CS LAB s.c.
ElectronicLaboratory

user * guide * manual *

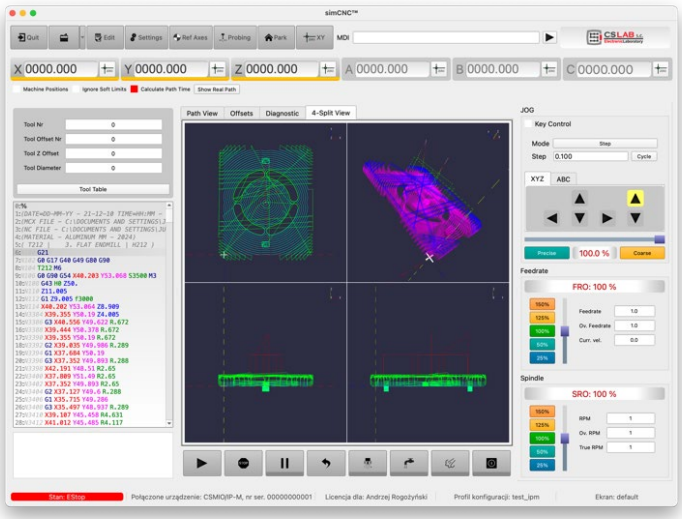
simCNC
Steuerungssoftware

Editor für die Bedienerschnittstelle

Inhaltsverzeichnis

1. Allgemeines

Die simCNC-Software verfügt über einen modernen grafischen Editor, mit dem Sie Ihre eigenen, genau auf die Kundenbedürfnisse zugeschnittenen, originellen Bedienerschnittstellen erstellen können.
In Verbindung mit der Skriptsprache Python und der Gestaltung unter Einsatz von gängigem CSS ermöglicht der Editor die Erstellung funktionaler und visuell ansprechender Schnittstellen.
Der Programmcode wurde im Hinblick auf Effizienz optimiert, um Responsivität und Benutzerfreundlichkeit zu gewährleisten. Es wurde auch ein System zur Auto-Positionierung und Skalierung der grafischen Elemente verwendet, wodurch die erstellte Schnittstelle dynamischer wird und sich an ein breites Spektrum von Bildschirmgrößen und -auflösungen anpassen kann. Da die simCNC-Software plattformübergreifend ist, können Bildschirmentwürfe ohne Änderungen unter Windows, macOS und Linux verwendet werden.



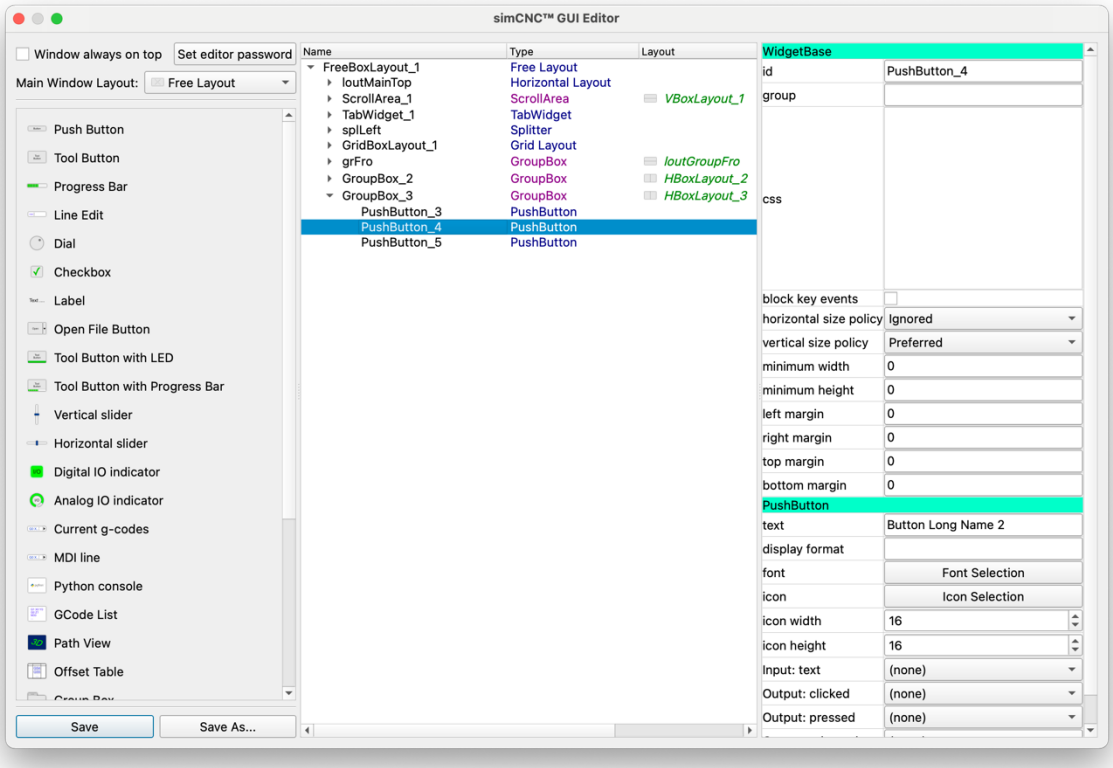
1.1. Empfehlungen und Systemanforderungen

