Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

Control System for CNC Cutting Machine

Bc. Filip Měšťánek

Supervisor: Ing. Petr Felkel, Ph.D.
Field of study: Open informatics
Subfield: Computer graphics and interaction
May 2017

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačové grafiky a interakce

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Filip Měšťánek**

Studijní program: Otevřená informatika
Obor: Počítačová grafika a interakce

Název tématu: **Řídicí software CNC Stroje**

Pokyny pro vypracování:

Proveďte rešerši existujících řídicích systémů pro CNC řezací stroje z pohledu uživatelského rozhraní a plánování trajektorie stroje a stanovte požadavky na vytvářený systém pro konkrétní typ CNC stroje. Navrhněte a implementujte řídicí systém CNC řezacího stroje (jak uživatelské rozhraní, tak výkonné jádro běžící na platformě reálného času) a funkce pro interpretaci NC souborů ve standardu ISO. Výkonné jádro bude napojeno na framework TG Motion obsluhující pohyby servopohonů. Systém vytvářejte jako rozšiřitelný pomocí modulů, interakci implementujte se zřetelem na odolnost rozhraní proti nechtěnému poškození stroje obsluhou. Řídicí systém bude mít možnost manuálního ovládání stroje. Součástí bude grafické okno zobrazující aktuální dění na stroji (nahraný NC soubor, trajektorie hlavy, apod). Systém důkladně otestujte alespoň na pěti různých trajektoriích, rozhraní podrobte uživatelským testům.

Seznam odborné literatury:

Dokumentace frameworku TG Drives pro ovládání servopohonů, TG Motion, 2016

Vedoucí: Ing. Petr Felkel, Ph.D.

Platnost zadání: do konce zimního semestru 2017/2018

[illegible]

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
-4-

L.S.

[illegible]

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 8. 3. 2016

Acknowledgements

I would like to express my deep gratitude to my supervisor Ing. Petr Felkel, Ph.D., who helped me extensively with writing the thesis, provided me with valuable insights and was a great support during the development. Another thanks goes to Petr Kretík from the PTV company, who regularly reviewed and consulted the application with me from the technical side. Finally, I want to thank all who participated during the inspection and testing of the interface, especially Silvestr Rovný who supplied me with many ideas.

Declaration

Abstract

This thesis describes the development cycle of a control system for CNC cutting machines. It describes all phases of a system development, from the analysis of current control systems and identification of their flaws, the choice of architecture and appropriate technologies, across design of the user interface to the testing phase. It focuses on the design of the user interface, which was identified as the main deficiency of current systems. The developed system uses TG Motion Framework, which realizes communication with hardware peripherals.

Keywords: control system, CNC, high pressure water-jet

Supervisor: Ing. Petr Felkel, Ph.D.

Abstrakt

Tato diplomová práce popisuje cyklus vývoje řídícího systému pro CNC řezací stroje. Práce se zabývá všemi fázemi vývoje systému od analýzy stávajících řídících systémů a identifikací jejich nedostatků, výběru architektury a vhodných technologií, přes návrh uživatelského rozhraní až po fázi testování. Důraz byl kladen na návrh uživatelského rozhraní, které bylo identifikováno jako hlavní nedostatek stávajících systémů. Vyvinutý systém využívá TG Motion Framework, který realizuje komunikaci s koncovými periferiemi.

Klíčová slova: řídící systém, CNC, vysokotlaký paprsek

Překlad názvu: Řídící Software CNC Stroje

Contents

Figures

Chapter 1
Introduction

From the beginning of times, people have looked for ways to make their work easier. Beginning with simple tools which evolved into complex machines during Industrial Revolution. Manufacturing became faster and prices were dropping. The machines were driven mostly by gears and cams. But the true revolution began with the birth of computers during 50s. This led to creation of computer numerical control (CNC) machines, machines driven by computers, which were far more superior to their predecessors. They were universal, could run 24/7, and produced almost identical products.

CNC Machines fall into many categories. There are milling, cutting, drilling, bending machines and many more. They differ in principles and functionality but share the aim of refining materials. As opposed to manually operated machines, CNC machines are controlled by control systems. Control system is a special software whose task is to control individual parts of the machine like motors or valves in order to achieve the desired result. The system itself is commanded either manually by an operator or through a specialized code, mostly the G-code. G-code is a semi-standardized language for CNC Machines whose instructions describe motions and actions of the machine.

Many control systems originate from 80s or 90s which is very obvious from their user interfaces. CNC machines are getting faster and more precise but from the functional side not much have changed. Therefore, there isn't enough driving force to innovate the interfaces because the older ones still suffice. From a functional point, there is nothing bad with them. It is the user interface, which tends to be clumsy and unfriendly. UIs weren't a priority back then (this can be seen in every subfield of engineering) as there was much less computing power. But situation today is completely different, UI technologies and principles made a huge progress, user demands drastically increased. Goal of this thesis is to develop a functional modern control system for a laser and water-jet CNC cutting machine. When developing the control system, I will use modern techniques to create a functional, robust and user friendly application.

1.1 Detailed Assignment

Create a control system for operating a CNC laser cutting machine (with the intentions to operate also a high pressure water jet machine in the future). Use the TG Motion Framework (developed by TG Drives s.r.o.) which handles low level control of servomotors and other IO devices (described in detail in Chapter 2.4). The control system will be divided into PLC and a user interface. All of the components will run on the same physical PC. PLC is a specialized application running in real-time which is plugged into the TG Motion. It uses the framework to control the movements and other IO operations. The user interface is a standard application which provides interface for operators of the machine. Focus on the ease of use of the interface. The system will be created in cooperation with a CNC machine manufacturer *PTV, spol. s r.o. company*, which will use it for controlling their machines.

Features

This is a list of features the Control System will implement:

- Loading and executing CNC programs. Code of the CNC program will be based on the ISO standard. Only 2D programs will be supported.
- Graphical window showing machine activity (CNC program, trajectory of the cutting head, limits of the machine, etc.). The window will be adjustable by the user (zooming, moving, selection of display planes).
- Input through qwerty keyboard, mouse, or touchscreen (and possibly virtual on-screen keyboard).
- Manual control of cutting-head movement.
- Support for multiple cutting heads.
- Diagnostic information (speed of the cutting head, cutting time, etc.).
- Ability to transform the coordinate system of the CNC program (rotation, scale, translation).
- Resistance of the system against accidental damage by the machine operator ("foolproof").
- Customizable settings for the CNC programs.
- Authorization system with multiple levels of authorization.
- Event and error logs with the option to send them via internet.
- Ability to load a shape from a predefined shape library.
- Multi-language support.
- Application extensibility.

Chapter 2

Analysis

All types of CNC cutting machines (water, plasma, laser) share the same concept (as opposing to e.g. a milling machine, whose job is not to separate the material, but mill it) and therefore can be driven by a single control system with only minor adjustments. Figure 2.1 shows an image of a high pressure water-jet CNC machine. The machine consists of a heavy movable frame which spans the whole machine and is attached on both sides (exceptionally just to one side on smaller machines). On the frame, there is a carriage (or multiple) which can move across the frame. On the carriage, there is a specific cutting tool which can move up and down. This setup creates a three dimensional coordinate system usually using the right hand orientation. Sometimes, the cutting tool has the ability to rotate in multiple directions (which is called a 5-axis). The cutting tools are also similar since they are all some sort of circular streams. Inside the machine, there is a catcher. That is the part, where the material to be cut is put on. For water-jets, it is filled with water to stop the beam after it cuts through the material. The usual dimensions of the machine range from 0.5m x 0.5m to 5m x 10m. Maximum thickness of a material is different for individual tools – 1.5 cm for laser, 15 cm for plasma and 40 cm for water (these are just estimates, the real value depends on type of the material and machine performance).

Thanks to the universal nature of CNC cutting machines, the usage is really diverse. They are used by big companies for non-stop cutting of parts for their internal production. Such a machine can be constantly cutting for several days with a three phase shift of operators. Other use is by self-employed companies who do a contract cutting for other customers. Also in a slightly modified form, they can be used as a part of an assembly line. Many times, the CNC cutter does a rough cut of the product which is then refined by other tools. The cutting stream is always linear. This makes it ideal for cutting straight shapes.

2.1 High Pressure Abrasive Water-Jet

Water-jet cutting uses a high pressure water (ranging 500 to 6000 bar) and usually also mixes some kind of abrasive material to the water stream [1],[2]. Soft materials like rubber are cut by pure water, harder materials like metals

Figure 2.1: Water Jet CNC Machine

use the abrasive. To be really precise, the water-jet does not really cut the material, but it grinds a narrow line into it (for simplification, I will still call it cutting over the course of this work). Big advantage of the water jet is that it can cut heat sensitive materials because it does not melt the edges like for example laser does.

Behavior of the high pressure abrasive water-jet is not as straightforward as it would have seemed. To be able to cut straight edges and corners, it is necessary to use a lot of “tricks”. I will not go into physical details (because that is not the focus of this work) but in the following paragraphs I will describe all observed phenomena based on various research and testing done by the PTV company.

The beam loses its power with distance it travels. The reason for this is its conical shape which is less concentrated with increasing distance. Also when cutting the material, portion of the power was already used for cutting the upper part. This effect starts to be significant at around 1 cm of material thickness. The result of this is a V-shaped notch which results in a non perpendicular edge. If we are interested in just one side of the cut, perpendicularity can be achieved by tilting the beam into the material (see Figure 2.2).

Because of the decreasing strength of the water beam with increasing material thickness, the beam does not have enough time to cut the lower parts of the material and starts to fall behind (Figures 2.3, 2.4). This causes

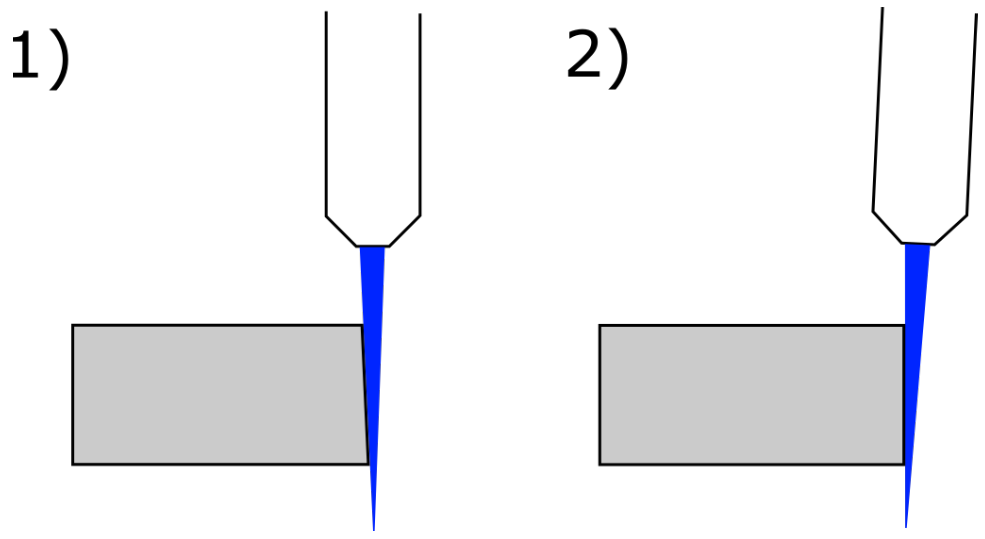


Figure 2.2: Ensuring straight edges, front view. The cutting head has to tilt into the material.

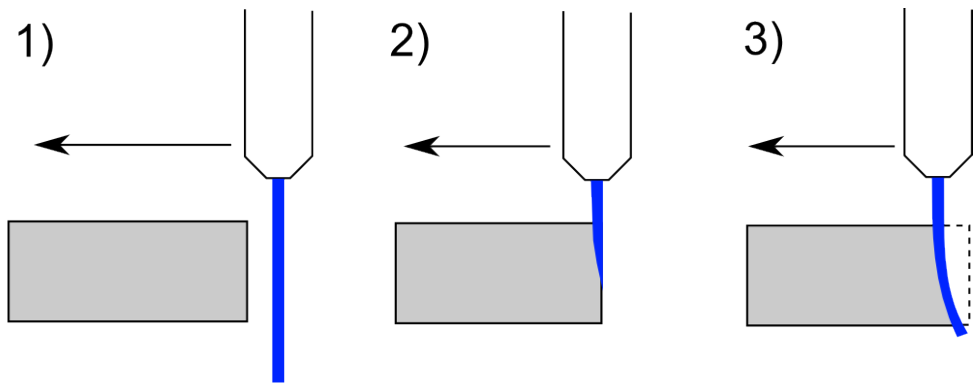


Figure 2.3: Cutting a material – side view. Notice the bent beam.

a visible coarse texture on the cut. In cases of cutting sharp edges or finishing cut of a material, it can fail to cut the material completely through. The effect can be decreased by decreasing the cutting speed.

The water beam is strongest in the middle of the beam and the power fades out to the edges. When the beam stays at one place for longer time, effective radius of the cut gets bigger. The ranging strength across the beam can cause non-straight edges or grooves in corners if the speed shall change during the whole process. If the speed slows down, it allows weaker peripheries of the beam to cut a wider hole. This can be prevented by virtually increasing the beam radius (which will cause the beam to move away from the material) when decreasing the cutting speed (Figure 2.5). This effect can be observed mostly in the corners, because the machine has to slow down to cut precisely the corner (Figure 2.6).

When cutting a part, the jet has to start the cutting somewhere. It should not start directly at the part rim as this would create artifacts on its edge. If

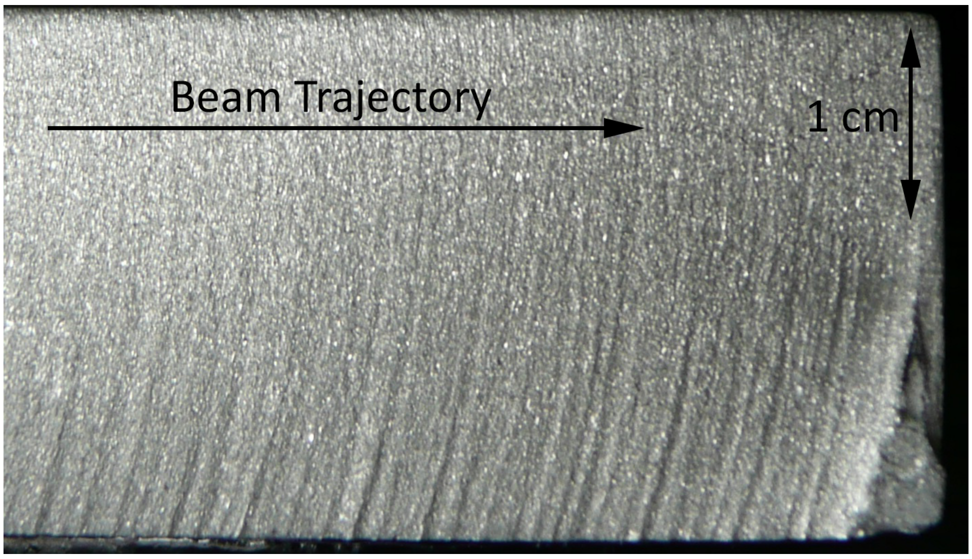


Figure 2.4: Image of cut material. Coarse texture was left by the water beam. Notice the uncut end of the material. Side view.

possible, it can start outside of the material sheet and then make its way to the part. Other possibility is to start in the material sheet but outside of the part, make a hole and then continue to the part. The process of making a hole inside the material is called piercing and it needs to be given special attention. The most straightforward way of doing the hole is by just waiting on one spot till the material is cut completely trough. The disadvantage of this approach is that the water bounces off the material back to the beam lowering its efficiency. The preferred way is to do circular movements with small radius which divert the direction of the bounce. Some materials can be difficult to pierce with water jet because of the huge initial impact force. Fragile materials like glass often crack around the pierced position. Probability of cracking can be lowered by decreasing the pressure of water during the piercing.

2.2 Control Systems

Every CNC machine manufacturer usually develops their own control system. The systems are part of company know-how and play their role in the company war. They are usually build specifically for the machines manufactured by the specific company. In this section, I will analyze the systems AremPRO Cnc886 and TG Motion Water and will discuss their strengths and weaknesses.

2.2.1 AremPRO Cnc886

AremPRO is a Czech company developing AremPRO Cnc886 control system [3]. The control system is focused on CNC cutting machines using a high pressure water jet and is currently used by the PTV company. The system can be used for abrasive and nonabrasive cutting. The software is being used

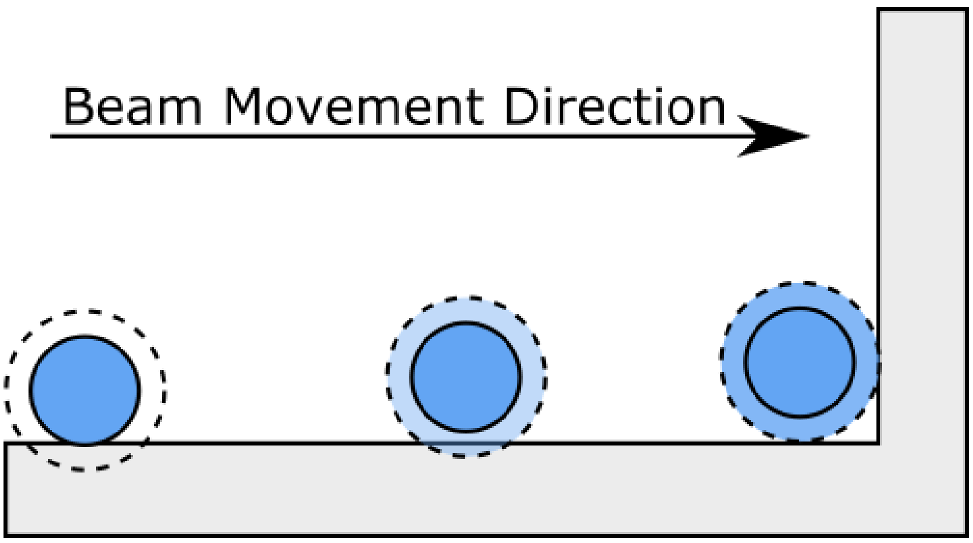


Figure 2.5: Dynamic radius control. As the beam approaches the corner, the cutting speed slows down. This causes the effective radius to increase. To prevent unwanted artifacts in the material, the beam radius needs to be adjusted. Top view.

since 2006 and after the years of active usage and testing it gained a lot of features aimed to fight physical properties of the high pressure water jet. As a result, the system is very precise. It is designed for Windows XP Embedded environment with real-time extension. It can control up to 2 cutting heads with tilt up to 60 degrees. From a user perspective, the application is logically divided into four main parts - library, manual, maintenance, and on-screen keyboard with status information. Formerly, the application used a specialized keyboard and mouse as input but during the years it evolved mostly to touch screen.

GUI of the application has a machine-like design and many controls resemble their real world counterparts. As seen in Figure 2.7, the interface consists of on-screen keyboard, visual window and six tabs. The problem of this GUI is that it displays all information and controls at the same time even if they have no function (for example axis 2 buttons when there is no second axis). Some buttons also change their meaning based on the context (for example *Start+* in the *Library* starts CNC program but in *Manual* it moves the machine along selected axis) which can be very confusing for the user. A lot of information/tabs are not relevant for a common user and could be shown only during a specialized service mode.

On-screen Keyboard with Status Info

Alongside the base application, there are also permanently visible panels with machine controls and basic information of machine's current state (like coordinates, speed, etc.). It is practical, as it allows the operator to respond quickly in every situation. The design resembles real world keyboard layout

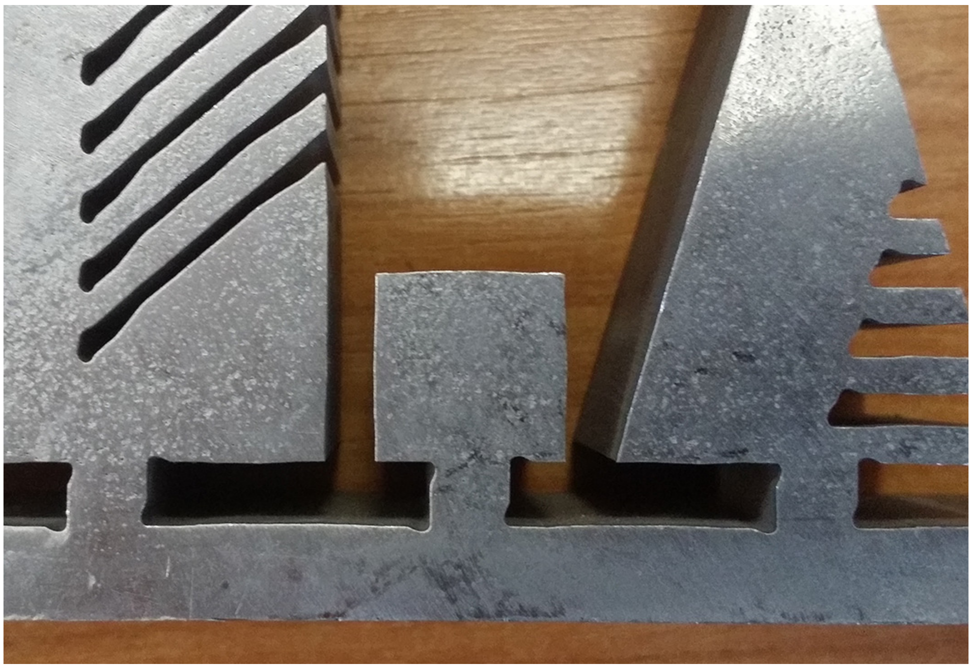


Figure 2.6: Bottom side of cut material without DRC.

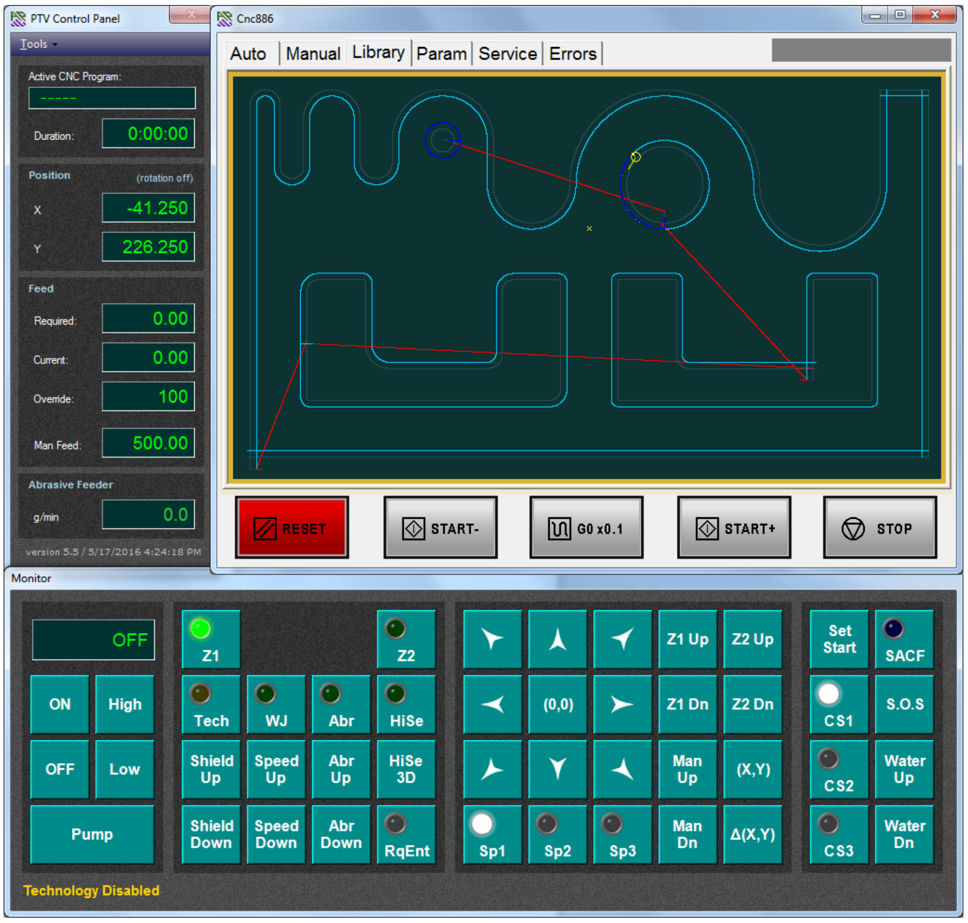
of the machine. However, some of the controls are not necessary during every situation (like when loading a CNC program) and only occupy the screen space.

