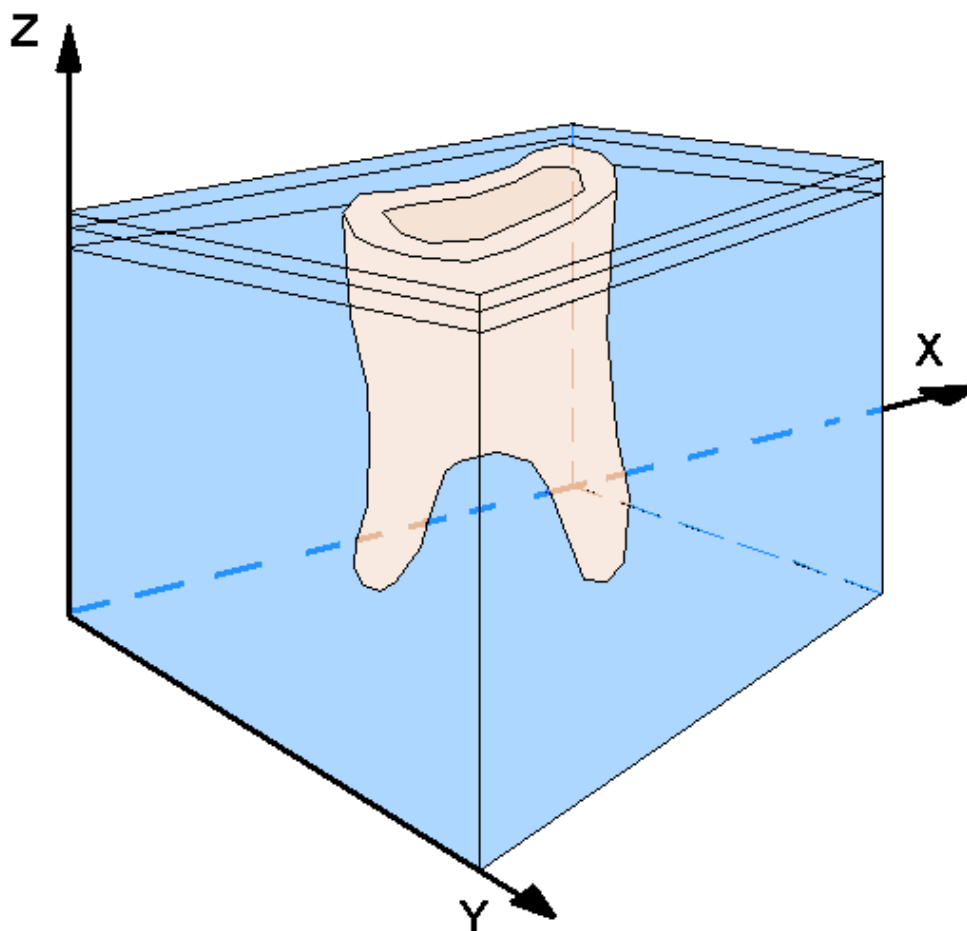
### SCHEMAT OGÓLNY



Rys 3.17 – Schemat formatu VOX

Format przewiduje maksymalnie 2048 warstw wzdłuż osi Z, o dowolnych wymiarach XY. Plik jest podzielony na warstwy, a te z kolei na linie. Linie są podzielone na segmenty. Najmniejszą jednostką modularną jest woksel wchodzący w skład segmentu, reprezentujący sześcian w kolorze zdefiniowanym przez 8-bitowe składowe RGB (24 bpp – True Color) umieszczony w układzie współrzędnych modelu (lokalnym). Wszystkie jednostki podziałowe są indeksowane w celu zapewnienia szybkiego przeglądania pliku i dostępu do dowolnej linii bez długotrwałego przeszukiwania całości pliku. Będzie to miało znaczenie podczas wykonywania interaktywnych przekrojów po dowolnym kącie.