Die simCNC-Software und der integrierte Grafikeditor stellen keine hohen Anforderungen an die Hardware.
Das Programm funktioniert sogar auf einem RaspberryPI4 mit 4 GB RAM. Um jedoch komfortabel arbeiten zu können, insbesondere wenn Sie eine Schnittstelle „von Grund auf“ erstellen, sollten Sie sich einen guten Monitor mit einer höheren Auflösung beschaffen, z.B. 27'' 2560x1440.
Es ist sehr bequem, zwei Monitore und einen Computer mit mehr Arbeitsspeicher zu besitzen, um Tools wie **Affinity Designer** oder **Photoshop** für die Vorbereitung grafischer Elemente oder **Visual Studio Code** für die Bearbeitung von Stylesheets (CSS) und Python-Skripten gleichzeitig nutzen zu können.

2. Allgemeine Regeln für die Zusammenarbeit mit dem Editor

2.1. Editorfenster

Das Editorfenster wird über das Menü „Konfiguration→Schnittstelleneditor öffnen“ geöffnet.



Das Editorfenster ist in drei vertikale Hauptbedienfelden unterteilt.
In dem linken Bedienfeld (von oben) befinden sich die folgenden Elemente:

- **Window always on top** – wenn diese Option markiert ist, befindet sich das Editorfenster immer im Vordergrund.
- **Set editor password** – Passwortschutz für den Bearbeitungsbildschirm
- **Main Window Layout** – Auswahl, wie der Hauptfenster-Container positioniert werden soll (siehe Beschreibung des Autopositionierungssystems weiter unten in dieser Anleitung)
- **Liste der Widgets** - um ein bestimmtes Widget im Projekt zu platzieren, klicken Sie mit der Maus und ziehen Sie es auf den entworfenen simCNC-Bildschirm
- **Save** – Änderungen speichern
- **Save As** – das Projekt unter einem anderen Namen speichern

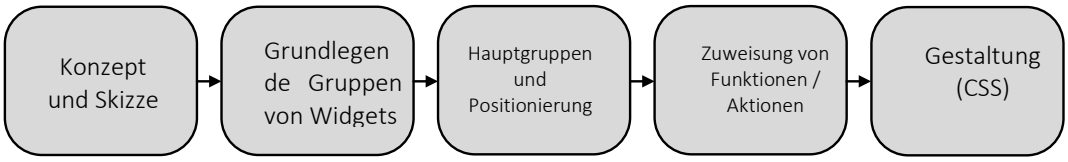
Das mittlere Bedienfeld zeigt einen Baum aller Elemente der entworfenen Schnittstelle, während das rechte Bedienfeld eine Liste der Eigenschaften des ausgewählten Widgets oder Containers anzeigt.
Die Eigenschaften sind im Abschnitt über Widgets beschrieben.