Library

This is the part, where machine operator will spend most of his work time. The library offers functions for handling and executing CNC programs and contains a built-in file manager. It also has a graphical window displaying important actions on the machine, like shape of the CNC program or current position of the cutting head. It is possible to move the cropping or zoom in/out, but the controls are outdated. It is not possible to pan the view, one has to pres specific buttons for moving or zooming. After that, it takes noticeable time to re-render depending on the complexity of current CNC program. When a CNC program is loaded, a modal window with configuration of the program is shown. Big advantage of this software is its ability to calculate cutting speed (for selected cut quality) based on the properties such as material thickness or machinability.

Manual

This tab offers functions for manually moving the machine, referencing, and other functions for custom machine accessory. It is useful if the operator needs to quickly separate a material without having a CNC program. The catch is, that this functionality is already provided by the on-screen keyboard


PTV Control Panel

Tools

Active CNC Program: -----

Duration: 0:00:00

Position (rotation off)

X -41.250

Y 226.250

Feed

Required: 0.00

Current: 0.00

Override: 100

Man Feed: 500.00

Abrasive Feeder

g/min 0.0

version 5.5 / 5/17/2016 4:24:18 PM

Monitor

OFF

Technology Disabled

Figure 2.7: AremPRO control system – Library tab with a magnified window showing cutting path and the on-screen keyboard

which is always available. This makes big portions of the tab in the current state completely redundant.

Maintenance and Errors

This section consists of the tabs *Service* and *Errors* which are both focused at maintenance. The *Service* tab contains information about various variables of axes, servos, and other IO modules. It is used during installation and calibration of the machine. Therefore the whole tab has only significance for the service technicians, not for a common operator. The other tab, *Errors*, contains log of previous actions and errors. If an error occurs, most of the controls stop working and the user has to go to this tab and confirm the error.

Conclusion

Cnc886 is a really good software for water jet cutting with many useful features. From a functional perspective, it provides all tools for operating a water jet machine. On the other hand, the support for maximum of 2 cutting heads is limiting and its crappy GUI makes it hard to use.

2.2.2 TG Motion Water

TG Motion Water is a control system made by TG Drives company for high pressure water jet cutting. The concept is similar to Cnc886, but the cutting features are far more limited. It can control only a single cutting head without tilting. On the other hand, it offers a simple authorization system containing 5 different levels. This provides means to distinguish machine operator from a trained mechanic who has access to more advanced settings/functions. It is designed for Windows XP with real-time extension and can be controlled by keyboard and mouse and/or touchscreen.

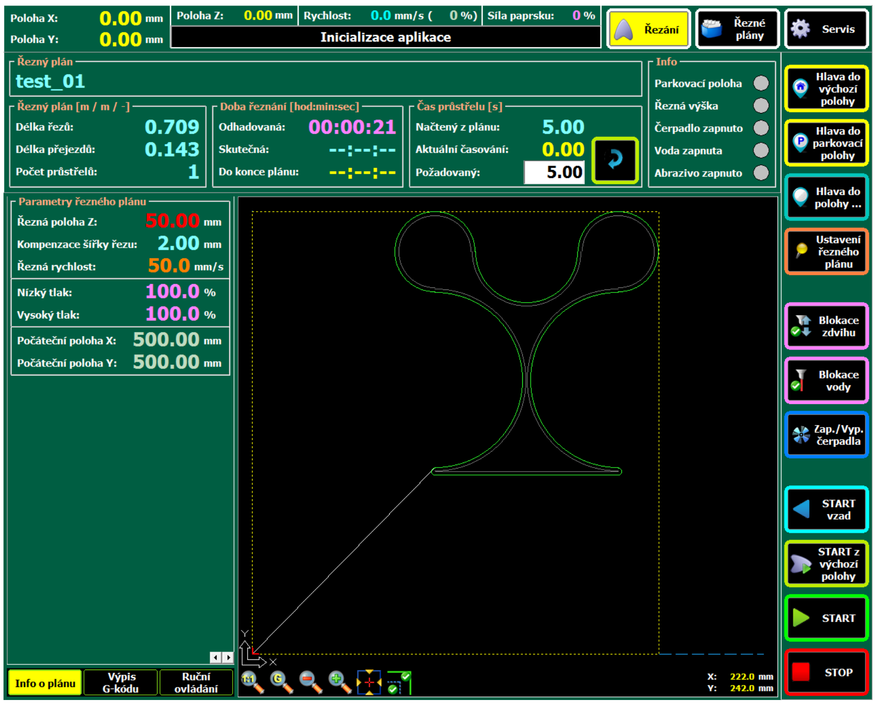
From a user perspective the program is nicely structured. When the program starts, it opens on the most important page where the operator will spend most of his time. Here we can find all the required controls and information for cutting. Graphical window is similar to the one from Cnc886, but can be freely moved with a mouse or finger. The maintenance part is nicely separated from the cutting features. The application also contains a small library of a few basic shapes.

What I found most annoying are missing default values for the CNC program configuration. This makes selection of a CNC program a painful process, because the operator has to enter every value each time. The system also cannot automatically compute the cutting speed, which you have to know either from experience or acquire it by trial/error. It is also not possible to specify the exact amount of abrasive for cutting. This is a big disadvantage, because using the right amount of abrasive for a given material can save a lot of money.

Detailed GUI Analysis

TG Motion Water GUI was designed with the touch screen controls in mind as opposed to Cnc886, whose controls mimic the real world button layouts. In the top right corner, there are three buttons which switch between three main screens – Cutting, Cutting plans (CNC programs) and Maintenance. Cutting contains all controls and information relevant to the cutting. In the cutting plans, one can load and configure CNC programs or choose a CNC program from a collection of predefined shapes. Maintenance contains configuration tools, user management system and an event log. There is also a status bar on the top which contains basic information like current state of the program or location of the cutting head. The status bar and a *stop* button are always visible independently from the current screen.

In this paragraph, I will analyze the *Cutting* screen as it is the most important one. On the top, there is information about the current CNC program. Information like currently selected program or remaining cutting time are important enough to be always visible, but information like length of the cut or number of piercings could be shown just upon user request. Next



Poloha X: 0.00 mm
Poloha Y: 0.00 mm
Poloha Z: 0.00 mm | Rychlost: 0.0 mm/s (0 %) | Síla paprsku: 0 %
Inicializace aplikace

Řezný plán
test_01

Řezný plán [m / m / -]
Délka řezů: 0.709
Délka přejezdů: 0.143
Počet průstřelů: 1

Doba řeznání [hod:min:sec]
Odhadovaná: 00:00:21
Skutečná: --:--:--
Do konce plánu: --:--:--

Čas průstřelu [s]
Načtený z plánu: 5.00
Aktuální časování: 0.00
Požadovaný: 5.00

Info
Parkovací poloha ●
Řezná výška ●
Čerpadlo zapnuto ●
Voda zapnuta ●
Abrazivo zapnuto ●

Parametry řezného plánu
Řezná poloha Z: 50.00 mm
Kompenzace šířky řezu: 2.00 mm
Řezná rychlost: 50.0 mm/s
Nízký tlak: 100.0 %
Vysoký tlak: 100.0 %
Počáteční poloha X: 500.00 mm
Počáteční poloha Y: 500.00 mm

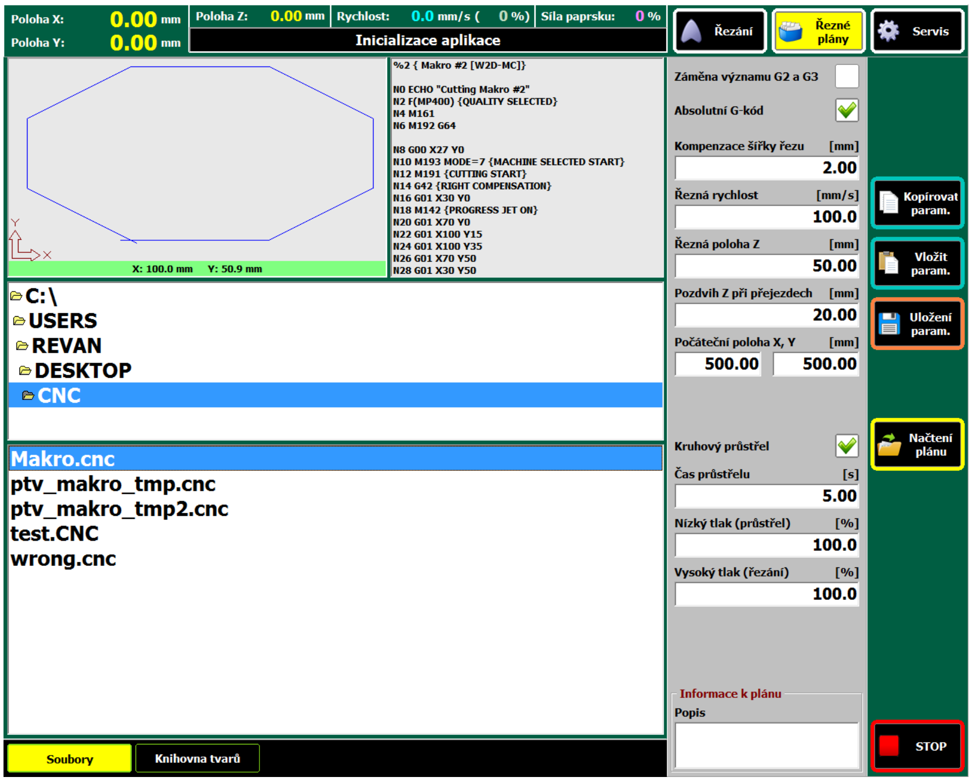
X: 222.0 mm
Y: 242.0 mm

Figure 2.8: TG Motion Water – Main screen with a CNC program

to it, there is an *Info* panel which shows on/off states of various devices. On the right, there is a menu with buttons for controlling the machine. Buttons for more complex actions like *Move to location* open a sub-menu where the user specifies additional required inputs. In the middle, there is the graphical window with additional icons on its bottom. Left from the window, there is a panel where it is possible to switch between three possible displays – additional CNC program info, CNC program G-code, and manual control. Additional CNC Program info is not really required to see all the time, it could be shown only during loading of the program. G-code is also not necessary to see when cutting. Manual control offers a way how to manually move the machine which is mandatory to have. Sadly, there is not an option to hide the menu completely to enlarge the graphical window.

Conclusion

To sum it up, compared to Cnc886, TG Motion Water offers a more user-friendly, polished interface with all parts being in their right place. It can be clearly seen, that the GUI was carefully designed. The big downside of the software is lack of some critical cutting options and a limited support for just one 2D head making it suitable just for companies with basic needs.



Poloha X: 0.00 mm
Poloha Y: 0.00 mm
Poloha Z: 0.00 mm
Rychlost: 0.0 mm/s (0 %)
Síla paprsku: 0 %

Inicializace aplikace

X: 100.0 mm Y: 50.9 mm

```
%2 { Makro #2 [W2D-MC]}

N0 ECHO "Cutting Makro #2"
N2 F(MP400) {QUALITY SELECTED}
N4 M161
N6 M192 G64

N8 G00 X27 Y0
N10 M193 MODE=7 {MACHINE SELECTED START}
N12 M191 {CUTTING START}
N14 G42 {RIGHT COMPENSATION}
N16 G01 X30 Y0
N18 M142 {PROGRESS JET ON}
N20 G01 X70 Y0
N22 G01 X100 Y15
N24 G01 X100 Y35
N26 G01 X70 Y50
N28 G01 X30 Y50
```

C:\
USERS
REVAN
DESKTOP
CNC

Makro.cnc
ptv_makro_tmp.cnc
ptv_makro_tmp2.cnc
test.CNC
wrong.cnc

Záměna významu G2 a G3		☐
Absolutní G-kód		✓
Kompenzace šířky řezu	[mm]	2.00
Řezná rychlost	[mm/s]	100.0
Řezná poloha Z	[mm]	50.00
Pozdvih Z při přejezdech	[mm]	20.00
Počáteční poloha X, Y	[mm]	500.00 500.00
Kruhový průstřel		✓
Čas průstřelu	[s]	5.00
Nízký tlak (průstřel)	[%]	100.0
Vysoký tlak (řezání)	[%]	100.0

Informace k plánu
Popis

Figure 2.9: TG Motion Water – CNC program selection

2.2.3 Other Control Systems

There are also many control systems which are not bound to a specific machine manufacturer. They are used by smaller manufacturers who cannot afford to develop their own software or by hobbyists, who build their own machines. To satisfy the broadest audience, the systems are usually multi purpose (e.g. the same system can operate a laser and a milling machine). Some of them are even open source (like LinuxCNC). In this section I will quickly examine GUIs of a few such systems.

As seen on the Figures 2.10 to 2.13, all the depicted systems have very chaotic user interfaces. The user is overwhelmed by buttons and numbers. The interfaces are nowhere near to being intuitive. As seen on the Mach 3 (Figure 2.11), the numbers are cropped, the button styles are nonuniform, and some of the buttons are even extremely scaled to the condition of being unreadable. The interfaces also contain options irrelevant to the user, or even worse, have no functionality. This can be clearly seen on the WinCNC on Figure 2.10. It contains multiple buttons with the label “Custom Button” which does not perform any action. There is no reason to bloat the interface with them. The UCCNC (Figure 2.12) shows current position in the G-Code. That is completely unimportant for the majority of users. If at all, it should be a switchable option. All these remarks will be taken into account when developing GUI for this control system.

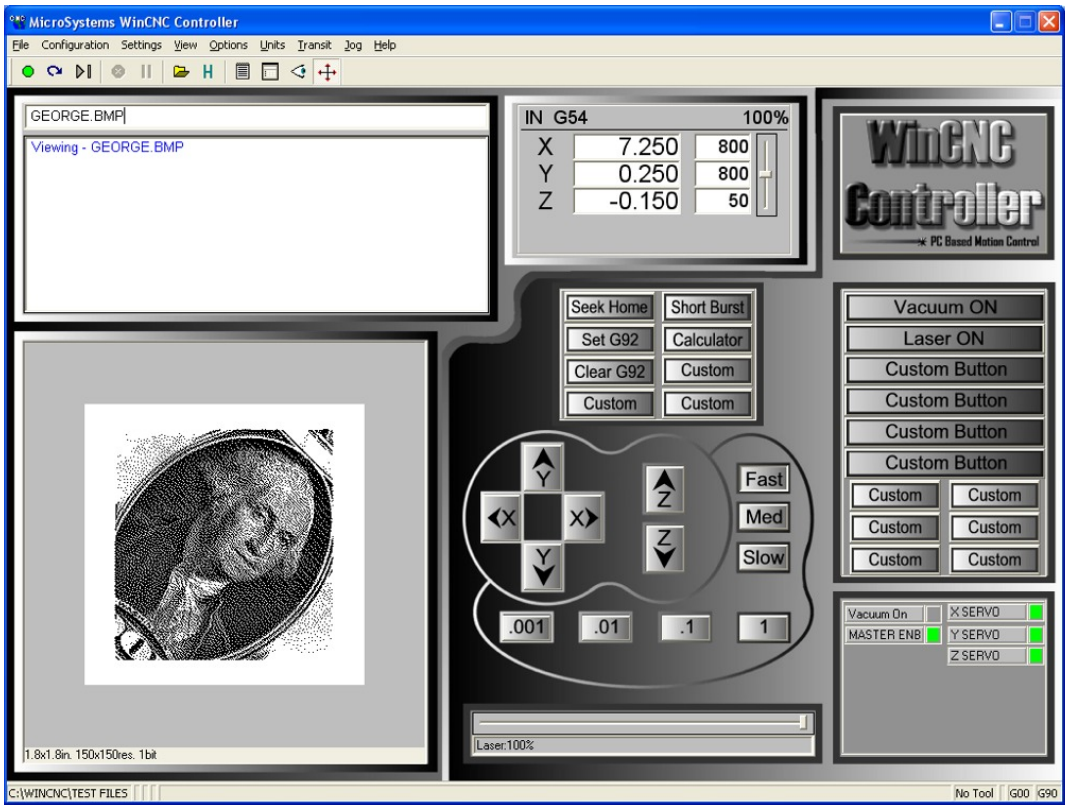


Figure 2.10: WinCNC – Customizable control system mainly for Plasma/Laser/Router machines. Runs on Windows XP and higher. It is sold together with a PC.

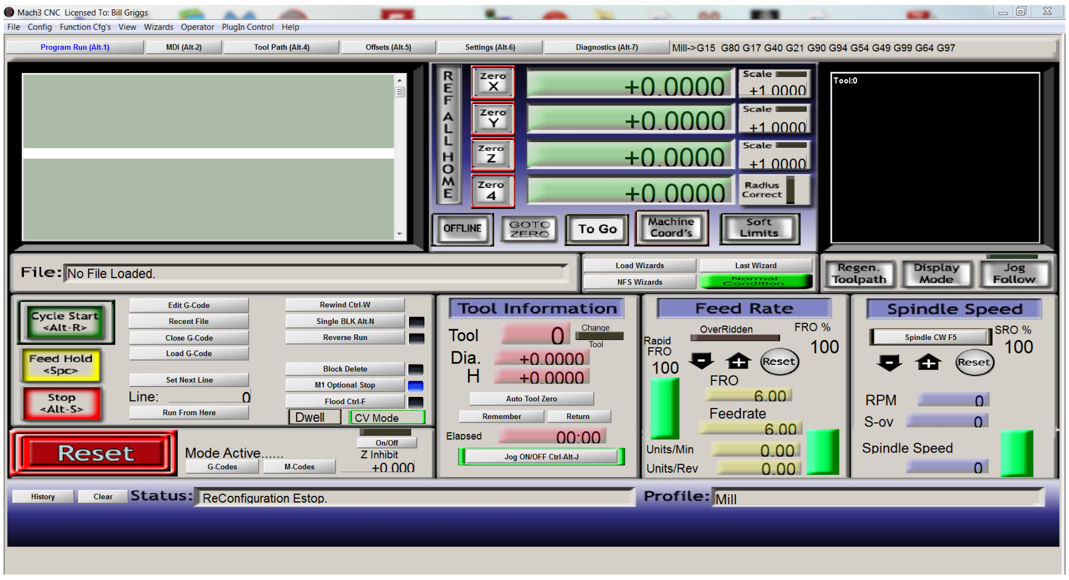
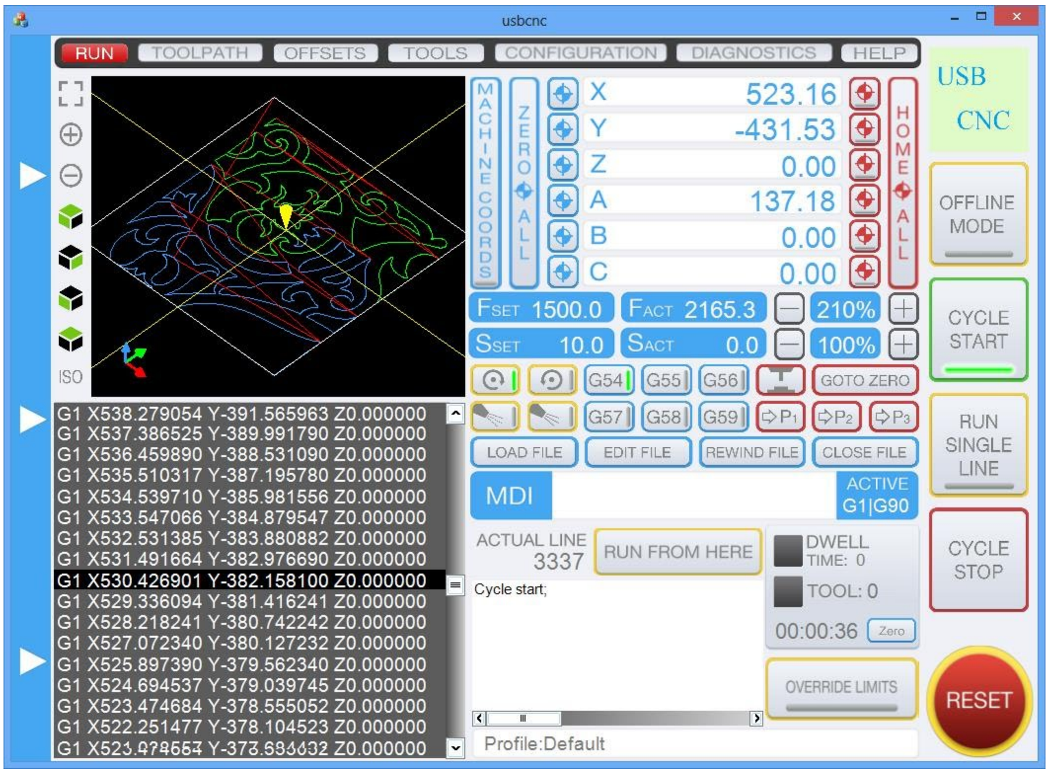


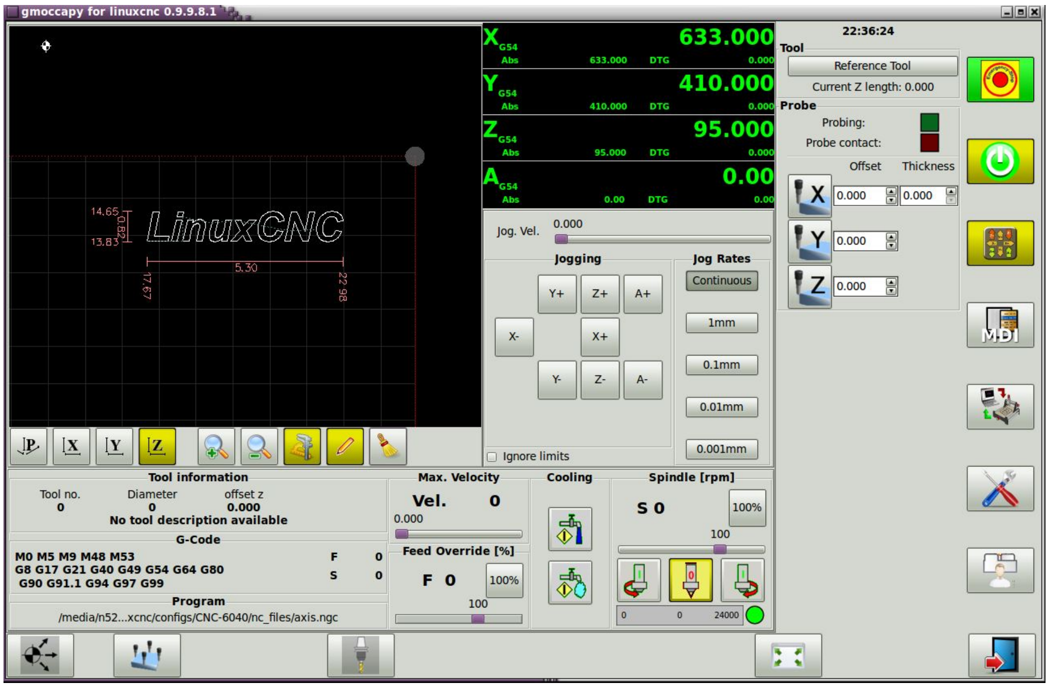
Figure 2.11: Mach 3 – Multipurpose control system. Labels itself as the “most intuitive CNC control software available”. Runs on 32-bit versions of Windows 2000 and higher.



```
G1 X538.279054 Y-391.565963 Z0.000000
G1 X537.386525 Y-389.991790 Z0.000000
G1 X536.459890 Y-388.531090 Z0.000000
G1 X535.510317 Y-387.195780 Z0.000000
G1 X534.539710 Y-385.981556 Z0.000000
G1 X533.547066 Y-384.879547 Z0.000000
G1 X532.531385 Y-383.880882 Z0.000000
G1 X531.491664 Y-382.976690 Z0.000000
G1 X530.426901 Y-382.158100 Z0.000000
G1 X529.336094 Y-381.416241 Z0.000000
G1 X528.218241 Y-380.742242 Z0.000000
G1 X527.072340 Y-380.127232 Z0.000000
G1 X525.897390 Y-379.562340 Z0.000000
G1 X524.694537 Y-379.039745 Z0.000000
G1 X523.474684 Y-378.555052 Z0.000000
G1 X522.251477 Y-378.104523 Z0.000000
G1 X521.028557 Y-377.683632 Z0.000000
```

Cycle start;

Figure 2.12: UCCNC – Multipurpose control system. Runs on Windows XP and higher.