Na rysunku 3.18 został pokazany układ współrzędnych modelu wraz z otaczającym go gipsem (kostką gipsową). Algorytm przetwarzający przekroje będzie wyznaczał dla każdej warstwy (przekroju) skrajne współrzędne X i Y, co umożliwi wyznaczenie centralnej osi zęba. Układ współrzędnych zaczyna się w jednym z narożników prostopadłościanu obejmującego ząb (bryły ograniczającej). Jest on z góry zdefiniowany na stałe, podczas tworzenia pliku VOX. W pliku zostaną zapisane tylko istotne informacje, tzn. bez otaczającego tła, które zostanie odrzucone przez



Rys 3.18 – Orientacja modelu w lokalnym układzie współrzędnych (LUW) z uwzględnieniem wymiarów kostki gipsu stomatologicznego

filtr graficzny, co znacznie zredukuje rozmiar pliku i zwiększy szybkość przetwarzania danych modelu.

## 4. Podsumowanie

Drugą pracę przejściową poświęciłem na uruchomienie maszyny, czyli na część sprzętową projektu i przystosowanie istniejącego oprogramowania wraz z drobnymi modyfikacjami, służącego do sterowania maszyną. Zaprojektowałem też format pliku przechowującego wokselowe modele zębów oraz algorytm przetwarzania i filtrowania obrazów w celu usuwania niepotrzebnych części obrazu (rodzaj prostej kompresji bezstratnej). Algorytm przetestowałem w prowizorycznym programie napisanym w C w środowisku MS Visual C++. Program jest tak chaotyczny, że nie zamieszam tu jego kodu, gdyż jest to jedynie próbka czy mój pomysł da się zrealizować. Z testów wynika, że algorytm działa poprawnie. Filtruje zakłócenia, odrzuca tło o podanym kolorze (nie musi być jednolite, ponieważ można ustawić tolerancję dla barwy, a porównanie następuje w modelu barw HSV a nie RGB – porównywanie składowej barwy H, z pominięciem nasycenia S i jasności V). Działa też wykrywanie krawędzi i eksport do przestrzennego modelu dla OpenGL. Szczegóły zostaną przedstawione w sprawozdaniu z pracy dyplomowej. Pozostaje wykonanie specjalnego oprogramowania o czym poniżej.

Trzeba ustawić jeszcze parametry kinematyczne i geometryczne, tzn. maksymalne przyspieszenia i prędkości posuwu, skalę jednostek przemieszczeń (na podstawie skoku śruby i skoku nominalnego silników), aby sterownik EMC wiedział jakie rzeczywiste przemieszczenie odpowiada jednemu impulsowi generatora sterującego silnikami. Trzeba stworzyć oprogramowanie automatycznie generujące programy technologiczne wykonujące pełny cykl szlifowania i skanowania oraz oprogramowanie do obróbki zeskanowanych przekrojów (budowa modeli wokselowych zapisanych w plikach VOX, wizualizacja 3D w OpenGL i Internecie). Wymienione oprogramowanie zostanie napisane w ramach pracy magisterskiej, będącej kontynuacją tej pracy przejściowej.

## 5. Literatura

1. Kazimierz Lal, Tomasz Rak, Krzysztof Orkisz: RTLinux system czasu rzeczywistego. Helion
2. Piotr Szymczyk: Systemy operacyjne czasu rzeczywistego. Wydawnictwo AGH
3. Bodo Heimann, Wilfried Gerth, Karl Popp: Mechatronika - Komponenty, metody, przykłady. PWN
4. Jan Kosmol: Serwonapędy obrabiarek sterowanych numerycznie. WNT
5. Jan Kosmol: Automatyzacja obrabiarek i obróbki skrawaniem. WNT
6. Ryszard Tadeusiewicz, Przemysław Korohoda: Komputerowa analiza i przetwarzanie obrazów. Wydawnictwo Fundacji Postępu Telekomunikacji
7. <http://www.linuxcnc.org>
8. <http://www.nist.gov>
9. <https://www.rtai.org>
10. <http://www.aero.polimi.it/~rtai>
11. <http://www.fsmlabs.com>