2.2. Tastenkürzel

Kürzel	Beschreibung
CTRL-C	Ausgewählte Elemente in die Zwischenablage kopieren
CTRL-V	Elemente aus der Zwischenablage in den ausgewählten Container einfügen
CTRL-Z	Die letzte Operation rückgängig machen (undo)
CTRL-Y	Die rückgängig gemachte Operation wiederholen (redo)
CTRL-S	Projekt speichern

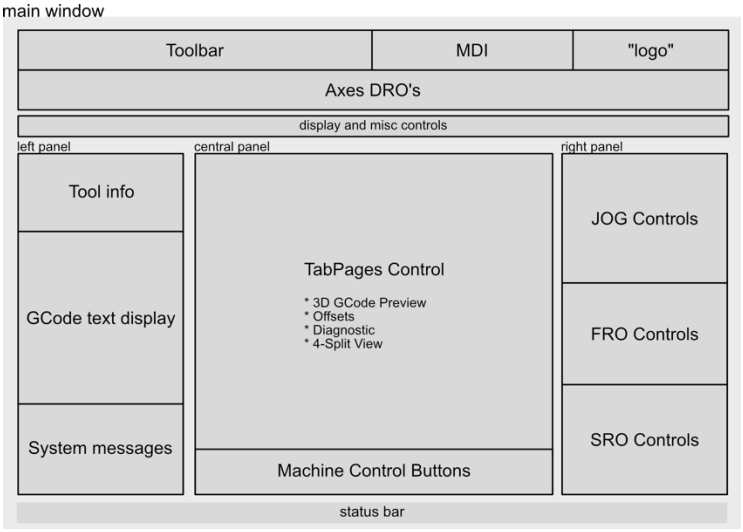
2.3. Phasen der Gestaltung einer neuen Schnittstelle (Workflow)

In diesem Kapitel wird ein allgemeiner Arbeitsablauf für die Gestaltung einer neuen Schnittstelle präsentiert. Machen Sie sich keine Sorgen, wenn einige der hier verwendeten Begriffe unklar sind. Eine detaillierte Beschreibung der einzelnen Werkzeuge finden Sie in den folgenden Kapiteln. Wir werden uns hier auf die allgemeinen Regeln konzentrieren.



2.3.1. Konzeptskizze

Um die Möglichkeiten des Auto-Positionierungssystems voll auszuschöpfen und sich nicht in der wachsenden Anzahl von Elementen zu verlieren, ist es gut, mit der Erstellung einer Skizze zu beginnen - einem allgemeinen Konzept, wie die Elemente unserer Schnittstelle platziert werden sollen.



Die Abbildung oben zeigt ein Beispiel für eine Konzeptskizze der simCNC-Standardbildschirm (**default**).

Oberer Teil:

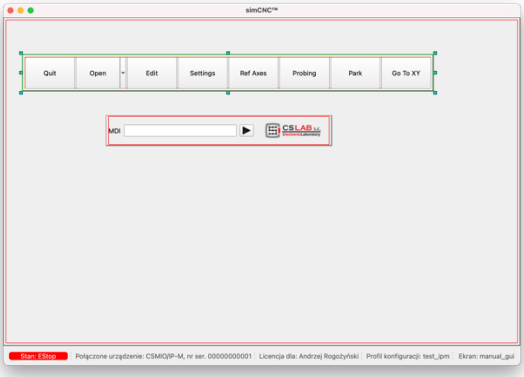
- **Toolbar** – Schaltflächenleiste
- **MDI** – Schnelleingabefeld für Maschinenbefehle
- **„logo“** – Firmenlogo
- **Axes DRO’s** – Anzeige der aktuellen Position der einzelnen Achsen der Werkzeugmaschine
- **Display and misc controls** – Steuerelemente für die Anzeige von Koordinaten und anderen Daten