Tool information

Tool no.	Diameter	offset z
0	0	0.000

No tool description available

G-Code

```
M0 M5 M9 M48 M53                    F  0
G8 G17 G21 G40 G49 G54 G64 G80      S  0
G90 G91.1 G94 G97 G99
```

Program

/media/n52...xcnc/configs/CNC-6040/nc_files/axis.ngc

Figure 2.13: LinuxCNC – Open source (GNU GPLv2 license) multipurpose control system. Runs on major distributions of Linux.

2.3 G-code

G-code is a programming language for CNC machines consisting of instructions describing movements or actions. It is used by variety of CNC machines, it can be also used by non-cutting machines like printers or plotters. It is standardized by the ISO 6983 [4], but some countries use different standards. The ISO 6983 standard defines the structure of the language and standard commands with their meaning. Each command consists of a letter, number and optionally parameters. The language got its name from the most common commands starting with the letter G. Commands starting with the letter G either describe the shape or the motion. Commands starting with the letter M represent miscellaneous actions which are often specific to type of the CNC machine. For example, the command G01 moves the tool in a straight line to a target position. It takes the target coordinates as arguments, therefore the full command could look like: G01 X15 Y20.

However in reality, manufacturers use their own variation of G-code, which is usually based on the standard but altered in a way. The requirements of manufacturers are too different to be covered by the standard. And the standard is too loose when it comes to certain areas, for example it does not even define how to write comments. Usually the G commands are used from the standard, but the more specific M commands differ. This makes individual programs incompatible between systems of different manufacturers. Because the G-code comes usually as export from a CAD program, the manufacturers provide so called post-processors. They give the CAD programs the ability to export their dialect of the G-code.

I will demonstrate the G-code on a simple example as seen in Figure 2.14. It is a G-code consumed by AremPRO control system for water-jet and cuts a simple rectangle. Strings between the curly braces are comments. The line N1 selects the level of quality of cut which was previously defined in the control system by operator. It defines how fast the cutting head will move. The line four moves the head to the initial position. Lines five and six start the cutting process. Line seven selects the compensation[1] for the cutter. Lines 8,10-14 describe the shape of the rectangle. Lines 15,16 and 18 stop the cutting process and end the program. Lines 2,3,9 and 17 are manufacturer specific commands.

2.4 TG Motion Framework

TG Motion is a framework developed by **TG Drives, s.r.o.** that allows to control movement of a single or multi-axis machines [5]. It uses EtherCat bus for communication with servos and digital I/O units (DIOs). Advantage of using TG Motion is that it already implements a lot of low-level functionality for operating a machine. Without the framework, I would have to do all the work myself which would drastically increase the difficulty of implementing

[1]http://emc.sourceforge.net/Handbook/node82.html

```
%Makro #1 {[W2D-MC]}
N0 ECHO "Cutting Program Makro #1"
N1 F(MP400) {QUALITY SELECTED}
N2 M161
N3 M192 G64

N4 G00 X-3 Y0
N5 M193 MODE=7 {MACHINE SELECTED START}
N6 M191 {CUTTING START}
N7 G42 {RIGHT COMPENSATION}
N8 G01 X0 Y0
N9 M142 {PROGRESS JET ON}
N10 G01 X150 Y0
N11 G01 X150 Y100
N12 G01 X0 Y100
N13 G01 X0 Y0
N14 G01 X0 Y-2
N15 M192 {CUTTING END}
N16 G40 {COMPENSATION OFF}
N17 M140 {PROGRESSJET OFF}

N18 M30 {PROGRAM END}
```

Figure 2.14: G-code consumed by AremPRO control system. It cuts a rectangle.

the control system. To guarantee a constant delay required by the HW, TG Motion runs in real-time. It uses real-time OS IntervalZero[2] for that purpose which is installed alongside Microsoft Windows.

2.4.1 Programmable Logic Controller

Programmable logic controller (PLC) is a pluggable library which can enhance the functionality of TG Motion. It can control servo drives, read and set values of digital I/O units and communicate with other peripherals through TG Motion. It is required to be written in a native code and has additional restrictions imposed by the real-time platform. It cannot have any external dependencies (except for standard library functions), cannot call any OS functions and cannot use dynamic memory management. Dynamic memory is not supported to guarantee consistent deterministic timing. TG Motion itself runs in cycles with predefined period time. Communication with outside peripherals is realized during every cycle. This imposes constraints on the PLC, as it needs to be fast enough to finish all calculations before the next cycle. Shared memory is used as a mean of communication between TG Motion and PLC. The PLC is explained in detail in Chapter 6.

[2]http://www.intervalzero.com/

```
Servo[00].Type=5
Servo[00].Node=1
Servo[00].Axe=1
Servo[00].Resolution=20

Servo[01].Type=5
Servo[01].Node=2
Servo[01].Axe=1
Servo[01].Resolution=20

Dio[00].Type=1
Dio[00].Node=1
```

Figure 2.15: Snippet of a TG Motion hardware configuration. The servo 00 and 01 are assigned the same axis – that means they are controlled synchronously as one.

2.4.2 Hardware Communication

Based on TG Motion manual [6], the framework can control up to 256 servo motors and DIO units. Only certain types of motors/units can be driven by the framework. They are listed in the manual and many of them are manufactured by the TG Drives company. That is not a problem, as they are currently used for manufacturing of the CNC machines by PTV but it creates a vendor lock-in to the future.

The HW peripherals and their specifics are configured in TG Motion configuration file (see Figure 2.15). TG Motion handles HW differences between individual unit types and provides unified interface for the programmer. Servo motors work on basis of increments. Increment is the smallest amount the motor can move. Motor resolution is a count of increments per a full rotation. Position of a motor is stated in increments. To actually get position of the motor inside the machine, it is necessary to have a reference point with fixed and known location. The reference point is usually a HW sensor located in the farthest position of the axis. After every start of the machine, it is required to move the motors to their reference points to be able to measure their location within the machine coordinates. This process is called referencing.

Chapter 3
User Interface Design

In the introduction I stated that the biggest problem of current controls systems are their heavyweight and outdated user interfaces. Aim of this chapter is to design an easily usable and appealing user interface. First I created a mockup of the GUI. Then I tested it with real users and applied their feedback. Finally, I proposed new features which would improve usability of the interface.

3.1 Prototype

This section contains design of the applications GUI and describes various controls. GUIs of industrial software tend to be static, heavyweight, and overwhelm the user with a lot of unnecessary informations. I will try to design this one an opposite by following modern UI principles and using modern controls. The weaknesses of other control system mentioned in the analysis will be also taken into account.

Input

As mentioned in the specifications, there will be multiple supported input methods - touch screen, mouse, and qwerty keyboard. It is important to also consider the environment when designing the UI. Workplace around CNC cutting machines are usually dirty and operators can wear slim gloves. Therefore the UI must be designed in a way that it shouldn't require too precise movements. Also during cutting, operators tend to use touch screen for controlling the machine, keyboard and mouse are used just for non-frequent actions like servicing.

User and Task Analysis

To be able to design a pleasant GUI, it is required to analyze what kind of users will use the application and address their needs. Based on experience, users can be divided into 3 groups – machine operators, service technicians, and administrators.

Machine operator – Cuts parts and does simple service tasks like exchanging nozzles. He is often an employee of the company utilizing the CNC machine. His usual workflow is to load and start CNC programs and exchange material sheets after the programs are finished.

Service technician – Calibrates and maintains the machine itself. Often employee of the company manufacturing the CNC machines. His job is to make sure the machine is ready to use and well calibrated. Needs access to a low level configuration of the machine. Cuts mainly to check correct functionality. They spend more time with the hardware than with the software.

Administrator – Maintains more advanced settings of the application (like properties of materials, cutting settings, etc.) and manages other users accesses. It is usually owner or an experienced worker. He can also work as a Machine operator at the same time.

The most frequent user is by far the machine operator, therefore the UI should be designed around their usage. They are usually people with lower qualifications and with little experience using similar applications. Therefore the simpler the UI will be, the better. To appeal to more experienced users, there could be a possibility to switch to a more complex layout with advanced features.

The workload of the machine operators can be further broken down into three categories based on the type of company they work in:

Serial production – The company does a mass production of products. The parts they cut may already be final products or used as internal parts inside the final products. The operator usually cuts the same program over and over again. The CNC program is already adjusted for the type of material sheet used. The company may also employ multiple shifts per day to keep cutting 24/7. Operators job is just to exchange the material sheets and ensure the program is running correctly.

Contracting – The company is a job shop. They cut what customers require and every order can be different. The workload needs to be more flexible than during the serial production. The operator needs to select a proper material sheet and adjust the position of the CNC program for every contract. The customer may supply their own material sheets, but usually the job shop keeps their own materials which they reuse.

Manufacturer – The company itself is a producer of the machines. The operator is a skilled person with deep knowledge. He needs access to all settings and features as he uses them for improving the cutting process. They can alter internal setting like acceleration which would be too dangerous for a normal operator and could lead to damage.

Guidelines

Here is a list of generally valid guidelines which hold true for any GUI. They will be followed when creating the GUI.

- Use icons where appropriate. They help with navigation through the interface and can be understood even by people not proficient in the given language.
- Guide the user with colors. Green color is associated with positive action, red color with negative one. It will help the user to decide more quickly and even more importantly, it will prevent choosing the unwanted option.
- Each button should have only one function. In AremPRO, Start button was recycled for multiple actions. This can be really confusing and requires the user to remember the state of the application without getting any feedback.
- It should be clearly indicated which button is pressed. Operator should be able to see it from a distance.
- Stop button should be always available and in a fixed place. If a problem happens, the user will know a quick way how to stop any action.

GUI Mockup

When designing a UI, it is good to find the most common workflows and design the UI around them. As the theory says, the user will spend 80% of his time using 20% of the application functionality [7]. The most common workflow for the machine operator is as follows: power up the machine and start the software -> check state of the machine and reference it -> place a material sheet to the machine -> select a CNC program and set a zero point -> start the program and wait for it to end -> possibly select another CNC program, replace the material sheet and set a new zero point & repeat -> switch off the machine and the software.

When starting the application, the user will first need to provide his username and password. This will be a standard login dialog and therefore not important for sketching. For companies, who do not care for authorization this feature can be turned off and the user will be automatically logged in as a super user and this dialog will be skipped.

Figure 3.1 shows the main screen of the application. It contains necessary controls for controlling the machine and cutting CNC programs. (1) contains information about the currently selected CNC program. By clicking the *Manage* button a new screen will open with the option to customize the current program or to select a different one (Figure 3.3). This button is disabled during cutting. (2) is a status bar displaying current activity. Under the bar, there is a list of cutting heads with the option to enable/disable each head separately. (3) shows basic diagnostic information like position or speed of selected head. (4) contains sliders which can percentually adjust some

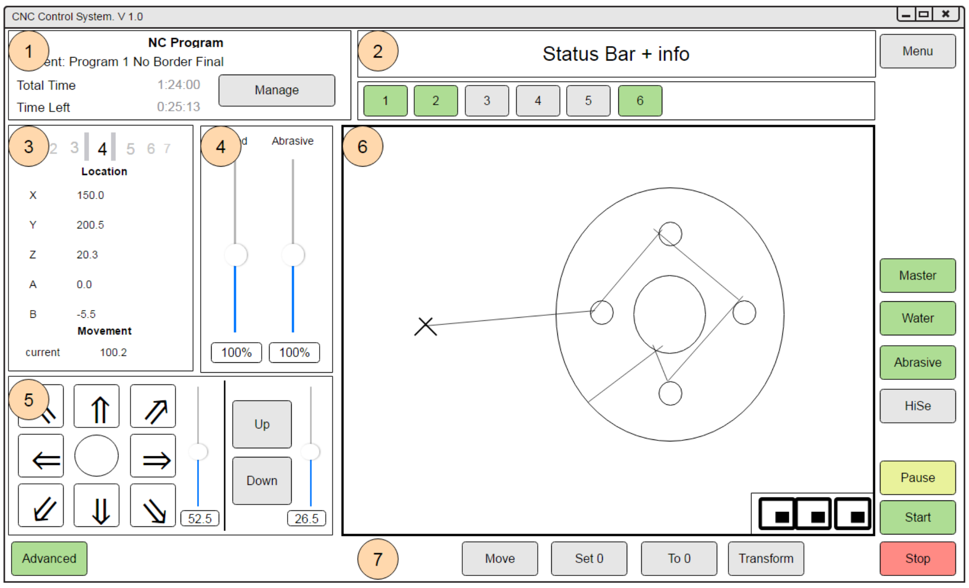


Figure 3.1: GUI Mockup – Main page

properties like cutting speed. This can help to fine tune the quality of the cut during cutting. (5) allows to manually control the selected heads. The sliders adjust the moving speed. By directly clicking on the number box the user can enter a specific value. For a more freeform movement control, the user can double-tap the middle circle to transform the control to a freeform circular control (Figure 3.2). The heads are controlled by holding the pointer inside the bigger circle, direction is specified by angle and speed by distance from the circle center. (6) is a graphical window displaying the CNC program and other information. On the bottom there are various controls for manipulating the view. Holding (or right clicking) a specific position in the window will open a context menu. It will contain an option to move the head to the currently selected position. (7) is a toolbar with the controls. The button *Menu* opens a menu where the user can access other pages like maintenance or event log. Buttons around the graphical window are either for controlling the machine or the CNC program. Button *Advanced* toggles between basic and advanced versions of the interface.

The Window show in Figure 3.3 allows to load and customize the CNC programs. By clicking the *Load* button, a drop-down will appear with possibilities to load the program from a file or from a database of pre-made shapes. In the top left corner, there is a preview of selected or currently loaded program. Beneath it there is its G-code. The right box contains information depending on the current state. Either configuration of the loaded program, a file browser when loading from a file or a shape gallery when loading a from Makro library.

Windows like maintenance (containing machine variables, etc.), log, user accounts will contain just tables of data and will not be accessed on regularly basis, therefore their design is not that critical and no sketch will be provided.

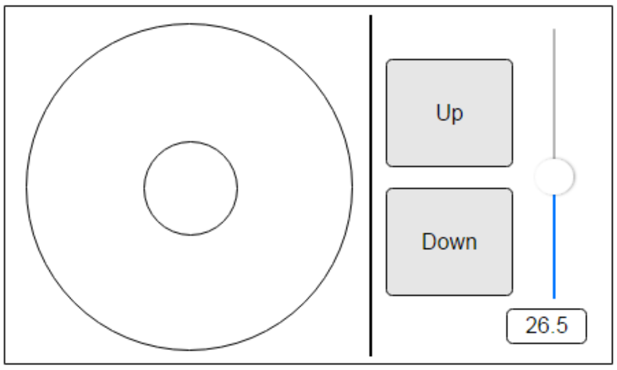


Figure 3.2: GUI Mockup – Circular control for freeform head movement

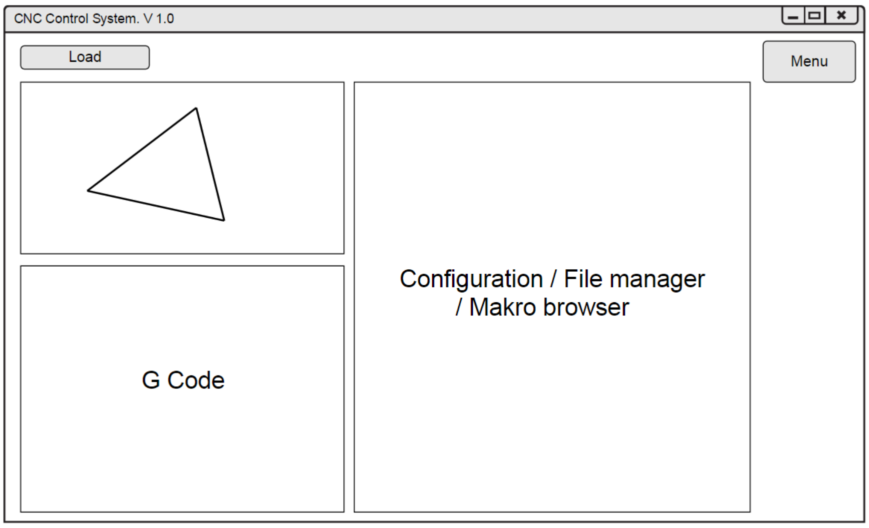

CNC Control System. V 1.0
Load
Menu
Configuration / File manager / Makro browser
G Code

Figure 3.3: GUI Mockup – CNC program selection and customization

In the next section, I will describe consultations of this GUI mockup with real users.

3.2 Usability Inspection

The proposed mockup was discussed in two sessions of usability inspection with the users with previous CNC cutting experience. We evaluated the given prototypes and current appearance of the program and tried to come with improvements. They also gave me feedback of using previous cutting software. Many of those were subtle quality-of-life improvements and are covered in the additional guidelines.

Manual and Automatic Mode

The idea is to split the cutting environment into two modes – manual (3.4) and automatic (3.5). Each of the modes would feature controls relevant only

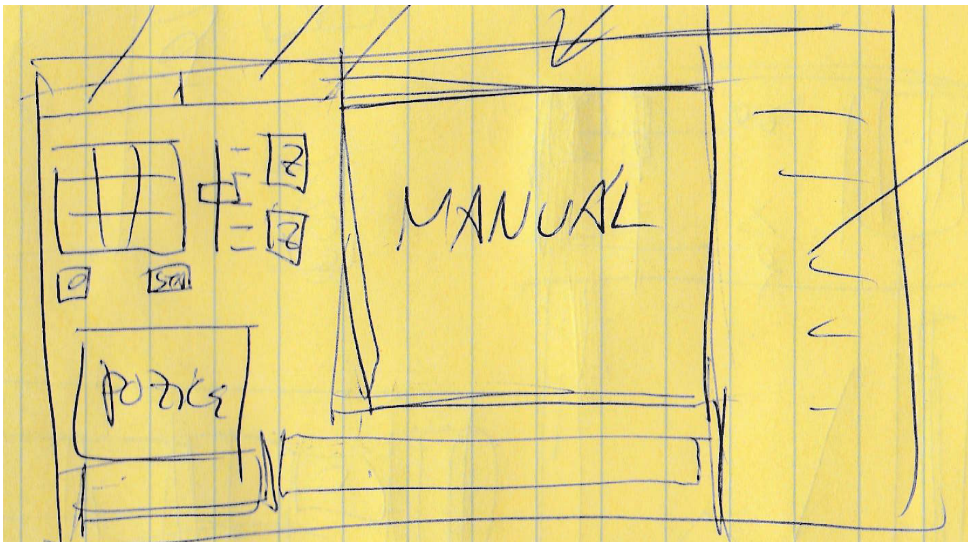


Figure 3.4: Manual mode – sketch created during usability inspection

for the given mode. During the manual mode, the user has full control of the machine motion. He is also able to manually enable/disable the cutting beam. If he selects and starts a CNC program, the environment will switch into the automatic mode. During automatic mode, the manual controls are replaced – machine is completely driven by the CNC program, so there is no need of having the movement controls on the screen. The controls in automatic mode would modify the machine behavior during the cutting. There would be sliders to modify speed of the cutting or abrasive amount (for water-jet) by a percentage. Also a progress bar indicating progress of the program would appear. When the program stops (or the user cancels it), the environment would switch back to the manual mode.

Speed Setter

When manually moving a cutting head, the user should be able to define the speed of the movement. Both showed systems solve it similarly. AremPRO uses a global variable for manual movement speed which is possible to increase or decrease by predefined increments (following a non-linear scale). When movement button is pressed, value of the variable is used as a speed for the movement. TG Motion Water's movement buttons have a unchangeable predefined speed. It is only possible to switch the movement to a jogging mode which will stepwise increase the movement speed, but again to a predefined value.

Both of these approaches are inconvenient to use as they do not provide a way how to quickly and precisely control the speed. Therefore, we designed the speed setter component (lower part of Figure 3.6). It is a complement to the movement control from the mockup with the ability of setting desired movement speed. It consists of a slider enriched with multiple reference points. Each reference point represents certain predefined (but configurable)

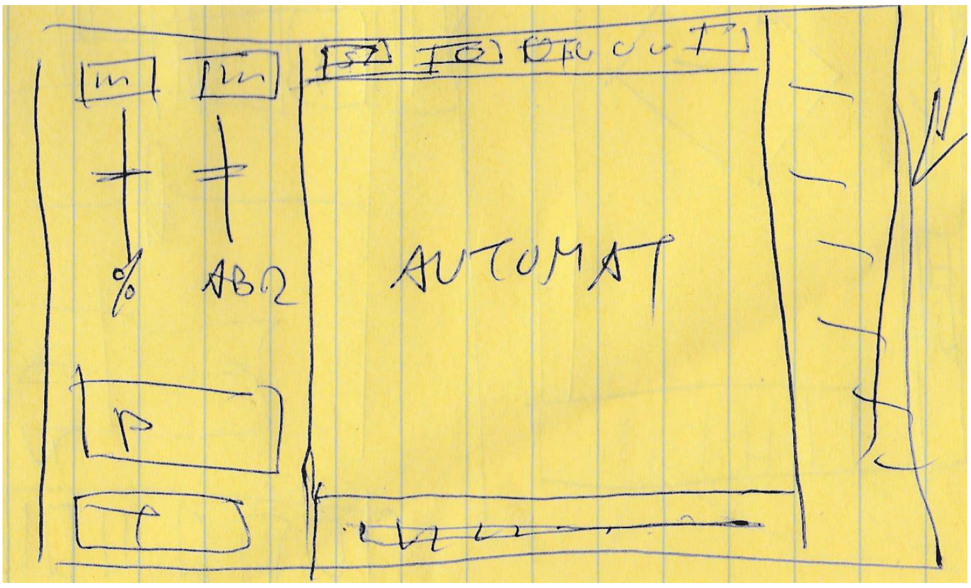


Figure 3.5: Automatic mode (active during cutting) – sketch created during usability inspection

speed and when clicked, it moves the slider to the given speed value. The user is also allowed to fine tune the speed by moving the slider or by clicking left or right side of the slider. Above the slider there is a textbox displaying currently selected speed. The control gives the ability to quickly select a rough speed but also allows to fine tune it if a higher precision is needed.

Additional Guidelines

Additional set of guidelines was acquired during the usability inspection. They are often influenced by results of bad user experience with the previous systems.

- **Avoid sound signals.** The environment and especially the water jet tend to be very loud therefore the signals would be easily missed. This would cause even more troubles if they were signalizing important events like errors.
- **Prevent accidental press of unwanted buttons.** Buttons with a big potential destructive effect (like move the head to the zero point) should be far enough from other buttons.
- **Avoid interchanging of different buttons (like start and stop).** In cause of a problem, the user might panic and hectically smash the button (e.g. with intention to stop the machine) which must not switch between the functionalities.
- **Keep buttons pressed when sliding the pointer from them.** This is even more important when the application is controlled by the touch

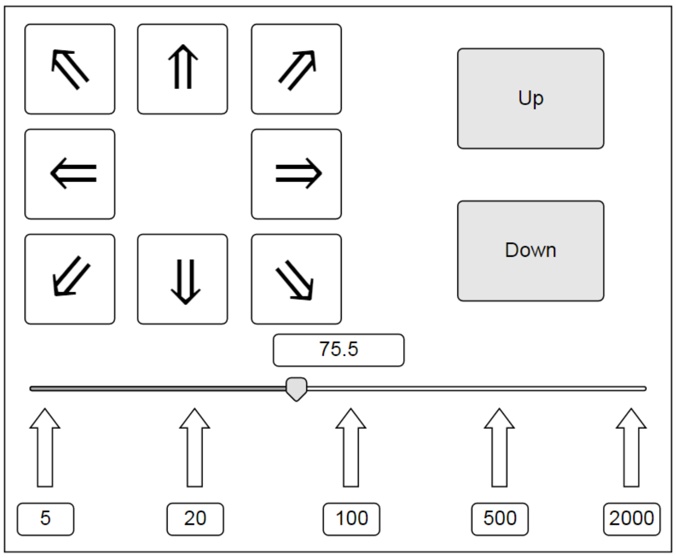


Figure 3.6: Manual movement window with the speed setter component.

input. The user might not look at the screen and accidentally slide from a button. It prevents unwanted trigger of other buttons.

Final Design

The final appearance of the interface will be a combination of the first designed interface (Figure 3.1) with all improvements from the usability inspection. The resulting interface will use two separate modes, one for CNC programs and the other for manual control. Many of the components featured in the mockup will be distributed into the two modes which would make the interface simpler. The movement control from the mockup will be replaced with the more sophisticated speed setter. All the mentioned guidelines will be followed.

3.3 Interface of the Future

In this section, I will propose how how an interface of the control system could look like in the future. It shows various ideas that would improve either its usability or automate certain tasks. These features will not be implemented in the scope of the thesis but serve as suggestions for future work.

3.3.1 Start from Arbitrary Position

The application has the option to pause cutting and resume it afterwards from the same position. But there might be situations where something goes wrong and the user needs to stop the application. Or a situation, where the abrasive gets stuck and the last few centimeters did not get cut. Now the operator is forced to start over again and waste time and resources. Therefore an option to continue cutting from an arbitrary position would be helpful.

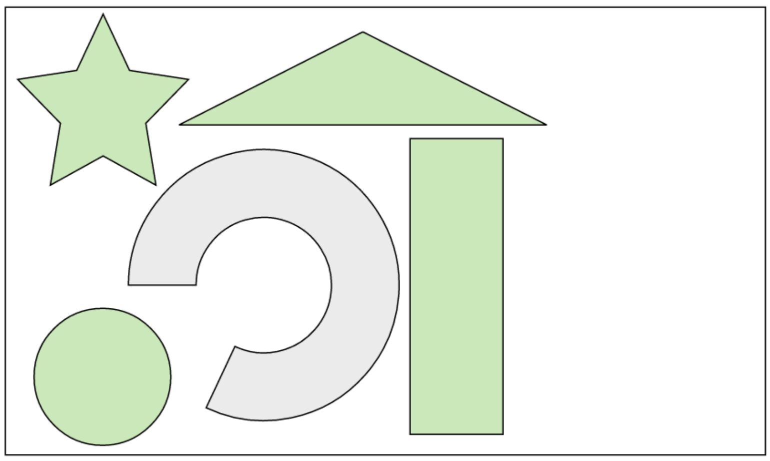


Figure 3.7: Cut history as shown in the graphical window. The big rectangle represents border of the current sheet. Green parts were already cut into it. Grey part represents the current program for which the operator searches an empty place.

The user picks a position on the graphical window which snaps to the nearest point on the path of the program. He is then able to fine tune the resulting position by moving along the contour of the shape.

3.3.2 Shape Projection

When the material sheet has holes already, it can be hard to find position for a next part to be cut to fit to the remaining material. To help the positioning, outline of the CNC program would be projected to the sheet. This would allow the operator to visually check, whether the program would fit in the sheet. It requires attaching a physical projector to the machine and calibrating the exact distance from the projector to the sheet so it does not distort the size. Also the projector would get dirty in short time, which requires either regular cleaning or a protective barrier.

3.3.3 Cut CNC Program Memory

Another solution to the problem of shape placement can be to automatically track already cut hole for each material sheet. The operator would see which parts were already cut in the graphical window (see Figure 3.7). This could help him to position the current part more accurately.

To extend this feature even further, there could be an option to save and load history for each sheet. Because position of the physical sheet has probably changed from the last cutting, there needs to be a calibration of its location, so the system can draw it accurately to the graphical window. This could be done either manually or automatically as suggested in Section 3.3.6.

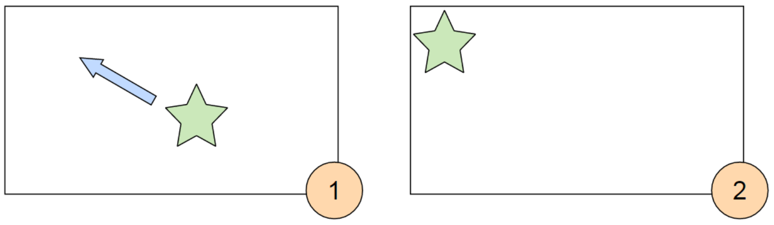


Figure 3.8: In the first screen, the user places the parts inside the sheet and selects a direction. On the second one, the part is placed automatically into a fitting spot in the corner.

3.3.4 CNC Program Queue

Some CNC programs can be pretty short which requires the operator to be at the machine all the time to place and start new programs every few minutes. This function would allow to queue up multiple CNC programs in succession and then execute them as a batch. This function would mainly help the companies, which cut different programs every time and cannot prepare them beforehand.

3.3.5 Automatic Head Docking

After finishing the cutting process, the operator usually needs to either take the cut part out or even replace the whole material sheet. This task is hindered by the cutting head which usually ends above the cut part. Therefore the operator must first move the head to a remote place to perform the task. The solution would be to define a docking position where the head would go after finishing a CNC program. There can be multiple docking position based on the location of the head like a nearest corner.

3.3.6 Material Sheet Position Recognition

Finding the correct location for a CNC program is one of the most common and annoying tasks of a operator. To make it easier, the machine would have the ability to recognize position of a material sheet on the cutting table. Coupled with the 3.3.3, it would be much easier to find a correct spot for the program on the display. To find the position of the sheet, it would use a camera and image recognition. To make the process easier, the sheets could contain some sort of identification, like a QR code in their corners.

3.3.7 Nesting/Position Selection of a CNC Program

Coupled with features from Sections 3.3.3 or 3.3.6, the system would be able to find a fitting position for a program. It could either work automatically or semi-automatically like in CAD programs, where the user positions the piece and then chooses a direction and the program finds the best fitting spot in that direction (Figure 3.8).

Enhanced version of this functionality could do this automatically and for multiple parts. Companies often invest in nesting[1] modules for their CAD programs. The CNC programs are often done and cut by different people. Therefore if the material sheet has different dimensions than anticipated, the operator has to wait for the program to be rebuilt.

3.3.8 Automatic Sheet Selection

The job shop companies can put a lot of money and effort into selecting the best fitting material sheet for their CNC programs. They usually pile dozens of used sheets and selecting the right one could take couple of minutes. And the exchange of the sheet can be also a physically demanding task. With the prerequisite of cut memory (Section 3.3.3), the program could automatically select the ideal sheet. It would be based on a similar system which is already used in automatic warehouses. Operating robots can automatically deliver and store selected items. They would automatically retrieve the selected sheet, exchange it with the current sheet and store the current sheet for future use. This feature would bring most benefits to bigger companies with multiple machines cutting diverse programs.

[1]"Nesting refers to the process of laying out cutting patterns to minimize the raw material waste."[8]

Chapter 4

Architecture

Software architecture is a basic foundation describing organization and behavior of components of a system [9]. When designing the software architecture, one has to consider functionality, environment, and future sustainability. Failing to consider a important architecture aspect could put the application at risk and changing a key architecture element would usually require to completely rewrite the application.

The architecture diagram is shown in the Figure 4.1. As seen on the diagram, the hosting computer runs two separate environments simultaneously. The CNC machine will be operated by TG Motion run on the real-time OS. Real-time systems can guarantee a consistent latency for applications which is a necessity when operating such HW. Inside TG Motion, there will be a plugged-in PLC. The PLC will be a simple program resembling a finite-state machine.

The second environment is a standard OS hosting the UI. The UI is the only part the user will come into contact with. It provides interface for controlling the machine to the user. The UI won't operate the machine directly, it will send commands to the PLC through a communication link and the PLC will then perform the commands.

Even if the UI is a technical software, its architecture is inspired by structure of modern web applications. To keep proper separation of concerns, it is split into Core (backend) and GUI (frontend). The Core contains the business logic and the data access. Important parts of the logic are implemented by services whose dependencies are managed by a container. The Core is unaware of the GUI or underlaying windowing system. The GUI contains interaction and visualization logic. The separation makes exchange of used GUI technology easier. Even if not yet planned, there could be multiple separate GUI clients. With today's wide usage of smartphones, I could image a mobile client with a minimalistic functionality.

The individual blocks shown in Figure 4.1 will be described in detail in the following chapters, PLC in Chapter 6, Core in Chapter 7, and GUI on Chapter 8.

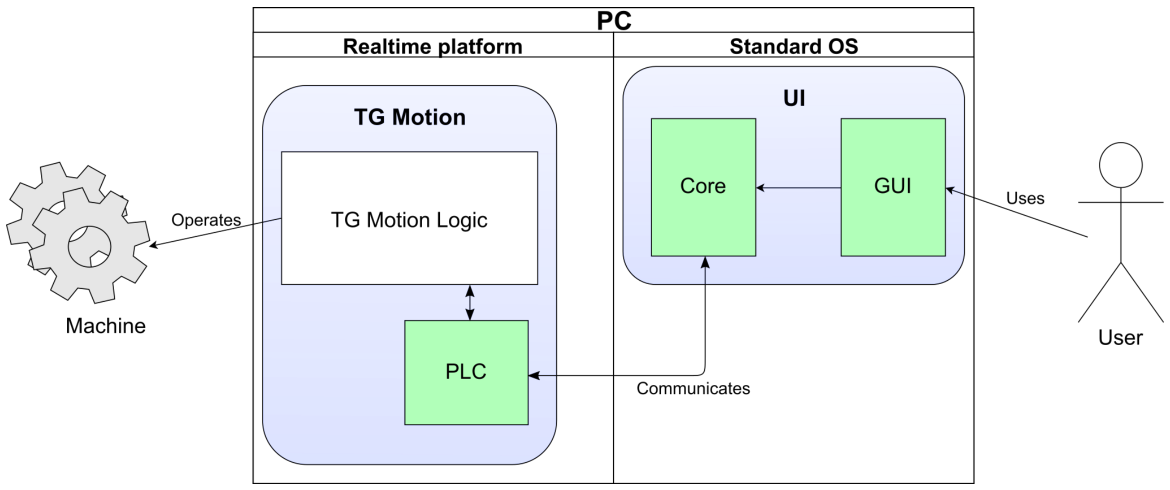


Figure 4.1: System Architecture Diagram – blue tiles represent standalone applications, rectangles represent individual components, green rectangles are developed as part of this thesis.

4.1 Architecture of a Module

Modules provide a way how to extend the behavior of the application. Architecture of a module is similar to the architecture of UI. Modules will usually contain business logic and a GUI. There needs to be a way for the application to recognize given library as a module. This could be either configured within the application core, or in a more pleasant way, the application itself could discover the modules by scanning a predefined folder. The installation of a module should be simple to be able to be performed by an inexperienced user. Something as dropping file(s) to a folder. There is no way how to extend the PLC, therefore the extensibility is limited just to the UI.

Chapter 5

Technology

This chapter describes selection of technologies (languages and frameworks) used for implementation of the control system. The reasons for picking the specific technologies are explained.

PLC

Due to the nature of PLC, we are constrained in use of third-party libraries. The PLC can use only standard C++ libraries without any external dependencies. The communication between PLC and UI will be realized through shared memory. Shared memory is asynchronous, has little overhead, and is supported by all major operation systems. It is also inherently supported by TG Motion.

GUI

The first and most important thing to choose is the GUI framework. Based on my previous experience, I will avoid using C/C++ since it tends to be cumbersome and too complex for a GUI development. The advantage of C/C++ is its speed and multi-platform support which both are not a priority here. The preferred choice is a higher level language like C# or Java. Both languages offer a many GUI frameworks with the most notable being JavaFX, Swing, and SWT for Java and WinForms, and WPF for C#. All the frameworks offer a big set of standard GUI components out of the box. After doing a research and testing the frameworks, I found the WPF being the most favorable and advanced. It is a fairly new framework (considering all the possibilities) running on .NET Framework with a clean separation between look and functionality. Layout is defined in a separate declarative XAML file [10] which is connected with a class implementing the advanced view logic ("code behind"). It also offers an easy way to re-skin the application by using style resources (which work similarly to CSS files).

Extensibility

Based on the application requirements, there is a need for extensibility support. That means extending the features of the application by installing separate modules. The process of installing new module should be simple enough for a common user to handle. By default, the .NET Framework [11] supports extensibility by using DLL files. The libraries can be loaded into the hosting application to provide additional functionality. However, it still requires a lot of plumbing to integrate them with the application. Every module can provide services, forms, windows, or other controls. This is handled by the **Prism** library [12] for WPF which simplifies dll loading and management but also offers tools for GUI integration. Advantage of Prism is that it can also work with different GUI framework than WPF.

Rendering

The draw window showing machine activity is assumed to be the most performance demanding part of the application. The required performance scales with the size of the CNC programs, which cannot be anticipated. Best candidates for rendering the visual window are low level APIs like DirectX or OpenGL. The deciding advantage of DirectX over OpenGL is its support by WPF. DirectX can be directly used only from a native application which leaves with two options. Either write the window code in a separate native program and connect it via Interoperability or use a C# DirectX wrapper (like SharpDX [13]) and call the API directly. I have chosen the latter since it is the easier way and it does not require to create additional application. The trade-off is worse performance[1] but the impact shouldn't be significant. It will require performance tests in the later stages of the application to really assess this assumption. I also considered to use the drawing engine of WPF, but it turned out to be too slow when used with many objects. I experienced frame drops starting at around 10 000 lines.

Data Storage

Internal application data needs a way to be stored. Standard way of storing data is to use a database. An alternative is to develop a custom storage system but that would not bring any benefits. There are no special requirements by the application for the storage. It is a single user application with low amounts of data required to be stored and with minimal data traffic. Therefore every modern database would suffice. Due to my experience and broad use I chose to use a relational database over a non-relational.

Configuration

As a configuration storage I have chosen the standard .NET Framework configuration files. The framework provides a built-in way how to store application or custom configuration which can be also used for storing CNC program specific configuration files. It is more advanced than just a key-value file with possibilities of sections, custom datatypes, and additional tags per

[1]based on the benchmark http://code4k.blogspot.cz/2011/03/benchmarking-cnet-direct3d-11-apis-vs.html, SharpDX is 1.52 to 2.32 times slower than native.

record. The only downside is that it uses the XML format which is not very user friendly (compared for example to ini or JSON) when being edited directly, but that shouldn't be a common use case.

Chapter 6

Programmable Logic Controller

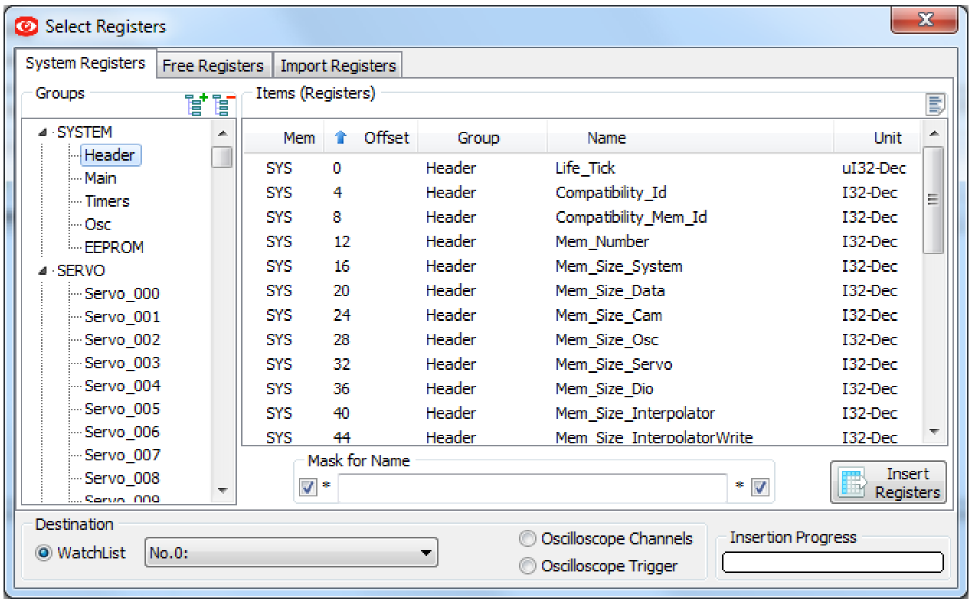
This chapter describes the structure and implementation of programmable logic controller (PLC). Unlike the rest of the Control System, PLC is implemented using C++. Structure of the PLC is also extensively different from the rest. Real-time programming diverts from standard programming paradigms. The program runs in cycles (like for example game engines do) and every action encompassing IO operations is asynchronous. Performance is critical to maintain a reasonable delay. Even duration of 1ms per cycle, which looks reasonable at first sight, is too slow. For example with machine speed of 1 m/s (a realistic cutting speed) the tool would travel 1 mm during the cycle. That is too much for shapes requiring frequent changes of speed of individual axes (like a simple circle).

6.1 Shared Memory

Shared memory is a block of memory that can be accessed by multiple applications simultaneously. The memory is used to control TG Motion and also to interchange data between PLC and UI. It is logically divided into sections, with each section having a different purpose. As an example, one section is reserved for controlling servo motors. The sections consist of variables, where each variable has a defined position in the memory, data type and size. The memory is not strongly typed, therefore it is needed to know an exact location and type of the variables. The memory layout is described in the TG Motion manual [6]. One section of the memory is reserved for communication between UI and PLC. The structure is left to the programmers of the PLC/UI. TG Drives also offers many utility programs for monitoring or accessing the shared memory. One of the tools is **Control Observer** which is shown in Figure 6.1. It has the ability to read/write memory blocks, see history of values in the course of time, etc.

6.2 Structure

The PLC needs to follow an exact structure required by TG Motion. It has to export six functions – **Program_Ini** and **Program_01** to **Program_05**.



Select Registers

System Registers | Free Registers | Import Registers

Groups

- SYSTEM
  - Header
  - Main
  - Timers
  - Osc
  - EEPROM
- SERVO
  - Servo_000
  - Servo_001
  - Servo_002
  - Servo_003
  - Servo_004
  - Servo_005
  - Servo_006
  - Servo_007
  - Servo_008

Items (Registers)

Mem	Offset	Group	Name	Unit
SYS	0	Header	Life_Tick	uI32-Dec
SYS	4	Header	Compatibility_Id	I32-Dec
SYS	8	Header	Compatibility_Mem_Id	I32-Dec
SYS	12	Header	Mem_Number	I32-Dec
SYS	16	Header	Mem_Size_System	I32-Dec
SYS	20	Header	Mem_Size_Data	I32-Dec
SYS	24	Header	Mem_Size_Cam	I32-Dec
SYS	28	Header	Mem_Size_Osc	I32-Dec
SYS	32	Header	Mem_Size_Servo	I32-Dec
SYS	36	Header	Mem_Size_Dio	I32-Dec
SYS	40	Header	Mem_Size_Interpolator	I32-Dec
SYS	44	Header	Mem_Size_InterpolatorWrite	I32-Dec

Mask for Name
☑ * * ☑

Insert Registers

Destination
● WatchList No.0:
○ Oscilloscope Channels
○ Oscilloscope Trigger

Insertion Progress

Figure 6.1: Control Observer – Tool for monitoring the shared memory. Currently showing the **System.Header** section of the memory. Notice the individual variables, their offsets and data types.

Program__Ini is called during the load of the PLC and should contain initialization logic. **Program__01 - Program__04** are called periodically. They only differ in priority, where the higher the number, the higher the priority. The period can be configured for each function and can range between 100-10000 μs. Functions with a lower priority can be interrupted by the ones with a higher priority. The functions with a higher priority can be used for manipulating the hardware, the lower priority ones for diagnostics. **Program__05** is called synchronously with CNC program execution and cannot be interrupted. This is the most performance critical function as it can be called multiple times per cycle. According to manual, it should not exceed 10 μs.

6.3 Manual Movement

Servo motors are controlled through the shared memory within **SERVO** section. There are multiple ways of controlling them. The most straightforward one is to directly manipulate the requested position. During each cycle, the requested position is sent to the servo which tries to reach it until the next cycle. This can be a bit tricky, as it is required to also handle acceleration and deceleration properly. For example, if the servo would be given a too big movement request, it would create a high strain on the outer HW which could lead to its damage. To overcome this problem, TG Motion offers a “Profile Generator”. It allows to easily control the movement of a motor by setting either the target position or the target speed. The rest is handled by the generator. The servos are controlled individually, therefore each can use

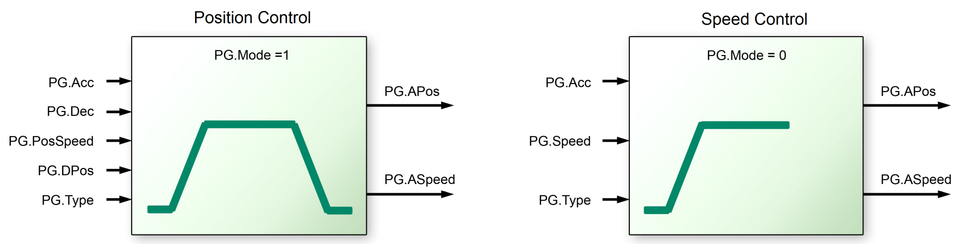


Figure 6.2: Profile generator – two possible modes of controlling the movement of a motor. **Acc** and **Dec** are allowed acceleration and deceleration. The **PosSpeed** is a maximum movement speed and **Speed** is a requested speed. **DPos** is requested target position. Type represents the shape of the ramp-up and ramp-down functions. Image taken from [6]

a different movement method. TG motion also offers a way how to group servos so they can be controlled as one. For example the big frame contains a servo on both sides, but logically it is just a single axis and their movement needs to be synchronized.

6.4 CNC Program Cutting

Load and execution of CNC programs is done through special library functions provided by TG Drives. They are called from the application UI (see Figure 7.2). When a CNC program is active, TG Motion calls the function **Program_05** synchronously with its execution. The function can alter the positions computed by TG Motion. It can also implement a custom G-code M functions.

One such example is a circular piercing. Usually, the G-code contains a piercing command, but it does not contain the actual motion of the nozzle performing the piercing to avoid pollution of the code. Also the ideal diameter and number of rotations can differ between materials which would make the CNC program bound to a specific material. The motion is therefore calculated procedurally by this function.

Chapter 7
User Interface Core

This chapter contains implementation details of important aspects of the UI's core (see Figure 4.1). The core is divided into packages (.NET namespaces), where each package represents a comprehensive block of functionality. Design of individual classes follows the traditional OOP principles. Even if the control system is a technical application, some of its principles are inspired by web applications, mainly by the Spring Framework.

7.1 Internal Mechanisms

Inversion of Control

The important business logic is implemented by services. The services use the *Inversion of Control* principle which is achieved by the *Dependency Injection*. Dependencies aren't managed by the specific services themselves but by a third party called *container*. The classes declaratively specify their dependencies (either to interfaces or classes) and the actual implementations are provided by the container during the runtime based on the configuration. This allows to exchange implementations without the receiving class knowing about it. It is a significant advantage during testing as we can easily supply mocks of the dependencies. It also allows the modules to override default their functionality by providing modified implementations of the services.

The dependency container used for this purpose is Managed Extensibility Framework [14] (MEF). The primary design purpose of MEF is extensibility, but it can be also used as a DI container. It has the ability to scan an assembly, discover the components, and then inject them through the dependency injection. It also provides a built-in support with Prism (see the next section).

Extensibility

Modules are separate packages that can enhance the application with additional functionality. They can range from a single dll library to a more complex structure. A module is recognized when it contains a class sub-classing the **ControlSystemModule**. It serves as a descriptor of the module and provides the core application with vital information about the module. Prism scans the “Modules” directory and automatically loads modules located inside. Out of the box, Prism supports the module discovery only inside a single directory. The ability has been expanded with the **SubdirectoryModuleCatalog** which searches also subdirectories.