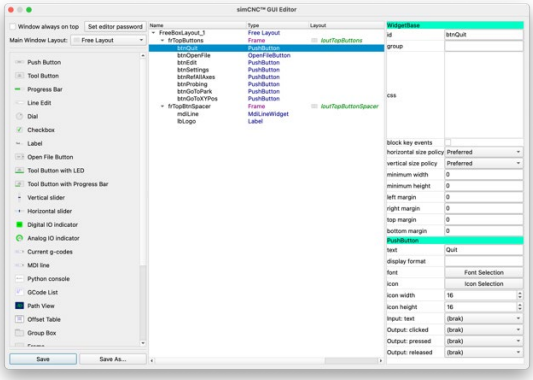
Unterer Teil, unterteilt in drei Abschnitte:

- Linker Abschnitt
  - **Tool Info** – Informationen über das aktuelle Werkzeug und Offset
  - **Gcode text display** – Anzeige des Inhalts der geladenen Gcode-Datei
  - **System messages** – Konsole für Systemmeldungen
- Zentraler Bereich
  - **TabPages Control** – Bedienfeld der Registerkarten
    - 3D-Ansicht der Gcode-Datei
    - Arbeitsoffsets
    - Diagnostik
    - 3D-Ansicht der Gcode-Datei in vier Projektionen
  - Leiste der Arbeitssteuerungsschaltflächen
- Rechter Abschnitt
  - **JOG Controls** – Bedienfeld für die manuelle Steuerung von Achsen
  - **FRO Controls** – Bedienfeld zur Regulierung der Vorschubgeschwindigkeit