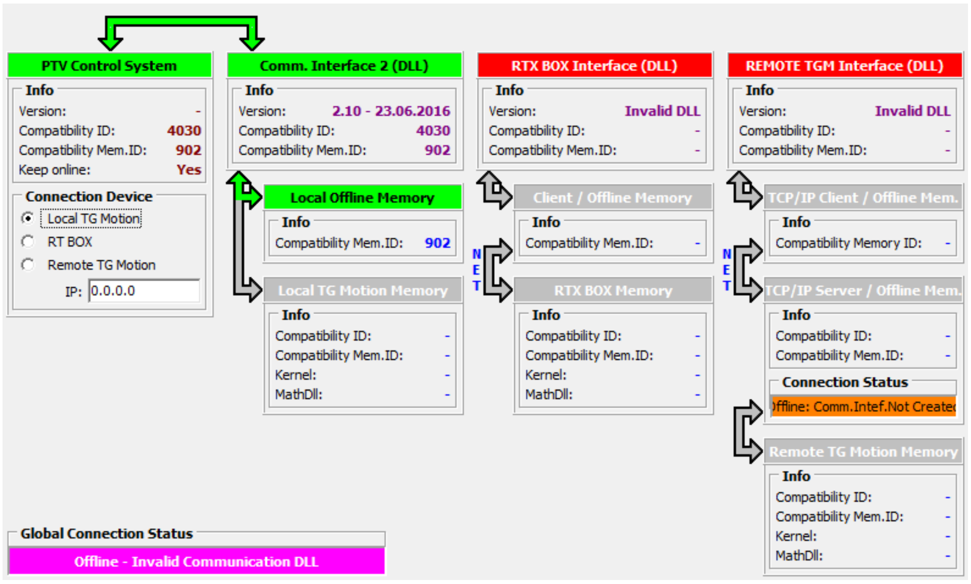
In the case of a more complex module, the module can have its own additional dependencies. That is a problem, because by default, .NET looks for dependencies in the location of the running application. But the modules are located in a subdirectory. That would require to copy all dependencies of the modules to the root directory which is unacceptable from a user perspective. The solution was to hook up the assembly resolve event of .NET, which is fired when the assembly dependency cannot be resolved. During this event, we can locate the dependency in the folder of the specific module and load it from there.

PLC Communication

The UI communicates with PLC through the shared memory. As the UI is not a part of TG Motion, it cannot work with the memory directly like PLC does. Also the memory is not managed by the .NET virtual machine which would make a direct access almost impossible. TG Drives company developed a library for communication across the shared memory for this purpose. It can establish a connection to the memory and safely read and write into it. Sadly, the library is written in a native code and therefore cannot be directly used from the application. I wrote a separate project **TG.Communication**, which wraps the native library into C# code using PInvoke [15]. The wrapper is called from C# and calls the native methods.

The read and write functions of the communication library can work with the common data types and always require an offset. The offset indicates location of the variable in the memory. This requires to use a list of variables and their respective positions which is a very functional approach. To make it more OOP friendly, I created a data structure **SharedMemoryDefinition**, which resembles the structure of the memory. The data structure is instantiated and proxied with a proxy (**SharedMemoryProxy**, implementation of a **RealProxy**) intercepting all reads and writes of attributes of the data structure. The proxy analyzes the structure of the object and calculates offsets for every attribute and when accessing them, it calls the shared memory read/write functions instead. Benefit of this approach is that implementation details of accessing the shared memory are hidden and outside classes work with the shared memory as with a simple object.

During application startup, connection to the shared memory is established. Then the PLC is loaded. Load of the PLC is done by writing PLC filesystem location to a variable in the memory. After that, TG Motion loads the PLC and signalizes success of the operation. Because the PLC is dependent on TG Motion, it will not be loaded when there is no TG Motion running. A lot of UI functionality depends on the PLC and its feedback. Therefore I created a “simulation” mode, which simulates job of the PLC and can be run without TG Motion (more in Section 7.1). Shared memory is still created for the sake



PTV Control System
Info
Version: -
Compatibility ID: 4030
Compatibility Mem.ID: 902
Keep online: Yes
Connection Device
Local TG Motion [✓]
RT BOX
Remote TG Motion
IP: 0.0.0.0

Comm. Interface 2 (DLL)
Info
Version: 2.10 - 23.06.2016
Compatibility ID: 4030
Compatibility Mem.ID: 902

Local Offline Memory
Info
Compatibility Mem.ID: 902

Local TG Motion Memory
Info
Compatibility ID: -
Compatibility Mem.ID: -
Kernel: -
MathDll: -

RTX BOX Interface (DLL)
Info
Version: Invalid DLL
Compatibility ID: -
Compatibility Mem.ID: -

Client / Offline Memory
Info
Compatibility Mem.ID: -

NET

RTX BOX Memory
Info
Compatibility ID: -
Compatibility Mem.ID: -
Kernel: -
MathDll: -

REMOTE TGM Interface (DLL)
Info
Version: Invalid DLL
Compatibility ID: -
Compatibility Mem.ID: -

TCP/IP Client / Offline Mem.
Info
Compatibility Memory ID: -

NET

TCP/IP Server / Offline Mem.
Info
Compatibility ID: -
Compatibility Mem.ID: -
Connection Status
Offline: Comm.Intef.Not Created

Remote TG Motion Memory
Info
Compatibility ID: -
Compatibility Mem.ID: -
Kernel: -
MathDll: -

Global Connection Status
Offline - Invalid Communication DLL

Figure 7.1: TG Motion memory connection info provided by the TG Drives memory connection library. TG Motion can be running locally or on a remote PC. The green arrow shows current communication link.

of consistency even without a running TG Motion.

Simulation Mode

The application is designed to run with the TG Motion Framework and with the connected hardware of a CNC machine. Lot of computations are performed outside of the UI or is gathered as a feedback from the machine's hardware. This would deny us to test some areas of the application without all of the mentioned tools available which would make some testing really cumbersome. Therefore I designed a simulation mode, which has the ability to simulate the behavior of these external tools without them being connected.

Many services, which are dependent on running TG Motion have their simulation counterparts. Their job is to mimic the real behavior, but of course it is not 100% identical. To switch to simulation mode, it is required to run the application with the "–simulation" parameter. Behind the hood, it replaces the services depending on external tools with the simulation services. This uses the dependency injection principle. Simulation services have a **SimulationAttribute** attribute which prioritizes them for injection during simulation mode.

7.2 CNC Programs and Cutting

Parsing a CNC Program

As it was explained in Section 2.3, the G-code exists in many dialects. To make the application independent of actual dialect, it uses an inner structure for representation of the G-code. Each type of G-code command has its class counterpart inheriting from **CommandBase**. The parameters of the commands are exposed through standard C# properties which makes work with them a lot easier. To create an inner representation from G-code, a parser is used. That is a class implementing the **ICncParser** interface. Each G-code dialect will have its own implementation of the interface. To support a new dialect just requires to write a new parser for it. The application currently supports the CncC886 version of the G-code[3].

Start Position Configuration

A start position configuration describes positioning of a CNC program in the inner area of the machine. Coordinates of a G-code are usually local, therefore without the start configuration the program would be always located around the lower left corner (as mentioned in Chapter 2, the machine coordinate system uses the right hand orientation). The start configuration contains a 3D translation and a rotation. The rotation is possible only around the Z axis. In a future, scaling might be also added. The user can store multiple start configurations which might help when switching between CNC programs. When displaying a program in the draw window, a transformation matrix is created from the start position configuration and used to move graphical representation of the program to its appropriate location.

CNC Material and Machine Configuration

The same CNC program can be cut into different materials with different thicknesses or by different machines. This needs to be handled by a configuration specific to a CNC program. We do not want the user to configure the same program every time it is loaded, because there is a big chance the same program will be cut repeatedly with the same settings. Therefore, it is a good idea to store the configuration with the CNC program itself. Because the ISO G code has no notion of configuration, we need to find a different way. I chose to store the configuration as a XML file with the same name at the same directory as the original program. When the program is loaded, its configuration is loaded with it. If the program does not have a configuration yet, a default one is created. Standard .NET configuration mechanism is used for storing the program configurations.

A CNC program for a water jet machine can have different settings from the one for a laser machine. Therefore, the configuration also needs to be modular. The idea is to split it into sections. The base application contains a basic section with common configuration for all cutting machines (like type of material). Then the water jet or laser specific module supplies its own section with settings specific to that machine (like for water-jet the cutting pressure during the cutting). When saving the configuration, the resulting XML contains all the sections.

CNC Program Cutting

When a program is ready to be cut, it is bundled with material and machine configuration and start position configuration and sent to **CncExecutor**. Handling and execution of the programs is performed by a separate library by TG Drives. Similarly to the shared memory communication (Section 7.1), it was required to create a wrapper **TG.CNC** to allow execution from C#.

First of all, the program needs to be compiled. The compile function accepts a raw G-code but the program is represented by a sequence of internal objects (Section 7.2), therefore it is required to create a G-code from it. Dialect of TG Motion accepted G-code differs a bit from the Cnc886, the class **TgCncFormatter** is responsible for creating appropriate G-code from the internal representation.

After the program is successfully compiled, its starting position and rotation is configured. There are no specific methods how to set it, but the library accepts a G-code prefix and postfix which is applied to every executed G-code command. The offset and rotation is therefore supplied as a standard G-code within the prefix.

After that, the program is executed and the rest is handled by TG Motion and PLC. Other methods offered by the library worth mentioning are calculation of execution time, starting the program from a custom location, and setting of breakpoints.

Manual Cutting and Movement

As stated in the Chapter 6, the UI should not manipulate the HW directly. Therefore, it only sends messages to the PLC which does the job. To make it easier, the service responsible for manual movement, **IManualMovementService**, has a very similar interface to a profile generator (see Section 6.3). It can move the head either to a target position or to a target direction with a constant speed. PLC then splits the requested movement to individual profile generators for every axis.

To start cutting, it is necessary to toggle valve of the beam. From the applications perspective they are represented as a single digital output (DIO). Digital inputs/outputs are also controlled through the shared memory and occupy a separate section. A signal is sent to the PLC which enables/disabled the respective DIO.

7.3 Other Areas

License Management

Licensing is a way of protecting the manufacturing company from usage of illegal copies of the software. To prevent SW misuse, licenses will be stored in a licensing server located at the manufacturing company. The application periodically downloads and refreshes the licenses from the server. If there is no connection to the license server for a certain period of time (currently set to 20 days), the current licenses are considered as expired and therefore invalid. This also provides a leverage for the manufacturing company to cut off non paying customers.

The client-server communication is realized through Windows Communication Foundation (WCF) [16]. It lets us design the communication as a simple interface and the actual connection code is automatically generated. Another advantage of WCF is that communication protocol can be replaced just through configuration without changing a single line of code. The server code can be hosted as a Windows Service or inside IIS[1] or similar web server.

Each application installation has its own app GUID – an unique identifier. When requesting licenses, the application presents itself with the identifier so the server can fetch the ones belonging to that application. A license is issued either to the base application (core) or to a specific module and contains the expiration date (which can be unbounded).

Licenses are checked during startup. If the license for core is invalid, the application shuts down immediately displaying the error message. If the license for a module is invalid, the module is prevented to load and is disabled. After the startup, licenses are periodically checked and if one is invalid, the application shuts down within a short period of time. This is to prevent avoiding license checking by letting the application open for extended amounts of time.

Up to now, the application uses plain text licenses which are too vulnerable to hacking. Before going to production it will switch to a more secure principle using encryption and keys as described in the article [17]. For a debug build, checking of licenses is completely disabled.

Internationalization

Internationalization (usually shortened as i18n) means adapting the application to different cultures. As perceived by many, i18n is not just about localizing strings. There are many more differences between cultures like reading direction, letter capitalization, text sorting etc. Some of these aspects are already handled by the .NET Framework or by WPF itself. Every .NET thread runs under the specific culture which is used as input argument for actions like text formatting. Other aspects need to be handled by the application itself.

The application will be usually run in a single language environment. But in some circumstances, there may be users with other preferences (I myself prefer using English in applications over Czech). To satisfy this need, the application has a default language setting and each user has an option to set his preferred language to override the global option.

Given the specific requirements, the application needs to support dynamic localization – the application boots up in its default language and when a user logs in, it switches to his preferred language (if any). The .NET offers the

[1]Internet Information Services – Microsoft's web server

ResourceManager class for localization. A resource manager encapsulates multiple resource dictionaries (sets of keys and values) and can fetch resource from the most specific dictionary given a culture.

When using directly the resource manager with WPF, it can perform only one time localization on startup. To overcome the limitation, I have created a central point for localization – **ILocalizationService**. The service keeps track of all resource managers used by the application and when providing localization, it selects the most appropriate one. When the current culture of the application is changed, the service broadcast a message to all localizable objects to refresh their localizations.

Permission Management

In compliance with the different roles mentioned in Chapter 3, the application needs to impose restrictions to what is available to individual users. A permission management system was developed to suit the needs. Every user will have its own account. Before being able to use the application, they need to log in using their username and password. Based on their privileges, they are be granted access to specific areas. The passwords are salted and hashed to ensure their protection.

I chose to employ a fine grained permission system. Each permission represents a certain action. Every agenda can require one or multiple permissions. The permissions are organized in a hierarchical manner, where having a parent permission gives automatically the access to all child permissions. For example there is a super permission for having access to all configurations and child permissions representing specific sections of the configuration (seen in Figure 7.2). This way, an administrator is able to edit all configurations but a standard user could have access only to one specific section. The permissions the user has are stored within his user object and can be changed by the administrator.

Data Storage

In the past, applications used direct SQL commands to access and manipulate data in databases. Modern way is to use an ORM (Object-relational mapping) framework. It allows us to work directly with classes (entities) which are mapped to tables of the database. The framework translates every action to SQL and updates the underlying database behind the scenes. The chosen ORM framework is Entity Framework Core [18]. It is a fairly new project by Microsoft succeeding older Entity Framework 6. It still lacks some features but it is now the preferred choice by Microsoft and therefore more promising for future.

The used database provider is SQLite because of one reason – its storage engine can be embedded directly into the application and therefore does not require a separate running daemon. This is a big advantage because it eases the installation process of the application.

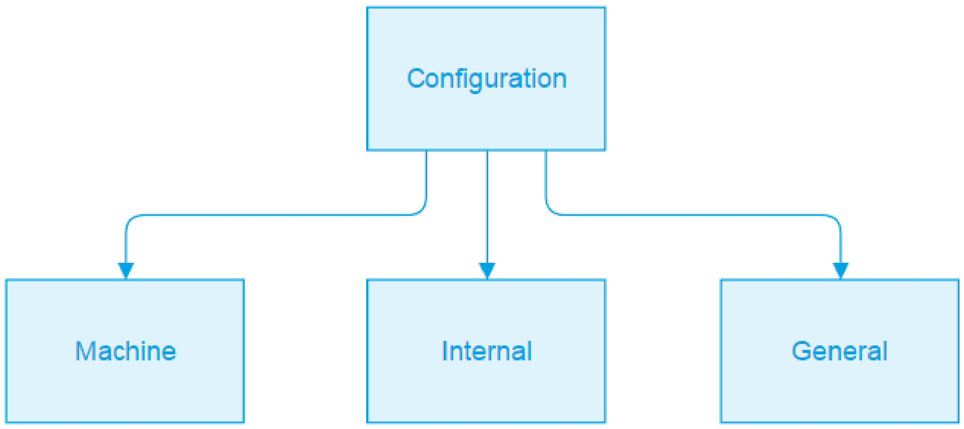


Figure 7.2: This image represents a hierarchical organization of permissions. The "Configuration" permission grants access to all its children, but the "General" permission does not grant access to the "Internal". In this example, each of the child nodes represents a permission to access a individual configuration section.

Every data access operation is executed through a **IDatabaseService** interface. Production implementation uses the SQLite storage engine to store data into a file located in the applications folder. Implementation for tests uses an in-memory database recreated every time the tests are run. This makes the testing easier and also ensures consistency between different test runs.

Application Configuration

The application stores its configuration through the .NET **ApplicationSettingsBase**. Each object can contain set of properties, which are stored as XML. It allows to store standard data types and when provided a **TypeConverter**, even the custom ones.

To leverage this functionality, I wrapped each **ApplicationSettingsBase** into a **ConfigurationSection** which contains additional information about the configuration (like the module it came from or required permissions). These are then registered into **IConfigurationManager** which keeps tracks of all configurations in one place. Thanks to that, all the configuration properties can be edited through a unified interface.

Events

Individual components and services need to share information between themselves. This is realized by broadcasting messages. It follows the principle of publisher – subscriber. The publisher publishes an event which is received by all subscribers.

The interconnecting class is the **IEventDispatcher**. It keeps track of all subscribers and their subscribed event types. Publisher can then publish the event to all subscribers through this class. The standard .NET events provide similar functionality but this approach has the advantage of the possibility of having multiple publishers for same event. The components are loosely coupled which increases maintainability (subscriber doesn't need to know who the publisher is). This system is inspired by Spring's application events.

Logging

Logging can be split into two separate categories – event logging and software tracing [19]. They differ mainly in their content and in their target audience. The first is aimed at system administrators and contains more high level events. The latter is used for debugging with more lower level information. Because this is not a typical application (there are no typical system administrators), I will make no distinction between these two. Primary aim of logging is to find the cause of an error. Because there are many external factors in play (like hardware), the log will record every user action to be able to reconstruct the exact sequence of events.

Logging is already well explored area and there is no need to re-invent the wheel. For this task, I have chosen the **NLog** library. The library offers rich configuration options for filtering the messages and for customizing their format. To avoid hard dependency on the library, there is a separate layer in-between. When logging, the application uses specific internal format (**LogEntry**) which is then passed to **ILogger**. Implementation of this class is an adapter between the application and NLog. It converts the messages and passes them to NLog logger. The separation would make exchange of the logging library easier in the future.

Chapter 8

Graphical User Interface

This chapter is dedicated to realization of the graphical part of the application. As opposed to Core, it contains only visualization and interaction logic. The GUI is implemented in a separate C# project. It uses WPF as the underlying GUI technology which is analyzed in Section 8.1 and in Section 8.2 demonstrated on an example. The Section 8.3 shows skinning capabilities of WPF and Section 8.4 shows its extensibility with the help of Prism. The last Section 8.5 examines the visual window.

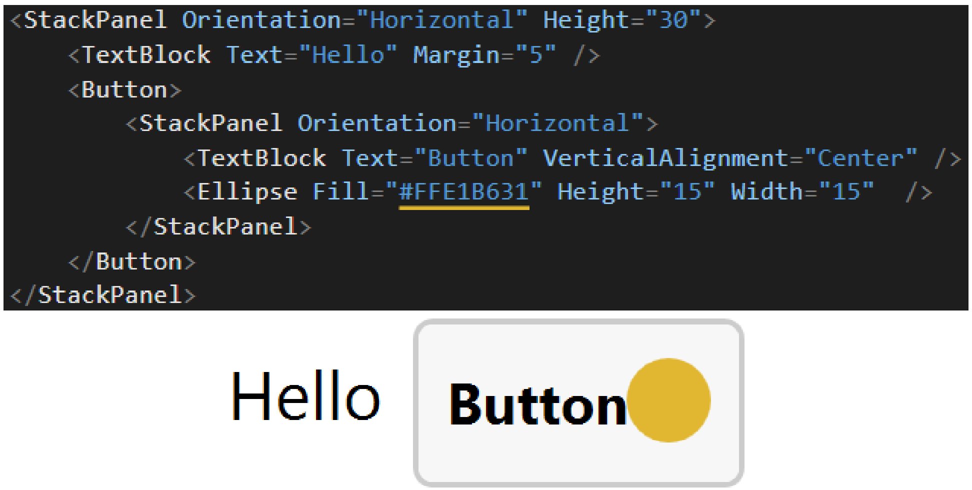
8.1 Windows Presentation Foundation

WPF is GUI framework developed by Microsoft available since .NET Framework 3.0. It makes creating user interfaces much easier as it handles a lot of common tasks like layouting or user input. Applications created by WPF can be run on desktop computers, but also for example on tablets with full versions of Windows. WPF rendering is vector based with resolution independence, therefore it does not experience problems with variously sized displays (as opposed to the previous technology, Winforms, based on GDI). WPF uses internally DirectX as rendering engine which makes it fast and responsive. Because of that, it can natively host custom DirectX content, which I used for the visual window (Chapter 8.5).

GUI created by WPF is a composite interface consisting of many independent components assembled together. A WPF component consists of a class and a XAML file which are bound together. The XAML file describes the visual side of the component, it can also contain other components. The class (or so called code behind) can contain an advanced interaction logic but because of richness of XAML it is often not required. The resulting hierarchy of components is called logical tree.

Interaction between components and business logic is handled through data binding and viewmodels. Every component has a property called data context where object called viewmodel can be plugged in. Operations or interactions of the component can be related to its data context.

Through **data binding** it is possible to exchange data or events between the control and its data context. For example we can have an object containing property Name and a text input. We set the object as the data context and



```
<StackPanel Orientation="Horizontal" Height="30">
    <TextBlock Text="Hello" Margin="5" />
    <Button>
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="Button" VerticalAlignment="Center" />
            <Ellipse Fill="#FFE1B631" Height="15" Width="15"  />
        </StackPanel>
    </Button>
</StackPanel>
```

Figure 8.1: Example of a interface defined in XAML and its result. Notice, how the button can also contain another components.

when the user enters text, it is automatically set to the Name property of the object through data binding.

A **viewmodel** is an intermediate class between UI and business logic. Its role is to transform data from business logic to the format required by the UI. It listens to events of the GUI and then sends changes back the the underlying business logic accordingly [20].

Older GUI frameworks used to lay out components by defining exact position in pixels from edges of parent component. That system is not very flexible and is unusable for complex interfaces. In the past, resolutions were not changed too much so it was sufficient. But today resolution range from small ones of phones and tablets to 4K of PCs.

WPF uses special components called panels for arrangement. Every panel uses a different strategy to lay out its children. WPF itself provides variety of panels which fit most common scenarios. The arranging process has two phases. In the first one, panels ask children how much space they need and tell them how much they can get. In the second pass, they assign the children a rectangle representing the resulting location [21]. This allows to write custom panels which can arrange their children in any fashion.

8.2 Example of Designing a Form

The GUI is assembled from many small parts. They are called user controls and usually represent a coherent block of functionality. They derive from the WPF **UserControl** class. Visual Studio offers a visual designer and additional tools to help designing the user controls. The Control System consists of many user controls, I will show the process of creating a control on one specific example. All the user controls used within the GUI reside in the **PTV.ControlSystem.GUI.View** namespace.

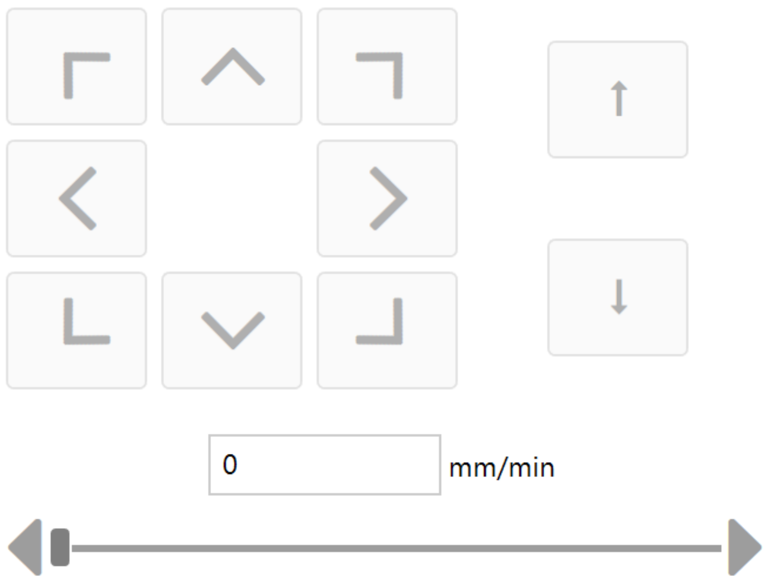


Figure 8.2: Manual movement user control. View taken from Visual Studio XAML designer.

For the purposes of demonstration, I chose the "manual movement user control". This control allows to manually move the cutting head. It is depicted in Figure 8.2. In the upper left part, the buttons move the head in X/Y plane. The two buttons on the right move the head in Z direction (up/down). The speed of the movement is controlled by the speed setter. It is a separate user control which is embedded into the manual movement user control. The control was previously described in Section 3.2 and Figure 3.6. The reference points cannot be seen on the image, because they are created during the runtime based on the configuration which is not available during the design time.

Basic layout of this control is done by a vertical stack panel. Stack panel stacks child controls behind each other. The diagonal movement buttons are wrapped in a uniform grid which has same width and height for every cell. The text box indicating currently selected movement speed is not a part of the speed setter. The current speed is provided as an output of the speed setter and is displayed in the text box with the help of data binding. The XAML of the control contains not only the structure of the control but also the styles used in the control. The styles define properties of the buttons (they all share the same size) but also their behavior like disabling them when the machine is executing a CNC program. The control has also its own view model, which sends the user commands to services in the Core module. For example, when the user presses the movement button, the event is routed to the view model which sends a signal to the **IManualMovementService** to start moving the head in a given direction and speed.

8.3 User Interface Stylization

WPF successfully separates appearance from behavior. The components can completely change their appearance while still maintaining their original behavior. In WPF, every control is defined by a class. The class implements all required behavior. For example, for a button it would be to perform some action when it is clicked.

The visual part is defined in a so called control template. A control template is defined in XAML and consists of multiple visual elements altogether defining look of the control. It can also contain visual states or animations, like changing color when mouse is hovered over the control. When creating a control, the developer usually supplies a default template. This can be overridden either globally for the whole application, or for just a visual subtree. For example, buttons inside toolbar have different appearance – the toolbar supplies its own control template for buttons which is then applied to all buttons inside.

So as explained, changing appearance of the application means just to use a specific set of templates. By default, WPF provides appearance matching version of Windows the application runs on. That means, if the application is run on Windows 7, it will look different from one run on Windows 10. As this may be good for an office application to have appearance the user is used to, for a technical application it is better to stick with one visual across all platforms. Therefore, I used templates from an open source framework **MahApps.Metro** [22] providing appearance similar to the modern Windows 10 Metro theme. If the appearance should change in the future, it would just require to change the template library. The Figures 8.3 to 8.5 show look of the application with different template libraries.

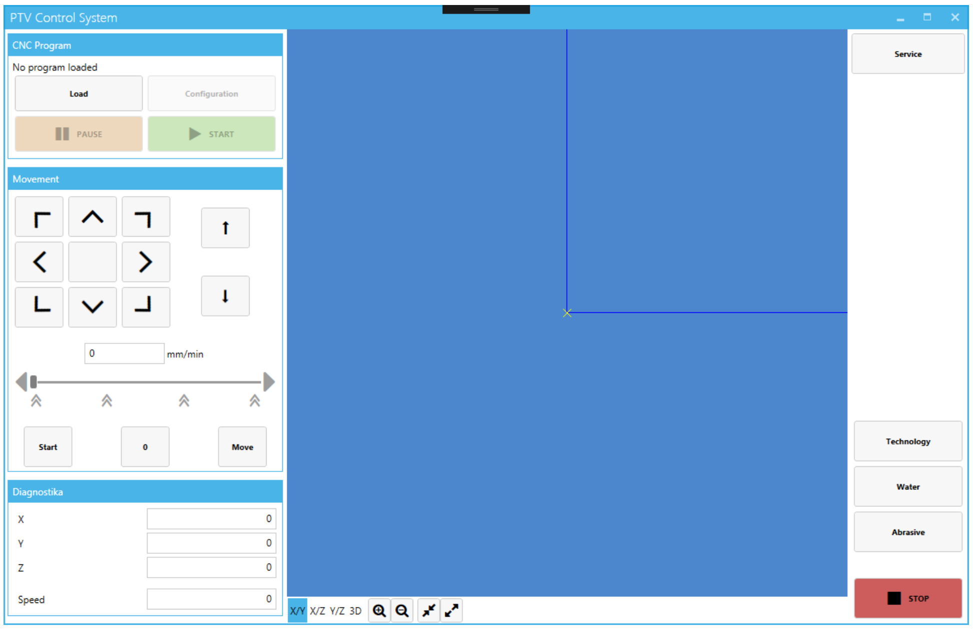


Figure 8.3: MahApps.Metro Theme

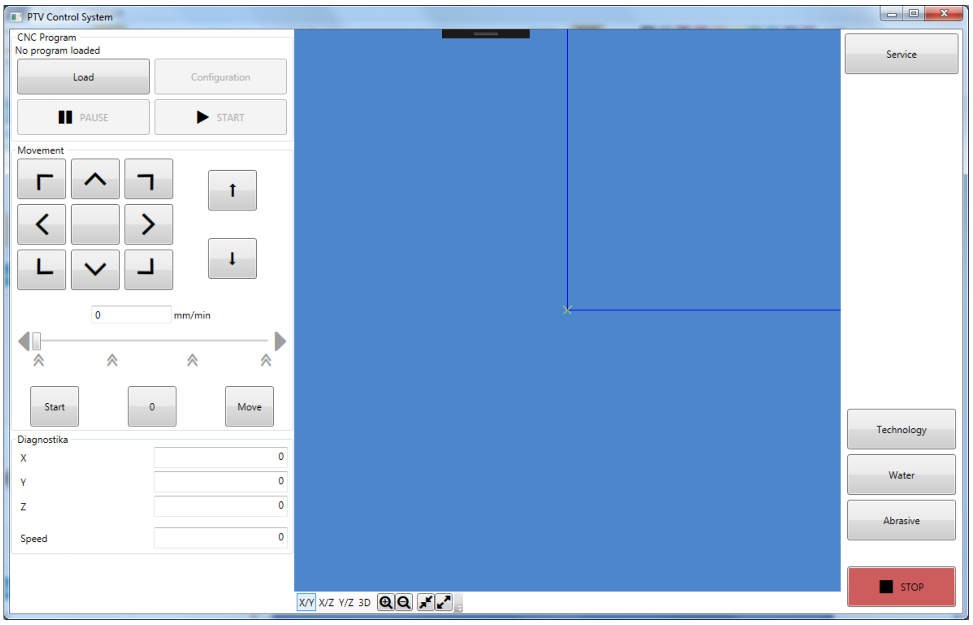


Figure 8.4: Windows 7 Theme

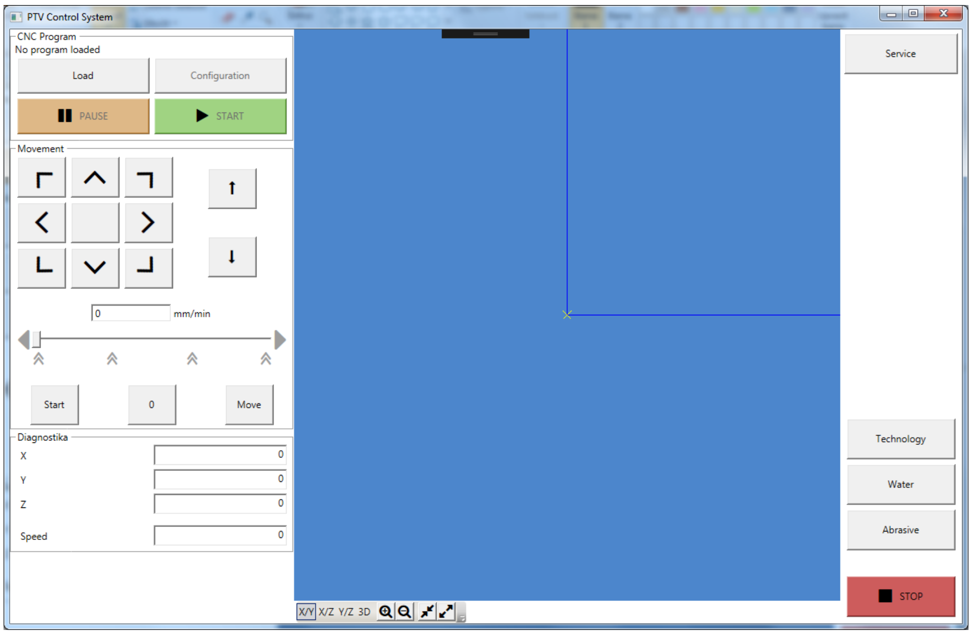
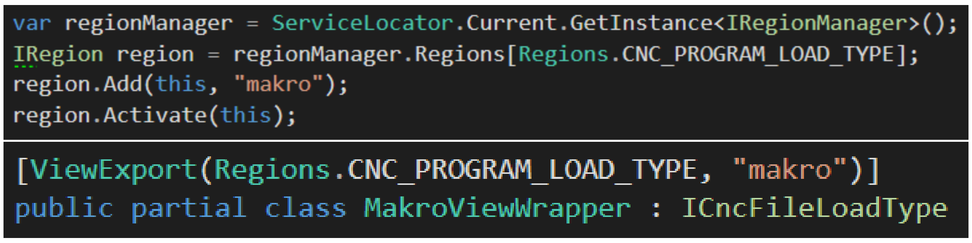


Figure 8.5: Windows 95 Theme

8.4 Extensibility

Extensibility of GUI means the ability to dynamically add new controls to it. In Prism, components that can dynamically host the views are called **regions**. Each region is uniquely identified by its name. The region knows how to display the hosted views. The traditional example is a dashboard page – a dashboards are usually shown when users opens a application and summarizes some important information. When a specific module is installed, it can provide its own dashboard which is then shown on the main dashboard page. The dashboard page, the region, knows how to lay out all the dashboards. It can show them in a grid, stack them vertically or show one at a time with arrow for switching between them. It all depends on implementation of the region. When using prism, every standard WPF panel can be used as a region.

The most important panels of the application are marked as regions. One example is the “service” tab which can be extended from the modules. To add a view to a region, it is first required to explicitly obtain a reference to a region manager, extract the region from the manager and then add the view to the region. I simplified this process. Now it is only required to mark the view with **ViewExportAttribute**. The attribute contains a target region for the view. When Prism loads a module, it searches for all classes with the attribute and automatically adds them to their target regions. The difference between both approaches can be seen in Figure 8.6.



```
var regionManager = ServiceLocator.Current.GetInstance<IRegionManager>();
IRegion region = regionManager.Regions[Regions.CNC_PROGRAM_LOAD_TYPE];
region.Add(this, "makro");
region.Activate(this);
```

```
[ViewExport(Regions.CNC_PROGRAM_LOAD_TYPE, "makro")]
public partial class MakroViewWrapper : ICncFileLoadType
```

Figure 8.6: In the first example, the view has to be explicitly added to the region. In the second one, it is only required to annotate the class of view with an attribute and the view is automatically created and added to the region.

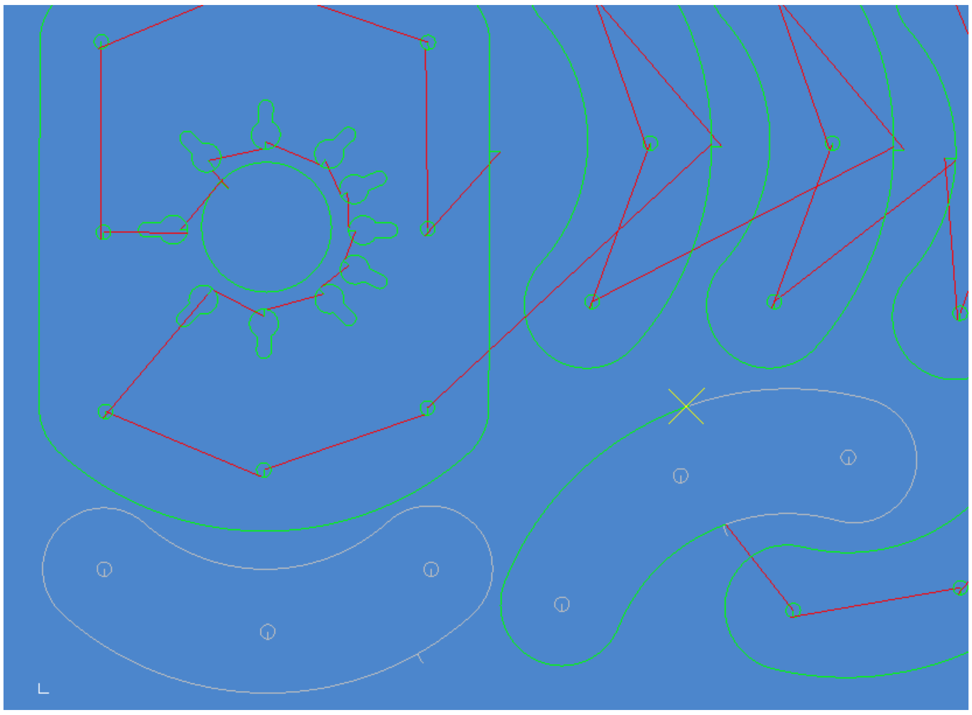


Figure 8.7: Visual Window – a cut-out showing a CNC program in progress

8.5 Visual Window

Visual Window is a component displaying currently running processes on the machine. It can show the path of a loaded CNC program and its progress, start configuration, and position of cutting heads. Currently, all the displayed objects are made of lines, no polygons are used as no filled shapes are required. The objects use a proper tree-dimensional cartesian coordinate system and are rendered through an orthographic camera.

8.5.1 Displaying a CNC Program

A CNC program consists of a sequence of G-code commands. They are far from being able to be conveniently displayed on a screen. The G-code commands are parsed into an internal structure which is more machine friendly. Now we need to create a sequence of lines to represent the shape of the program. For some G-codes, like linear move, it is really straightforward. The rest of the movement commands needs to be converted to lines to be able to be displayed by the DirectX API. The conversion can be configured for a maximum error difference from the original shape. Some of the important M commands, like piercing, also are represented in the window.

8.5.2 Showing Progress of a CNC Program

When cutting the program, the already cut path is distinguished from the path still needed to be cut. As seen on the Figure 8.7, already cut path has a greyish color, the path remaining to be cut has a green color. To be able to distinguish between them, each segment of the path is assigned an increasing ID. When cutting, the service executing the cutting process keeps track in which section of the program the process currently is. This value is fed to the shader drawing the program where it is compared to the ID of the segments.

8.5.3 Optimizations

When testing the visual window, I was able to get a stable framerate for the program consisting of 200 000 lines, which is much more than a common program would usually need. From the set of real life programs I received for testing, the biggest one had around 9 000 lines. However, if optimization would be required, these are the possible enhancements.

Some CAD programs tend to export circles as a sequence of lines. But number of lines per circles may be too fine grained. This holds true also for export of curves. As the high precision might be good for the actual cutting, for displaying it is usually unnoticeable. Before the actual displaying, the program can be pre-processed and number of lines decimated by an approximation of the resulting shape with a specific maximum error. Approximation of shapes consisting of lines with a specific error margin can be solved with the shortest path algorithms [23]. This would complicate the calculation of IDs for the segments because one segment is equal to one G-code command in the current implementation.

The detail of the model differs at different zoom levels the operator selects. The previous technique can be used to build multiple levels of detail (LOD) of the shape. The more further the camera would be zoomed, the less detailed LODs would be displayed.

If the window is zoomed in, lot of parts of the program could be outside of the visible frustum. But they are still sent to the GPU and they consume considerate amounts of computational resources before they are culled. A structure like octree or bounding volume hierarchy would solve the problem by showing only showing the visible ones. If we break the cutting path into this structure, we can feed the GPU only the visible or partially visible segments.

Chapter 9
Makro Module

The makro module adds the ability to load CNC programs from the library of predefined shapes. Normally, a shape needs to be drawn in a CAD program and then exported to the CNC program. This takes some time, even more if the operator cannot do it himself. The makro library contains many common shapes which can be configured and cut directly eliminating the need of a CAD program.

When the module is enabled, selection of loading a CNC program is extended with a new tab containing the macro shapes (Figure 9.1). When the user selects a shape, detail of that shape is shown (Figure 9.2). Here he can configure the dimensions of the shape or switch the representation of the shape – some shapes allow to be switched into an inner representation[1]. This is done by clicking on image of the shape. When finished, the module creates a CNC code from the shape which is loaded into the control system.

The makro module was created as a proof of concept for modularity of the control system. It is based on a standalone application which generated CNC code from the shapes and saved it to disk. The program was developed by me a few years ago as a contract for the PTV company. The original program was written in C++ and used Win32 API for its GUI. I rewrote it into C# and WPF. This allows the same code base to be included into the Makro module and still be used as a standalone application. The Makro module only embeds the executable and configures it to suit the needs of the control system's API.

In the Makro executable, every shape is generated by a separate method. The method contains a geometrical representation of the shape. When supplied with desired dimensions, an intermediate class transforms the geometric representation into the actual CNC code. The thumbnails of the programs as seen in the selection (Figure 9.1) or configuration (Figure 9.2) are static images previously exported from Inkscape.

[1]Inner representation creates a hole – the cutting beam is inside of the shape

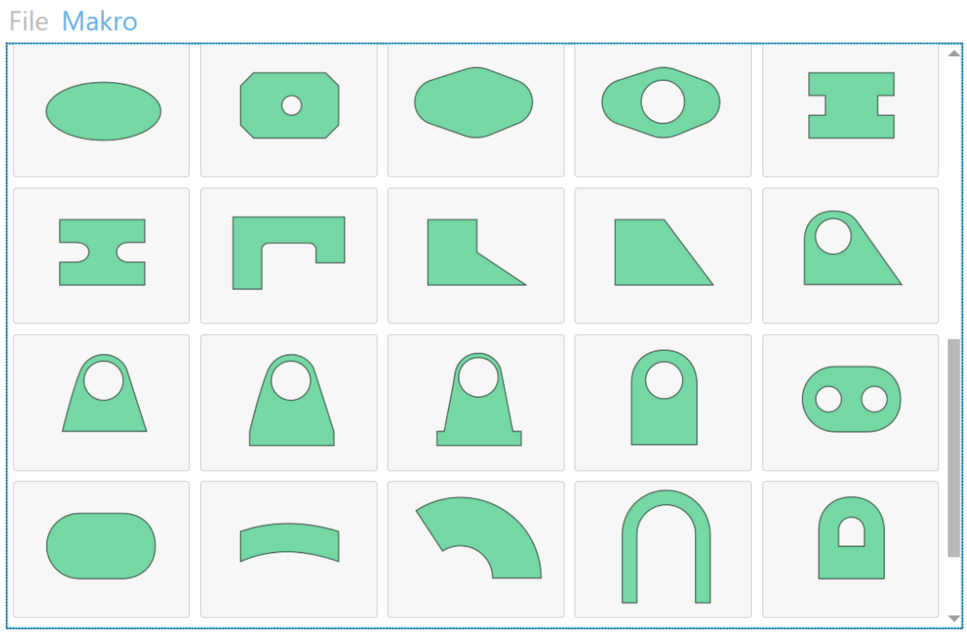
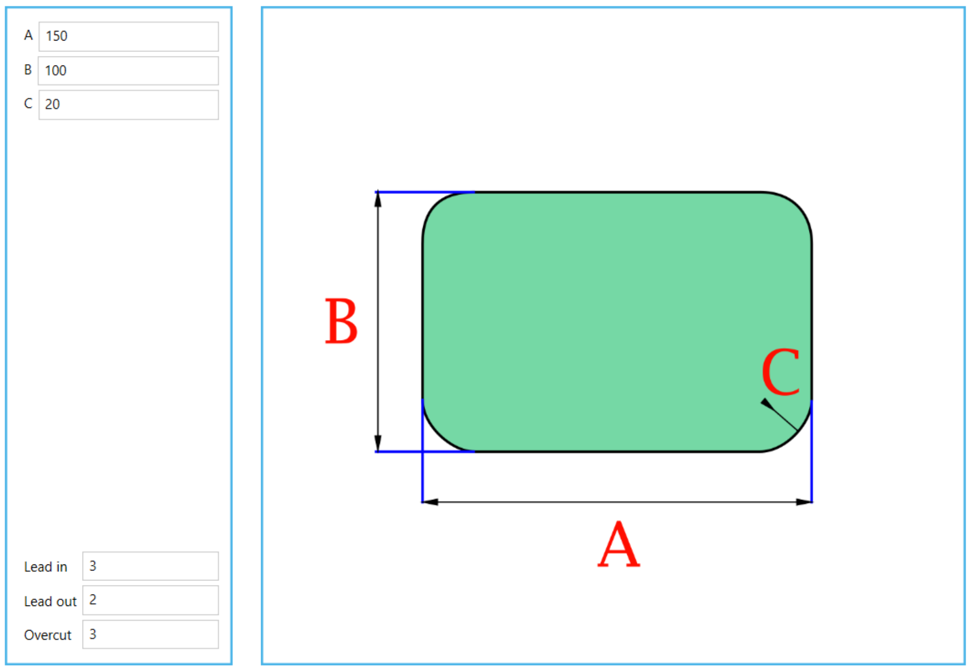


Figure 9.1: Makro – shape selection



A 150
B 100
C 20
Lead in 3
Lead out 2
Overcut 3

Figure 9.2: Makro – shape configuration

Chapter 10

Testing

Testing is an important part of development life cycle of every application. It differs in focus and how it is performed. Usually its aim is to ensure the application is bug-free. But there can also be a different reasons, like testing the ease of use or responsiveness of the application. In this chapter I will explain the techniques used for testing the application and their results.

10.1 Functionality Testing

Intention of bug testing is to find bugs affecting the application. The simplest form – manual testing, is used probably during the development of every application. It provides a simple way to check correct functionality. But its drawback, a need for a real tester, is pretty severe. Therefore it is being used in conjunction with other automated techniques. Bug prevention is important, as discovering bug in early stages of development can save a lot of effort in future. This holds thousand times for this application, as a critical bug could cause damage of HW or even worse, an injury. Automatic testing does help a lot with this issues, as the tests can be run automatically after every build. So if we introduce an error, we instantly know where it comes from. Here I list all techniques used for automated bug testing.

10.1.1 Assertions, Code Contracts

Assertions are statements used inside the real code which should always hold true. They can check conditions which cannot be expressed by the used programming language. As an example could be null checking. C# doesn't allow to define a non-nullable object. But it is often a requirement of methods that parameters cannot be null. This can be checked by an assertion, which would indicate a bug in case of a null parameter. Big advantage of assertions is that they can be turned off, as their evaluating uses CPU power which would slow down the production code. Assertions are used mostly in private classes or methods to verify correctness of the program logic.

Code Contracts [24] enhance this idea even further by allowing more expressiveness. One can define **preconditions**, **postconditions**, and **invariants**. These are then compiled into the resulting code. Apart from runtime checks, there are also tools which can take advantage of the conditions. Some of the expressions can be evaluated by a static program analysis. Code generators can include these conditions in a code documentation. As with the assertions, they can be also turned off. Code contracts are used mostly in public interfaces to verify correct input and output parameters.

10.1.2 Unit Tests

During **unit tests**, components are tested in isolation. The idea is to test if the given component is behaving as expected without outside influences. In practice, each unit test tests a single method of a class. This has the advantage over testing a class as a whole, if something breaks, it is easier to find the root of the problem. To properly isolate a class, its dependencies need to be mocked. Mocking allows to simulate behavior of a object in a deterministic way.

In the application, tests are implemented in a separate project from the production code. The reason behind this is to avoid including the tests into production assembly. The unit tests use a testing framework provided by Visual Studio [25]. Code coverage differs by specialization of given class. The more used core parts get better treatment than more specific classes. GUI has no unit tests at all as writing tests for GUI tends to be much harder. For example the geometric library (**PTV.Geometry**) has around 80% code coverage[1].

10.1.3 Integration Tests

Focus of **integration tests** is the opposite of unit tests – they tests interaction between components. Their intention is to expose defects in communication, when a component is expecting different data than what is being provided. They should be run after unit tests when we are sure the individual components work properly.