- **SRO Controls** – Bedienfeld zur Regulierung der Drehzahl der Spindel

2.3.2. Vorbereiten grundlegender Gruppen von Widgets

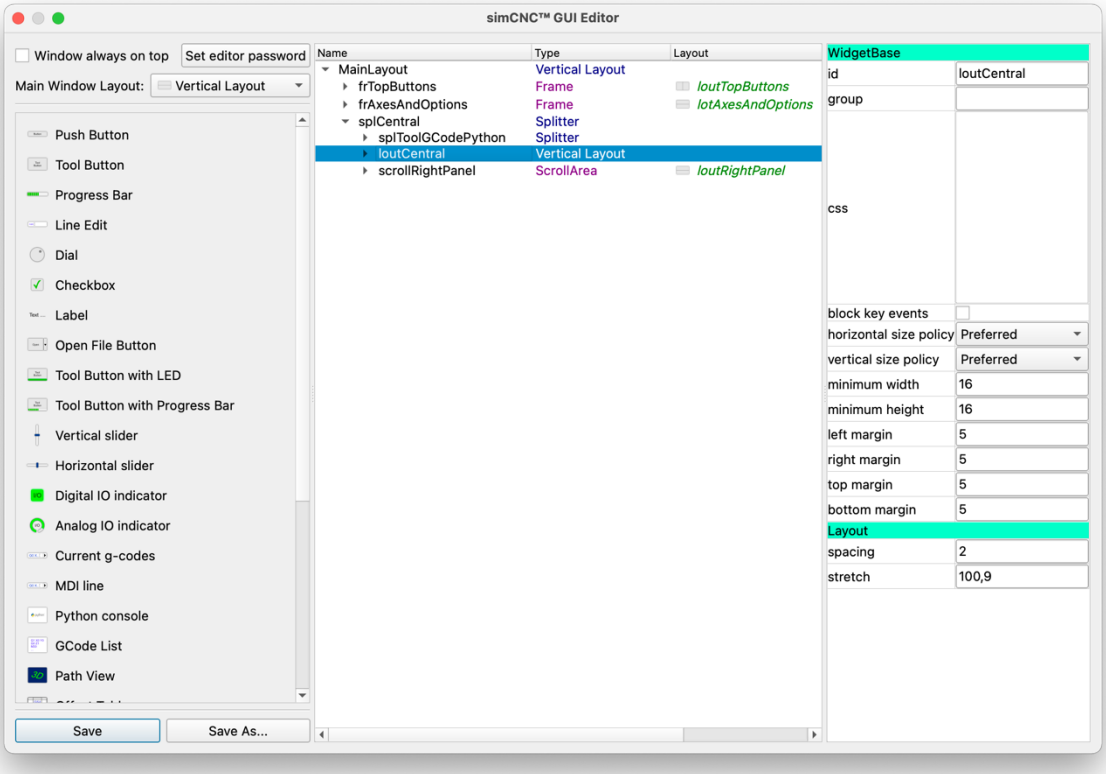




In dieser Phase können wir mit der Erstellung grundlegender Widget-Gruppen der Schnittstelle beginnen. In den Abbildungen oben sehen wir links das simCNC-Hauptfenster mit der Schaltflächengruppe und MDI, rechts das Editorfenster.
Am Anfang lohnt es sich nicht, sich mit all den Details zu befassen, z.B. mit den Symbolen für die Schaltflächen, Gestaltung usw. Dafür kommt Zeit. Stattdessen ist es ratsam, dafür zu sorgen, dass einzelne Steuerelemente in Gruppen platziert werden und für sie Positionierungsregeln zu definieren. Das Auto-Positionierungssystem wird in einem separaten Kapitel ausführlich beschrieben.
Es ist wichtig, den Widgets Namen zu geben, die es ermöglichen, sich leicht im Projekt zu orientieren.

2.3.3. Vorbereitung und Positionierung der Widget-Hauptgruppen

Der Entwurf der simCNC-Schnittstelle weist eine hierarchische Struktur auf. Grundlegende Gruppen können zu größeren Gruppen zusammengefasst werden, und schließlich kann die Positionierung des Hauptfensters des Programms definiert werden. Man kann sich einen Überblick über die Hierarchie der Projektelemente verschaffen, indem man sich das Mittelbedienfeld des Editorfensters - den Objektbaum - ansieht. Im folgenden Beispiel sieht man, dass der Hauptcontainer **MainLayout** drei Elemente enthält: **frTopButtons**, **frAxesAndOptions** und **splCentral**. **splCentral** enthält wiederum: **splToolGCodePython**, **loutCentral** und **scrollRightPanel**.

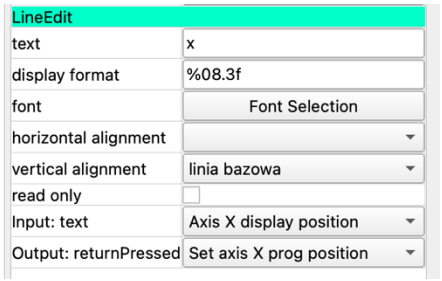


Der simCNC-Standardbildschirm hat drei Hauptgruppen, die vertikal positioniert sind:

- **frTopButtons** - Rahmen mit Schaltflächen, MDI und Logo
- **frAxesAndOptions** - Rahmen mit Achsenpositionen und Optionen
- **splCentral** - Gruppe mit dem linken, mittleren und rechten Bedienfeld

Es ist ratsam, sich in der Skizzenphase einige Gedanken über die Hierarchie und die Schnittstellengruppen zu machen. Dies erleichtert die spätere Änderung und Festlegung von Skalierungsregeln.

2.3.4. Zuweisung von Funktionen zu Widgets



In dieser Phase weisen wir den Widgets Eingabe- und Ausgabefunktionen zu. Zum Beispiel definieren wir für das Achsenpositionsanzeige-Widget eine Eingabefunktion „**Axis ... display position**“ und eine Ausgabefunktion „**Set axis ... prog position**“.
Die Eingabeaktion aktualisiert den Anzeigewert und die Ausgabeaktion ruft die Aktion „Position setzen“ auf, wenn der Inhalt des Widgets vom Bediener bearbeitet wird. Alle Eingabe- und Ausgabeaktionen sind in einem separaten Kapitel beschrieben. Für Schaltflächen kann auch **Run Skript** als Ausgabeaktion gesetzt werden und es kann ein Python-Makro erstellt werden, das ausgeführt wird, wenn der Bediener auf die Schaltfläche klickt. Damit können komplexere Aufgaben oder Aktionen durchgeführt werden, die nicht in der Standardliste enthalten sind.

2.3.5. Gestaltung kaskadierender Stylesheets (CSS)

Das Schnittstellensystem des simCNC-Programms verfügt über eine integrierte Unterstützung für **CSS**-Stylesheets. Die Gestaltung ist optional, wird aber empfohlen, wenn man eine dynamische und visuell ansprechende Schnittstelle schaffen möchte. Die Befehle können direkt im Bildschirm-Editor in das **css**-Feld eingegeben werden oder (empfohlen) es kann eine separate Datei mit der Erweiterung **.css** erstellt und in dem Verzeichnis des Bildschirms platziert werden. Eine sehr praktische Funktion ist die Möglichkeit, Gruppen von Widgets für die Gestaltung zu definieren. Wir geben dazu im Bearbeitungsmodus den Gruppennamen in das **group**-Feld der Eigenschaften des Widgets ein und können dann die visuellen Attribute in der **css**-Datei für alle Widgets modifizieren, die diesen Gruppennamen haben. Im Standardbildschirm haben zum Beispiel alle Schaltflächen in der oberen Leiste den Gruppennamen **ctrlButtons**.

WidgetBase	
id	btnSettings
group	ctrlButtons

Und so sieht das Fragment der Datei **style.css** aus, welches das Erscheinungsbild der gesamten Gruppe definiert:

```
[group="ctrlButtons"]{
  color: #404040;
  background-color: rgb(170, 170, 170);
  font-size: 12px;
}
[group="ctrlButtons"]:hover {
  color: rgb(112, 14, 14);
  background-color: lightgray;
  font-size: 13px;
}
[group="ctrlButtons"][darkTheme="true"]{
  color: #FFFFFF;
  background-color: rgb(60, 60, 60);
  font-size: 12px;
}
[group="ctrlButtons"][darkTheme="true"]:hover {
  color: rgb(182, 14, 14);
  background-color: lightgray;
  font-size: 13px;
}
```

Abgekürzte Schreibweise des Widgets „id“ bei der Eingabe von „css“ im Editorfenster

Wenn wir den **css**-Inhalt in das Editorfenster eingeben, können wir die verkürzte Widget-ID-Notation verwenden. Dazu geben wir **#id** ein, wie unten dargestellt.

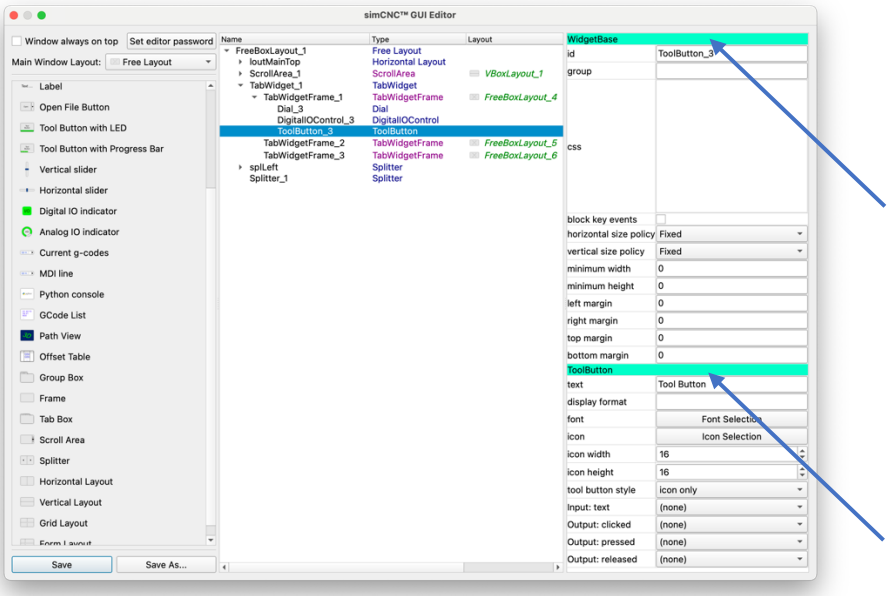
WidgetBase	
id	btnEdit
group	ctrlButtons

css	#id {
	color: yellow;
	background-color: red;
	}

Die normale, vollständige Notation würde lauten:
[id=”btnEdit”] {
...
}
Natürlich ist auch die Standard-Notation zulässig.