Integration tests usually need to prepare the whole run environment of the application. For example the application creates an in memory database every time integration tests are run. This makes them much slower opposed to unit tests. If they would be executed after every build, it would increase build times which would lead to frustration of developers. Therefore it is a good practice to separate them from unit tests and let them run on a separated machine in defined intervals, like every day during the night.

In the application, integration tests are implemented at the same level as unit tests. They are implemented only for a few higher level components. They are not a part of any automated process and need to be run manually.

[1]As calculated by Visual Studio Code Coverage tools [26]

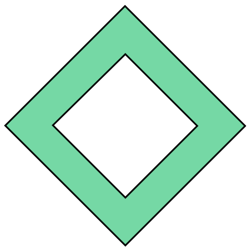
10.2 Usability Testing

The aim of **usability testing** is to evaluate the ease of using an application. It is performed by real people who perform typical tasks while being observed. I have prepared seven tasks and tested them on two participants with different experience in the field. Goal of the testing session was to identify problems with usage of the user interface.

10.2.1 Testing Scenarios

Before the actual testing, the participants were given a quick overview about basic functions of the interface. The participant without previous experience with CNC machines was also given a short training in this area as it is necessary a prerequisite. The testing environment was prepared with all below mentioned files ready in their places. A super admin account with all permissions was used for the testing. Because of practical and safety reasons, testing was performed using only the simulation mode on a common PC without actual cutting machine. Therefore when evaluating the results, we need to keep in mind that feedback from the machine was missing. Here a list of scenarios, which performed by the participants:

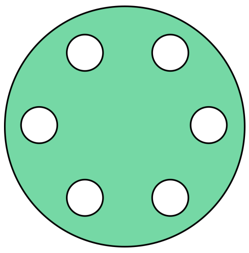
1. **Loading the CNC program from file** Load the CNC program located in *D:/CncFiles/Customer/test_program.CNC*.



2. **Configuring the CNC program** Configure the program to be cut on 2mm thick steel material with the cutting best quality. Use circular piercing with 1s duration.
3. **Setting a start configuration** Position the program to be cut within the material sheet. The sheet is 500mm x 500mm rectangular shape and is located within the machine with its left bottom corner at the 200X 100Y location. Rotate the program 45 degrees so it aligns with axes of the sheet.
4. **Executing the CNC program** Start the program. After the first hole is cut, pause it before it starts cutting again. After a brief delay continue the cutting and let it finish[2].

[2]This mimics a real life situation when the operator is required for example to refill the cutting abrasive material

5. **Loading CNC program from a makro** Load the makro depicted on the image. Dimensions of the resulting shape should be 100mm diameter of the outer circle and 6 inner circles with 20mm diameter.



6. **Manually cutting a rectangle** Navigate to approximately center of the machine (imagine, there is somewhere a material sheet). Manually (without using a CNC program) cut a small rectangle (around 50mm length of edges). The speed while cutting needs to be approximately 1000mm/min.

7. **Administration of the application** Create a new user account for a new machine operator. Try to log in as the new user. Add a new material "Leather" with machinability set to 450. Install a new module and enable it. The module is located in *D:/Example.dll.*

10.2.2 Testing

Participant 1 is experienced worker within the field of CNC cutting machines. He has previous experience with using the AremPRO control system and is highly skilled with computer software. He had no troubles performing the first two tasks. The respective buttons were quickly found. For the third task, he did not know how to set the starting point. After showing the button he knew how to configure the starting point as the form is similar to the one from AremPRO. The tasks 4 and 5 were also performed without problems. A remark was given, that the makro is missing units for the dimensions but he assumed mm as it is a standard unit in the engineering field. The task 6 was also performed without problems. The speed setter was praised, however it was advised to show speeds for the various reference points. Task 7 was also performed without troubles. The participant was just confused, why nothing appeared when he enabled the module (a restart is required for the change to take effect).

Participant 2 is completely new to the field of CNC machines, has average experience with computer software. The first task was achieved without bigger problems. The subject was first disoriented about what is a folder and what is a file during navigating the file system. He managed to finish the second task, but was confused with setting the piercing type. He did not know, why he can set configuration for all piercing types even when only one piercing type can be selected. He also did not know the difference between them. The third task caused him a lot of pain. As with the first participant, he could not locate the appropriate button. According to his words, the form is too complicated and was very hard to set the starting position because he could not see the sheet on the screen. During the fourth task, he stopped the program instead of pausing, which aborted the process and it was required to start it again. The fifth task was performed without any troubles. In the sixth task he started to press the movement arrows but nothing was happening. After explaining that it is required to set movement speed with the speed setter, he managed to fulfill the task. He added a suggestion, that the already cut area could be shown on the graphical window. Task 7 was completed without troubles. He noticed an inconsistency that the materials are saved automatically, but users have to be explicitly saved by pressing a button.

10.2.3 Evaluation

The testing sessions clearly show a huge difference between previously experienced and totally new users. Participant 1 experienced no big obstacles during performing the tasks. He was able to apply previous experience and also praised the simplified interface compared to AremPRO. However the second participant was insecure and got stuck several times. It is required to note, that a CNC machine is an expensive equipment and not available for broad public. We must assume, that every person using the machine must undergo at least some basic training. When applied to the second participant, some of the things he got stuck on would be explained during the training. However some controls, like the start configuration form are still too complex and can be overwhelming for a new user. Overall aim should be to make the interface more clear for new users. Here is a list of improvements which are a result from the testing sessions:

- CNC program load – Distinguish files from folder in the filesystem browser by using icons.
- CNC program configuration – Sort list of materials, do not show configuration for non-selected piercings, add help explaining individual piercings.
- Start configuration – Add label to the button, simplify the form.
- Makro – Add measurement units.
- Speed setter – Give the lowest selectable speed some meaningful value, add labels.
- Draw window – Add layer showing manually cut trajectory.
- Modules – Show a warning that a restart is required.
- CNC program execution – Allow continuing of the program even when stop is pressed.

Chapter 11
Discussion

The chapter Discussion is devoted to topics which could not fit into the main course of the thesis.

11.1 Clarity of Visual Window

The visual window displays trajectory of CNC programs, position of cutting heads and points of interest like piercing locations or the starting point. All the elements are drawn using 1px wide lines. This makes some elements hard to see, mainly if the operator is farther from the screen. I could try to improve it by increasing drawing width for the trajectory. On the other hand, this would worsen dense programs as the trajectories could overlap due to the high width. Maybe the best solution is to use dynamic width based on the trajectory.

Another improvement can be done to distinguish the already cut path more from the uncut one. Both paths are drawn using different colors but the difference is not entirely visible at first glance. This can be achieved by increasing contrast between the colors. The chosen colors need to be sufficiently contrasting but they still need to fit the the style of other elements in the window.

11.2 Automation of the Cutting Process

The cutting process is very dependent on the operator controlling the machine. He has to load, place, and configure the programs, select and exchange the material sheets. If we look at other production areas like car manufacturing, their processes are usually fully automatic. Why is it so different from water-jet/laser cutting? In an ideal world, the operator would only submit the part blueprint and target material and the rest would be handed by the machine automatically. From loading the material, configuring the program based on the material, selecting a proper starting position to cutting the program and putting the resulting part to an output tray. Many of the tasks would be solved by proposed features from the Section 3.3.

11.3 Current State of the Work

As it stands now, the software does not cooperate 100% with the hardware. When I was testing cutting of a CNC program, the servomotors got stuck in one moment. I believe, this is a synchronization issue caused in the PLC. However, when fixing the bug, it is required to proceed with caution, as the machine is very expensive and bugs like this can lead to permanent damage. After I finish the thesis, I will continue to investigate the cause of the problem.

Many internal machine parameters are currently wired into the program. In the future, they need to be configurable from the user interface as different machines require different configuration for ideal performance. They are currently optimized for the setting I used during development.

Chapter 12
Conclusion

As a part of the thesis, I have developed a new control system for CNC cutting machines. The main features of the system are the ability to manually operate a CNC cutter, automatic cutting of G-code programs, and a library of simple shapes. Notable advantage of the system is its extensibility, which, without interfering with the application base, allows to add new functionality or modify the existing one. The thesis focuses on the user interface which was created during several iterations. Based on the results of the user testing, the interface is simpler and more accessible compared to other analyzed systems. The thesis is not only a theoretical work, but its results will be used in practice.

12.1 Plans for the Future

The application is currently in its first version. Before it is completely ready to be distributed to clients, a phase of intensive testing and polishing needs to follow. The application is for many customers the only thing they come into contact with, therefore it will form their opinion towards the manufacturing company. After the thesis is finished, the system will be subjected to extensive testing under conditions simulating a common workload. In the thesis, I proposed a lot of new features which would improve effectiveness and make the work with the application more enjoyable. It depends only on the management of the company, which features will be implemented in the future.

Bibliography

[1] What is water jet, [Online]. Available: http://www.waterjets.org/index.php?option=com_content&task=category&sectionid=4&id=46&Itemid=53 (Accessed 04/05/2016).

[2] Water jet basics, [Online]. Available: http://www.ptv.cz/comparison-of-cutting-efficiency-depending-on-the-pressure-and-amount-of-abrasive-material/ (Accessed 28/05/2016).

[3] *Operation of the Cnc886/Win Control System*, 4. revision, AREM PRO, s.r.o., Nuslova 2275/15, 158 00, Praha 5 Nové Butovice, 2010. [Online]. Available: http://www.arempro.cz/resources/cnc886win_en_20101207.pdf (Accessed 19/04/2017).

[4] Iso 6983, [Online]. Available: https://www.iso.org/standard/34608.html (Accessed 19/04/2017).

[5] TG Motion, [Online]. Available: http://www.tgdrives.cz/en/control-systems-pc-and-panels/tg-motion/ (Accessed 04/05/2017).

[6] *TG Motion, návod k obsluze*, 4th ed., TG Drives, s.r.o., Olomoucká 79, 627 00, Brno-Černovice, 2016.

[7] B. Sandu. The 80/20 rule that you should know as a designer, [Online]. Available: http://www.designyourway.net/blog/inspiration/the-8020-rule-that-you-should-know-as-a-designer/ (Accessed 18/04/2017).

[8] What is nesting, [Online]. Available: https://en.wikipedia.org/wiki/Nesting_(process) (Accessed 18/05/2017).

[9] What is a software architecture?, [Online]. Available: https://www.ibm.com/developerworks/rational/library/feb06/eeles/ (Accessed 11/04/2017).

[10] XAML, [Online]. Available: https://msdn.microsoft.com/en-us/library/cc189036(VS.95).aspx (Accessed 24/04/2016).

[11] .NET framework, [Online]. Available: https://www.microsoft.com/net/ (Accessed 24/04/2016).

[12] Microsoft prism library, [Online]. Available: `https://msdn.microsoft.com/en-us/library/gg406140.aspx` (Accessed 24/04/2016).

[13] Sharpdx, [Online]. Available: `http://sharpdx.org/` (Accessed 21/04/2016).

[14] Managed extensibility framework, [Online]. Available: `https://msdn.microsoft.com/en-us/library/dd460648.aspx` (Accessed 11/04/2017).

[15] Microsoft. Calling native functions from managed code, [Online]. Available: `https://msdn.microsoft.com/en-us/library/ms235282.aspx` (Accessed 04/05/2017).

[16] Window communication foundation, [Online]. Available: `https://msdn.microsoft.com/en-us/library/bb332338.aspx` (Accessed 03/04/2017).

[17] Licensing solution, [Online]. Available: `https://www.codeproject.com/Articles/996001/A-Ready-To-Use-Software-Licensing-Solution-in-Csha` (Accessed 03/04/2017).

[18] Entity framework core, [Online]. Available: `https://github.com/aspnet/EntityFramework/` (Accessed 03/04/2017).

[19] Logging, [Online]. Available: `https://en.wikipedia.org/wiki/Tracing_(software)` (Accessed 03/04/2017).

[20] Implementing the model-view-viewmodel pattern, [Online]. Available: `https://msdn.microsoft.com/en-us/library/ff798384.aspx` (Accessed 19/04/2017).

[21] WPF - layout, [Online]. Available: `https://msdn.microsoft.com/en-us/library/ms745058` (Accessed 19/04/2017).

[22] Mahapps.metro, [Online]. Available: `http://mahapps.com/` (Accessed 15/04/2017).

[23] P. Šůcha. Combinatorial optimisation, the shortest paths in a graph, [Online]. Available: `https://cw.fel.cvut.cz/wiki/courses/a4m35ko/start` (Accessed 17/05/2015).

[24] Code contracts, [Online]. Available: `https://msdn.microsoft.com/en-us/library/dd264808.aspx` (Accessed 11/04/2017).

[25] Unit test basics, [Online]. Available: `https://msdn.microsoft.com/en-us/library/hh694602.aspx` (Accessed 11/04/2017).

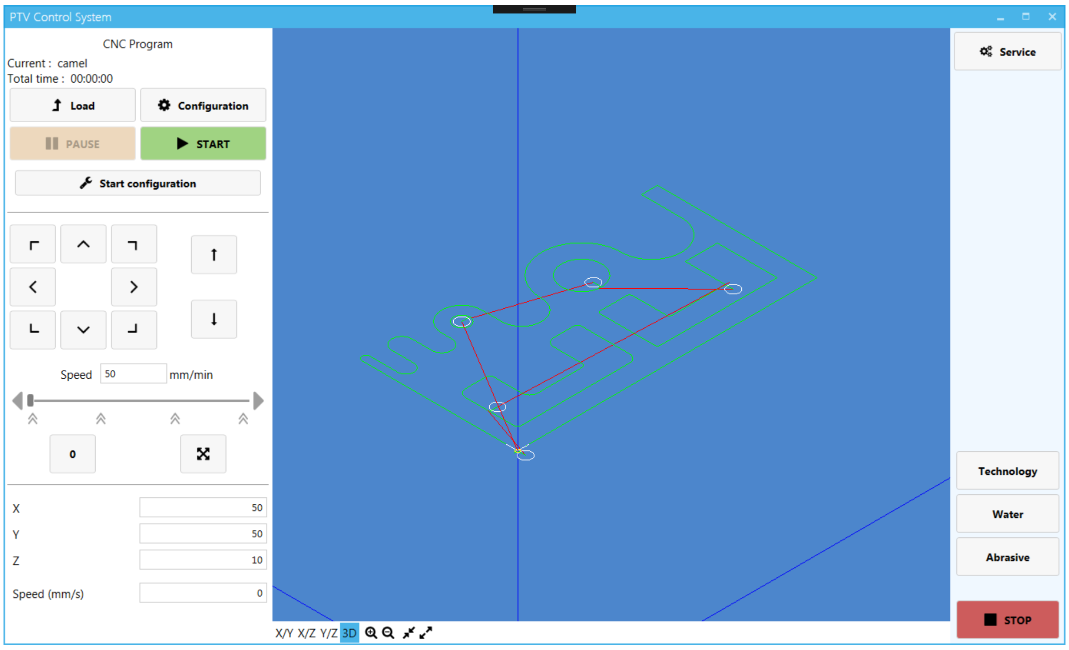
[26] Visual studio code coverage, [Online]. Available: `https://msdn.microsoft.com/en-us/library/dd537628.aspx` (Accessed 13/04/2017).

Appendix A
Abbreviations

API	Application Interface
CAD	Computer-Aided Design
CLI	Command Line Interface
CNC	Computer Numerical Control
CPU	Central Processing Unit
DI	Dependency Injection
DIO	Digital Input/Output
DLL	Dynamic Link Library
DRC	Dynamic Radius Control
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HW	Hardware
ID	Identifier
IDE	Integrated Development Environment
IoC	Inversion of Control
JSON	JavaScript Object Notation
MEF	Managed Extensibility Framework
MVC	Model View Controller
MVVM	Model-View-Viewmodel
NC	Numeric Control
OOP	Object Oriented Programming

ORM	Object-Relational Mapping
OS	Operating System
PLC	Programmable Logic Controller
PTV	PTV, spol. s r.o. Company
SQL	Structured Query Language
UI	User Interface
XAML	Extensible Application Markup Language
XML	Extensible Markup Language
WCF	Windows Communication Foundation
WPF	Windows Presentation Foundation

Appendix B
Additional Screenshots

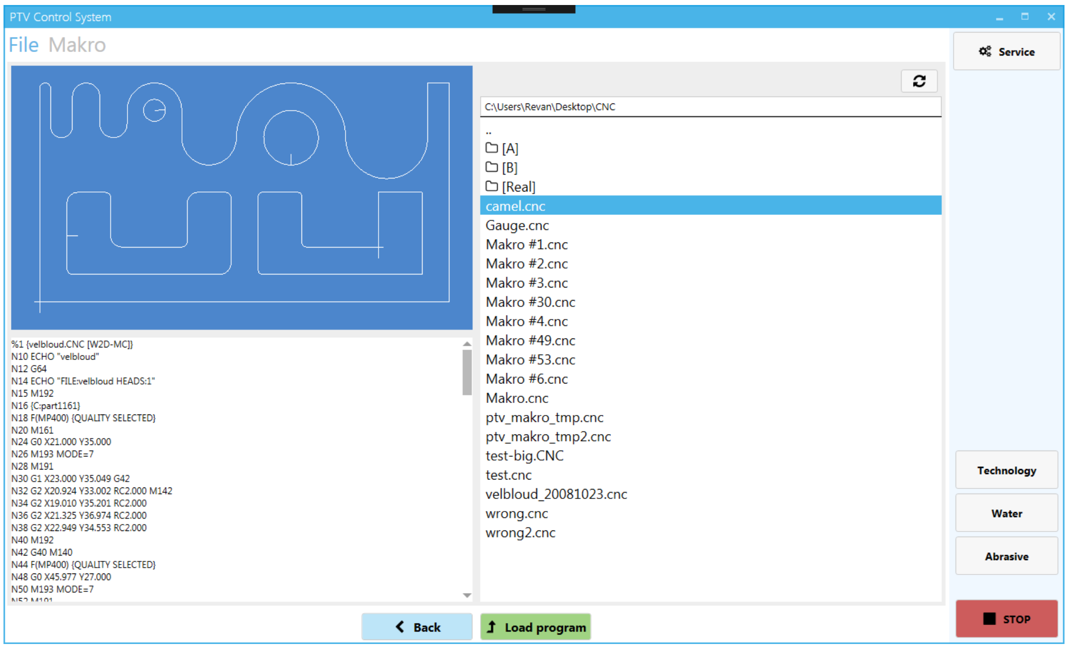


Current : camel
Total time : 00:00:00

Speed 50 mm/min

X 50
Y 50
Z 10
Speed (mm/s) 0

Figure B.1: Main screen with a CNC program

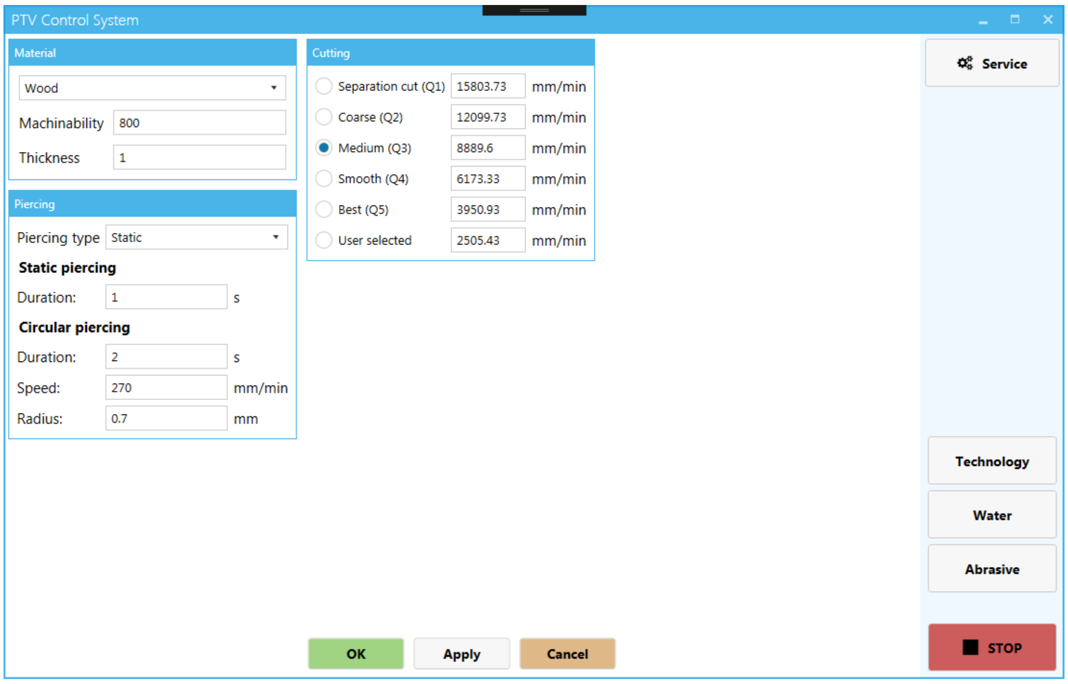


C:\Users\Revan\Desktop\CNC

- ..
- [A]
- [B]
- [Real]
- camel.cnc
- Gauge.cnc
- Makro #1.cnc
- Makro #2.cnc
- Makro #3.cnc
- Makro #30.cnc
- Makro #4.cnc
- Makro #49.cnc
- Makro #53.cnc
- Makro #6.cnc
- Makro.cnc
- ptv_makro_tmp.cnc
- ptv_makro_tmp2.cnc
- test-big.CNC
- test.cnc
- velbloud_20081023.cnc
- wrong.cnc
- wrong2.cnc

```
%1 {velbloud.CNC [W2D-MC]}
N10 ECHO "velbloud"
N12 G64
N14 ECHO "FILE:velbloud HEADS:1"
N15 M192
N16 {C:part1161}
N18 F(MP400) {QUALITY SELECTED}
N20 M161
N24 G0 X21.000 Y35.000
N26 M193 MODE=7
N28 M191
N30 G1 X23.000 Y35.049 G42
N32 G2 X20.924 Y33.002 RC2.000 M142
N34 G2 X19.010 Y35.201 RC2.000
N36 G2 X21.325 Y36.974 RC2.000
N38 G2 X22.949 Y34.553 RC2.000
N40 M192
N42 G40 M140
N44 F(MP400) {QUALITY SELECTED}
N48 G0 X45.977 Y27.000
N50 M193 MODE=7
N52 M191
```

Figure B.2: Loading of a CNC program



Material

	Wood
Machinability	800
Thickness	1

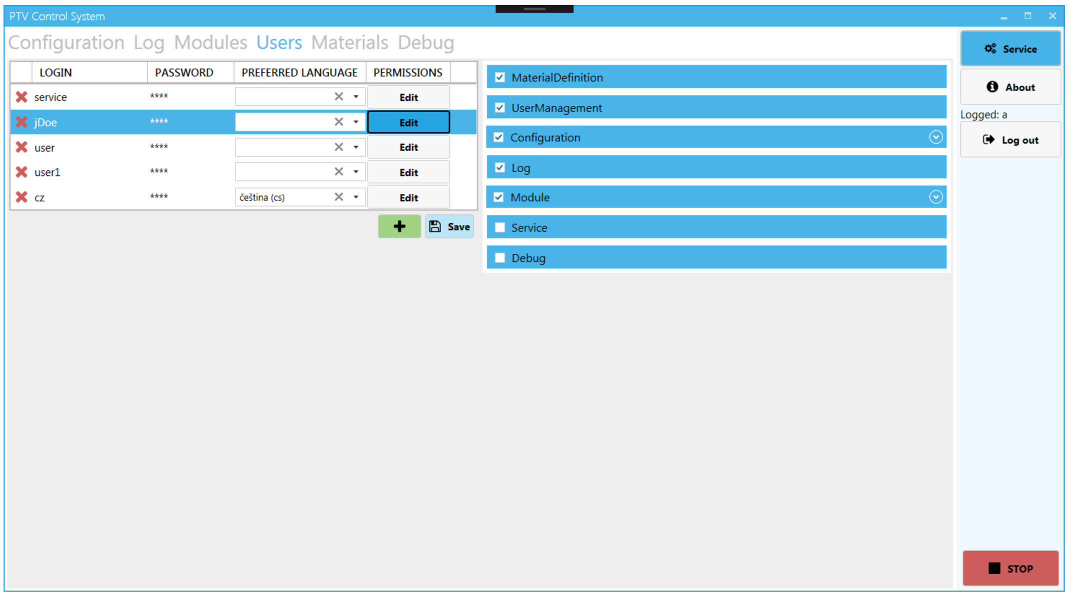
Piercing

Piercing type	Static	
Static piercing		
Duration:	1	s
Circular piercing		
Duration:	2	s
Speed:	270	mm/min
Radius:	0.7	mm

Cutting

Separation cut (Q1)	15803.73	mm/min
Coarse (Q2)	12099.73	mm/min
[✓ Medium (Q3)]	8889.6	mm/min
Smooth (Q4)	6173.33	mm/min
Best (Q5)	3950.93	mm/min
User selected	2505.43	mm/min

Figure B.3: Configuration of a CNC program



LOGIN	PASSWORD	PREFERRED LANGUAGE	PERMISSIONS
service	****		Edit
jDoe	****		Edit
user	****		Edit
user1	****		Edit
cz	****	čeština (cs)	Edit

- ✓ MaterialDefinition
- ✓ UserManagement
- ✓ Configuration
- ✓ Log
- ✓ Module
- ☐ Service
- ☐ Debug

Figure B.4: User management

Appendix C
Installation Manual

C.1 Requirements

To run the application in simulation mode:

- OS: Microsoft Windows 7, 8.1 or 10
- 2 Core CPU, 2 GHz
- 4 GB RAM
- 1 GB HDD Space
- DirectX 11 Capable graphics card
- DirectX redistributable (June 2010)
- .NET Framework 4.6.2

To run the application in production mode, it required to have additionally installed TG Motion Framework + Interval Zero. The programs are proprietary software and come together with purchase of a machine. Therefore they cannot be included with the thesis. The application will run in production mode without this software, but some commands will not have any effect because it expects feedback from TG Motion.

C.2 Installation

Copy the folder bin from the CD to hard drive of your computer. Be sure to meet all requirements listed in C.1. Now run the **PTV.ControlSystem.GUI.exe** executable. To run it in simulation mode, see C.3

C.3 Startup Arguments

Application can be started with following command line arguments:

--simulation Runs the application in simulation mode (7.1).

--lang [LANGUAGE_CULTURE_NAME] Sets the language as startup language.

--no-modules Modules won't be loaded.

--disabled-modules [MODULE1_NAME] [MODULE2_NAME] [...] Listed modules won't be loaded. Can be used if a certain module is breaking the application.

To start application with command line arguments on windows, open CMD. Go to the folder with the application and type: ExecutableName arg1 arg2 ... For example: "PTV.ControlSystem.GUI --simulation".

Appendix D

User Manual

The user manual to the PTV Control System. It will explain you the basics of operating a CNC cutting machine using the control system. The system can be used to operate a single headed, laser or water-jet, CNC cutter. The manual will not show how to configure such machine, that is a job for administration manual.

Logging in

When you start the application, you are presented with a login screen (Figure D.1). During the first start, two initial users are created – “service” and “admin”. Their passwords are equal to their user names. Service account is reserved for service technicians. It has access to very critical setting which shouldn't be manipulated by a common user. The admin is the account to go, it has all permissions except the before mentioned. Go and log in.

D.1 Main Screen

After logging in, the main screen is shown (Figure D.2). It is a central hub of the application where you will spend most of your time. Here you can manually control the cutting head, load and execute CNC programs or enter the service mode. All the mentioned functions will be explained in their respective sections.

Draw Window

Draw window is located in the middle of the main screen. It shows important things on the machine. The window can be zoomed or moved. For zooming, use either mouse scroll wheel, the icons with magnifying glass or input gestures. To move the window, hold left mouse inside the window and move the mouse or touch it with a finger and move the finger. You can also focus to see the full workspace of the machine or center to a loaded CNC program by clicking the buttons right of the magnifying glasses. By default, the window shows the view from the top. You can switch between the different views by clicking

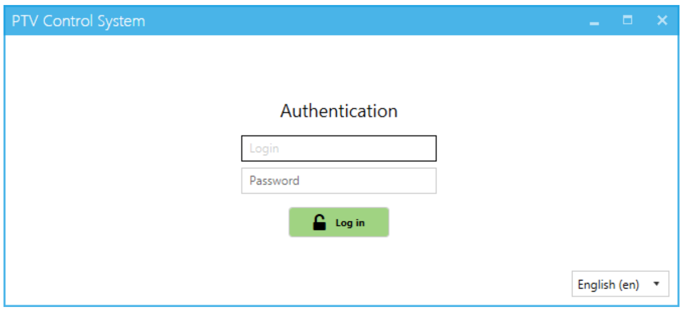


Figure D.1: Login screen. First screen shown during application startup.

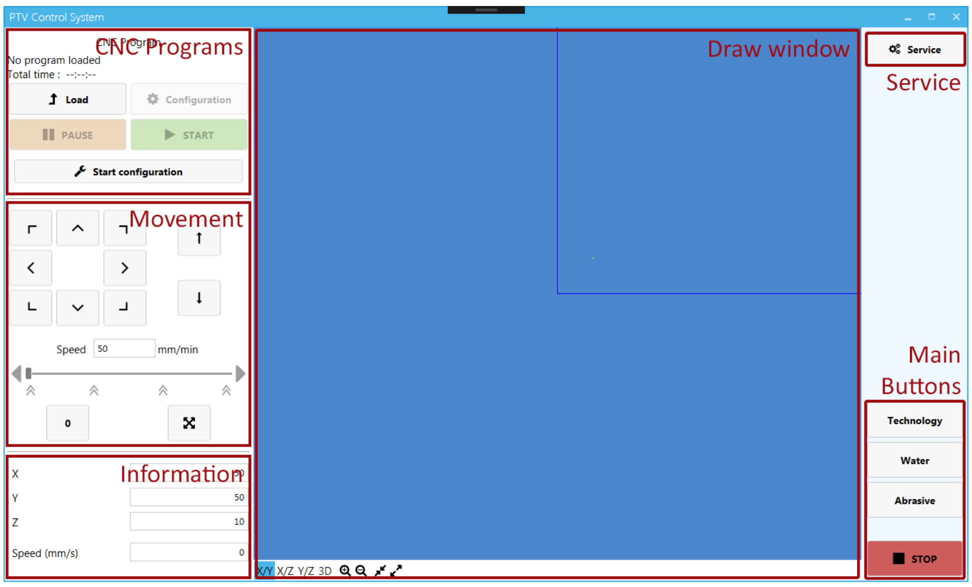


Figure D.2: Main screen

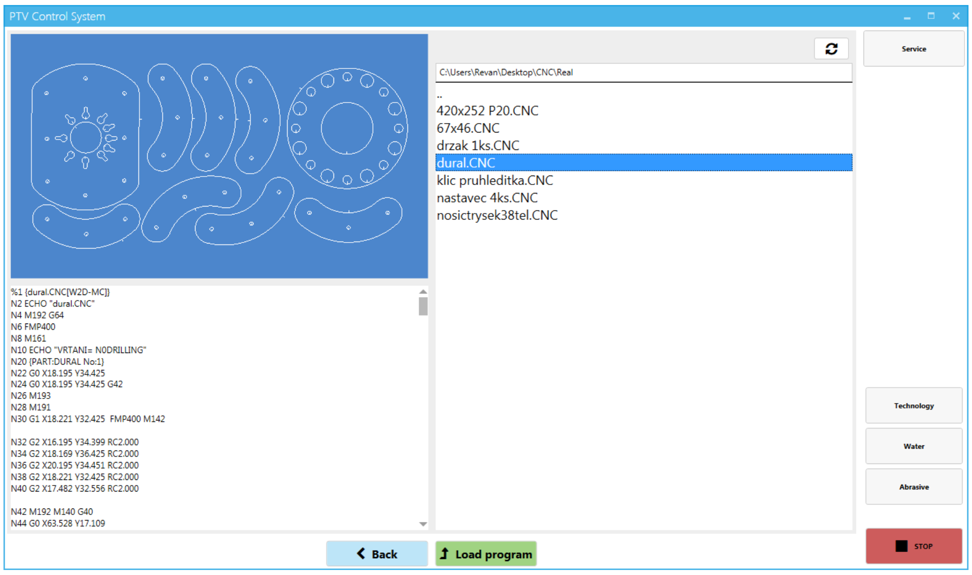
the X/Y, X/Z, Y/Z or 3D buttons. However, only the default X/Y view is fully supported.

D.2 CNC Programs

This section will show you how to load and execute CNC programs.

Loading a CNC Program

To load a cnc program, click the Load button in the upper left corner. This opens up CNC program loading screen (Figure D.3). In the right, there is a file explorer. Locate the folder containing CNC files and select the desired CNC program. A thumbnail of the program and its G-code is shown on the left side. To load it, click the **Load program** button. The program should be now visible and centered on the draw window.



C:\Users\Revan\Desktop\CNC\Real

```
..
420x252 P20.CNC
67x46.CNC
drzak 1ks.CNC
dural.CNC
klic pruhleditka.CNC
nastavec 4ks.CNC
nosictrysek38tel.CNC
```

```
%1 {dural.CNC[W2D-MC]}
N2 ECHO "dural.CNC"
N4 M192 G64
N6 FMP400
N8 M161
N10 ECHO "VRTANI= N0DRILLING"
N20 {PART:DURAL No:1}
N22 G0 X18.195 Y34.425
N24 G0 X18.195 Y34.425 G42
N26 M193
N28 M191
N30 G1 X18.221 Y32.425  FMP400 M142

N32 G2 X16.195 Y34.399 RC2.000
N34 G2 X18.169 Y36.425 RC2.000
N36 G2 X20.195 Y34.451 RC2.000
N38 G2 X18.221 Y32.425 RC2.000
N40 G2 X17.482 Y32.556 RC2.000

N42 M192 M140 G40
N44 G0 X63.528 Y17.109
```

Figure D.3: Loading a CNC program

Loading a Makro

Makro allows you to load a program from a library of predefined shapes. Before using it, be sure the module is installed and enabled (see Section D.4). Click the Load button like when loading a CNC program from a file. Now select the *Makro* tab and choose the desired shape (Figure D.4). Each shape offers configurable dimensions. After it is configured, click the **Load program** button. The program should be now visible and centered on the draw window.

Configuring a CNC Program

When the CNC program is loaded, it is possible to set its configuration. The configuration is used to adapt the program for the material into which it is cut. To open configuration, press the *Configuration* button next to the load button. In the configuration menu, you can select the target material, piercing type, and speed of the cutting. The system calculates various speeds based on properties of the material. If all of the offered speeds are unsuitable, you can also enter custom speed.

Setting Start Configuration

Before executing the CNC program, it is required to set start configuration. The start configuration describes position and rotation of the program inside the machine. To configure the start configuration, press the **Start configuration** button which shows a modal window (Figure D.5). There are multiple ways how to configure start configuration. In the **Parameters** box, you can input absolute coordinates. In the **Position** box, you can either set the

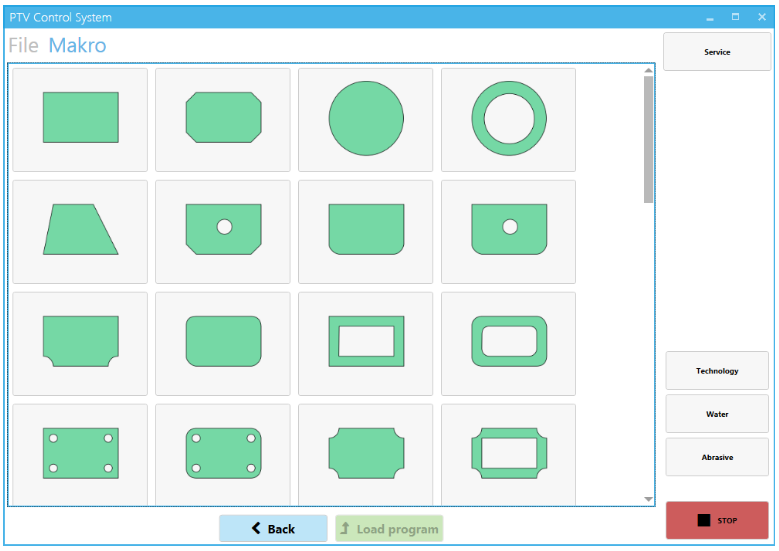


Figure D.4: Loading a makro

start configuration to the current location of the head or move it by specified amount. The **Rotation** box allows to set rotation more precisely.

Executing a CNC Program

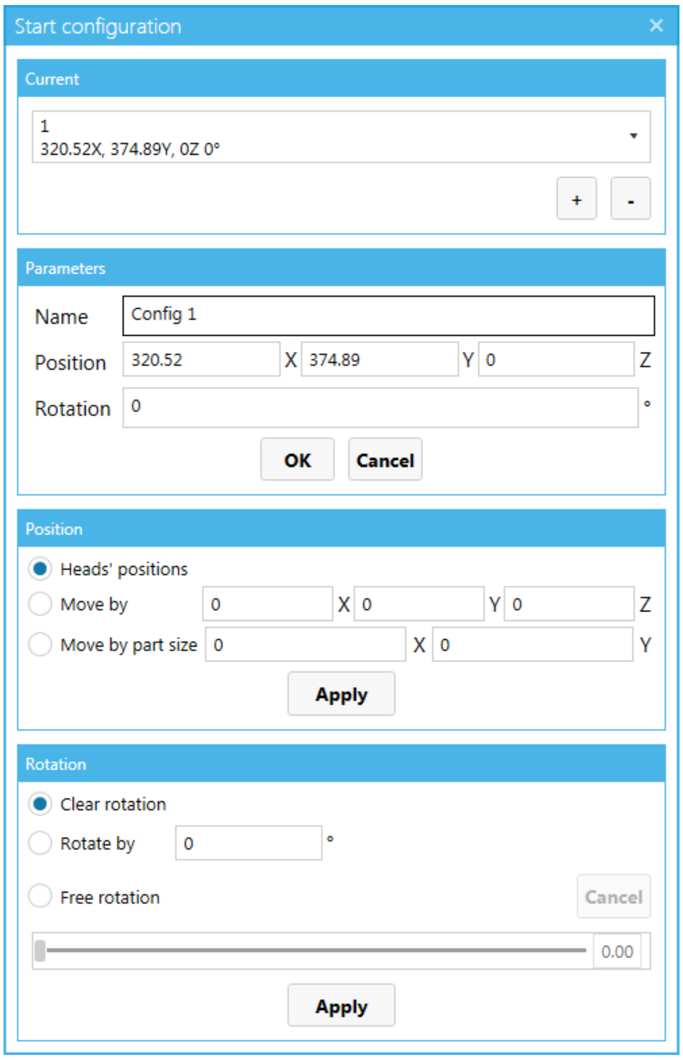
When the start configuration is set, we can see the program on the draw window in its target position. If the program is configured, you can start the cutting process. Pressing the green **Start** button will execute the program. The interface will now switch into cutting mode. Draw window displays progress of the program. We should see the cutting head moving and already cut parts of the program are drawn with different color. To stop the program, press the **Stop** button, to only pause it, press the **Pause** button. When the program is paused, you can either continue with the execution or completely stop it.

D.3 Manual Control

This section explains how to manually operate the machine.

Manual Movement

To manually move the cutting heads, use the Movement control located left from the draw window. The eight arrows represent movement in axes X/Y. The two arrows on the right move the head up and down (Z axis). Speed of the movements is defined by the value shown in the textbox below. To adjust the speed, either use the slider or double click one of the respective arrows below the slider to set the speed to a predefined value. The button with “0” moves the head to the location of start configuration (how to set it, see the


Start configuration

Current

1
320.52X, 374.89Y, 0Z 0°

Parameters

Name: Config 1

Position: 320.52 X 374.89 Y 0 Z

Rotation: 0 °

Position

(● Heads' positions)

Move by: 0 X 0 Y 0 Z

Move by part size: 0 X 0 Y

Rotation

(● Clear rotation)

Rotate by: 0 °

Free rotation

0.00

Figure D.5: Start configuration modal window

Section D.2). The button with diagonal arrows opens an absolute/relative movement menu (show in Figure D.6). This lets you to move the head either to a specific location (absolute) or by a specific amount (relative). Any movement can be canceled with the *Stop* button located in the lower right corner of the application.

Manual Cutting

Even though most of the times you will cut the CNC programs, you can also cut manually. Manual cutting is not precise but can be handy for simple actions like separating a material. The buttons for manual cutting are labeled as *Main buttons*. The *Technology* button is a master switch button for cutting. If toggled off, nothing will come out of the cutting nozzle. The *Water* button switches on/off the water-jet. The *Abrasive* switches on/off adding of abrasive into the water-jet. This button has no effect when water-jet is off. Use the

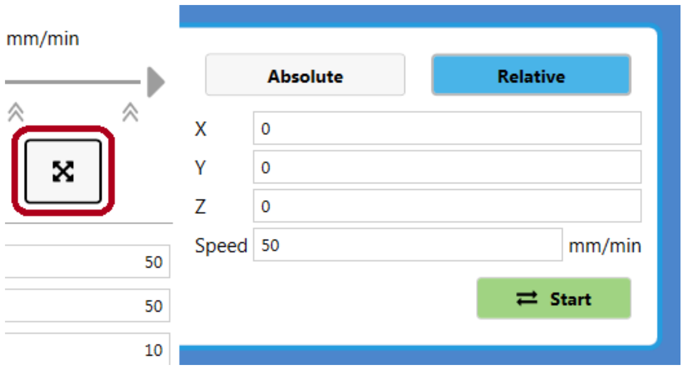


Figure D.6: Absolute/relative movement menu

manual movement buttons (see Section D.3) to move the head in desired motion during cutting.

D.4 Application Maintenance

The less frequent agendas aimed at maintenance of the application are hidden behind the *Service* button of the main screen. When selected, the application switches to a service mode. Individual agendas are explained in the following sections. You cannot switch to service if the machine is moving or executing a CNC program.

Configuration

In the configuration, we can change settings of the application. Each module can have its own configuration which is further divided into sections. In the table, all settings the user has access to are displayed. To change a setting, double click the Value column of the given setting and enter a new value. The configuration needs to be manually saved by clicking the *Save* button. Some settings require a restart to take effect.

Log

The log contains list of events which happened from start of the application. Each message can contain additional information which can be shown when clicking on the *Open* in the *Detail* column. The messages are distinguished by severity which indicates importance of given message. The log agenda can be crucial when tracing events after a collision. The logs are also stored into the *Logs* folder inside the folder where the application executable is.

Module Management

The application is extensible through installation of additional modules. To install a module, copy it to the *Modules* directory. When a new module is installed, it is disabled by default. The module agenda lets us to enable/disable available modules and also see their licenses. The first module is always *Core* which cannot be disabled and represents the core of the application. On the right side of the module name, license info about given module is displayed[1]. Application needs to be restarted when a module is enabled/disabled for it to take effect.

User Management

User management agenda lets us manage accounts of users and specify their permissions. In the left table, we can add/update/remove users. Each user can have its own preferred language. When he logs in, application switches to that language. If none is chosen, a global application language is used (can be set in configuration). To edit permissions of a user, select the user in the table and click the *Permissions* button. The individual permissions usually correspond to same named agendas. All changes performed in the user agenda need to be confirmed by clicking the *Save* button.

Materials

Materials agenda allows to manage list of known materials and their properties. They can be then used when configuring CNC programs. To edit a material, double click the desired column in the table. Changes done to materials are saved automatically. The "machinability refers to the ease with which the material can be cut"[2]. The cutting speed computed by the software for a given CNC program is highly influenced by machinability of a selected material.

[1]Non-production release does not validate licenses. In the production release, the application would either shut down or disable the module if no valid license would have been found.

[2]en.wikipedia.org/wiki/Machinability

Appendix E

CD Content

- bin/ – Application binaries
    - Logs/ – Contains application logs
    - Modules/ – Root folder for modules
    - PTV.ControlSystem.GUI.exe – main executable
- src/ – Source codes of the application
    - ControlSystem/ – Contains main sources of the control system
        - CppToCSharp/, MemoryAlocator/ – Custom tools used during development
        - PTV.ControlSystem.Core/ – Core
        - PTV.ControlSystem.Core.Test/ – Tests for Core
        - PTV.ControlSystem.LicenseService*/ – Licensing WCF client/server code
        - PTV.ControlSystem.GUI/ – GUI
        - PTV.ControlSystem.Module.Makro/ – Makro module source
        - PTV.ControlSystem.PLC/ – PLC
        - PTV.ControlSystem.Server/ – Server for hosting WCF services
        - TG.CNC*/ – TG Motion library wrapper for CNC program execution
        - TG.Communication*/ – TG Motion library wrapper for shared memory communication
        - PTV Control System.sln – Control system VS 2015 solution
    - Utilities/ – Supporting projects
        - PTV.Geometry/ – Geometric library
        - PTV.Geometry.Test/ – Tests for geometry
        - PTV.Utils/ – Generic utilities
        - PTV.WPFUtils/ – WPF utilities
        - Utilities.sln – Utilities VS 2015 solution
    - Makro/ – Sources of the standalone Makro library
        - PTV.Makro/ – Makro sources

    - MakroSvgExport/ – Custom tool for exporting PNG images from SVG files using Inkscape CLI
    - PTV Makro.sln – Makro VS 2015 solution

- thesis/
  - thesis.pdf – This thesis
  - src/ – LaTeX sources of the thesis including used template
- cnc/ – Example CNC programs
- licenses/ – Licenses of used third-party software (see F.2)

Appendix F
Others

F.1 Tools Used

List of major tools used during development.

Microsoft Visual Studio 2015 – IDE for C# and C++.

NuGet – Package manager for Visual Studio. Manages dependencies to third-party software listed in Section F.2.

Resharper – Extension to Visual Studio with lot a of helpful tools.

Git – Version control software.

Bitbucket.org – Hosting for Git repositories.

Inkscape – Vector graphics editor. Used for drawing of shapes for the Makro library.

F.2 Third-party Software Used

This is a list of third party software which is used within the control system. Names of the listed packages correspond to their names within NuGet. Dependencies of the following packages are not listed. Licenses for individual packages are stated in the brackets.

.NET Framework + System Libraries (net_library_eula_enu) – .NET Framework and standard libraries.

AvalonEdit (MIT) – Basic text editor. Used to display the G-code when loading CNC programs. Can handle big files as opposed to the default WPF text box.

CommandLineParser (MIT) – Parser for command line parameters.

Extended.Wpf.Toolkit (Ms-PL) – Collection of WPF controls and utilities.

JetBrains.Annotations (MIT) – Attributes for static code analysis. Requires ReSharper to be effective.

MahApps.Metro (MIT) – Metro style interface.

MahApps.Metro.Icon (MIT) – Collection of icons mostly used within buttons.

Microsoft.EntityFrameworkCore (net_library_eula_enu) – ORM data access technology.

Microsoft.EntityFrameworkCore.Sqlite (net_library_eula_enu) – Database provider for the Entity Framework Core.

NLog (BSD License 2.0) – Logging platform.

Prism (Apache 2.0) – Extensibility support.

QuickConverter (MIT) – Scriptable WPF converter. Avoids writing a specialized converter for every scenario.

Resource.Embedder (MIT) – Embeds satellite assemblies to main assembly.

SharpDX (SharpDX) – DirectX wrapper for C#.

SharpDX.WPF (FreeBSD License) – SharpDX integration into WPF.

SpicyTaco.AutoGrid (MIT) – Provides the AutoGrid and StackPanel WPF panels. They are more refined than their default counterparts.

UnitsNet (MIT) – Eases manipulation with physical units.

WPFTaskDialog (CPOL) – Modal windows for WPF emulating native Windows task dialogs.

F.3 Code Statistics

Code statistics including every project used within the application. The metrics were calculated by the tool **cloc v1.72**[1].

Language	files	blank	comment	code
C#	562	4962	4172	24518
MSBuild script	21	0	147	3595
XAML	59	339	36	2071
C/C++ Header	23	282	169	1080
XML	2	7	0	857
C++	25	165	59	538
SUM:	692	5755	4583	32659

[1]github.com/AlDanial/cloc