3. Widgets

Wenn man ein Widget im Bearbeitungsmodus auswählt, wird im rechten Bedienfeld des Editorfensters eine Liste seiner Eigenschaften angezeigt.

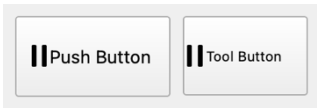


Im obigen Beispiel des **Tool Button**-Widgets sieht man, dass die Eigenschaften gruppiert sind. **WidgetBase** ist eine Gruppe von Eigenschaften, die allen Widgets gemeinsam sind. **ToolButton** ist eine Gruppe von Eigenschaften eines konkreten **Tool Button**-Widgets.
In der folgenden Tabelle sind Eigenschaften beschrieben, die allen Widgets gemeinsam sind.

Bezeichnung der Eigenschaften	Beschreibung
Id	Widget-ID. Es sollten intuitive Identifizierer (ID) vergeben werden, insbesondere wenn später CSS-Gestaltungsfunktionen verwendet werden oder wenn geplant wird, aus Python-Makros heraus auf Widgets zu verweisen. Zum Beispiel: **btnStartGCode** ist ein viel besserer Name als **buton_123**.
Group	Identifizierer der Widgetgruppe. Sehr nützlich für die Gestaltung (css). Wenn Sie mehreren Widgets denselben Gruppennamen geben, können Sie die Code-Menge im css-Stylesheet erheblich reduzieren.
css	Ein Feld für die schnelle Eingabe von Befehlen zur Gestaltung eines Widgets. Besonders nützlich beim Experimentieren mit verschiedenen Eigenschaften. Letztendlich ist es besser, eine (oder mehrere) **.css**-Datei(en) im Verzeichnis des erstellten Bildschirms zu bilden.
block key events	Wenn Sie diese Option markieren, leitet das Widget keine Informationen über die auf der Tastatur gedrückten Schaltflächen weiter. Nützlich z.B. bei Eingabefeldern wie MDI. Dadurch wird sichergestellt, dass eine Änderung der Cursorposition im Bearbeitungsfeld nicht zu einer Bewegung der Maschine führt, wenn die JOG-Kontrolle von der Tastatur aus aktiviert ist.
Horizontal size policy	Richtlinien für die horizontale Skalierung des Widgets. Eine ausführliche Beschreibung finden Sie im Abschnitt über die Auto-Positionierung.
Vertical size policy	Richtlinien für die vertikale Skalierung des Widgets. Eine ausführliche Beschreibung finden Sie im Abschnitt über die Auto-Positionierung.
Minimum width	Minimal zulässige Breite des Widgets.
Minimum height	Minimal zulässige Höhe des Widgets.
Left margin	Linker Rand des Widgets.
Right margin	Rechter Rand des Widgets.
Top margin	Oberer Rand des Widgets.
Bottom margin	Unterer Rand des Widgets.

3.1. Push Button und Tool Button

Diese beiden Schaltflächen-Widgets haben nahezu identische Funktionalität. Sie unterscheiden sich ein bisschen visuell. ToolButton hat zusätzlich die Möglichkeit, die Position des angezeigten Symbols zu wählen. Funktionell sind die beiden Widgets identisch.

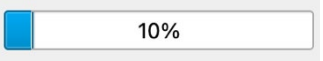


Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Text	Text, der auf der Schaltfläche angezeigt wird
Display format	Zeichenkette, die das Format der angezeigten Werte definiert. Kompatibel mit dem „printf"-Standard: https://en.wikipedia.org/wiki/Printf_format_string Z.B. **„%.3f"** zeigt einen Fließkommawert mit drei Dezimalstellen an.
Font	Einstellen der Schriftart für das Widget
Icon	Auswahl des Bildes, das als Symbol für die Schaltfläche verwendet wird
Icon width	Breite des Symbols
Icon height	Höhe des Symbols
Tool buton style	Stil der ToolButton-Schaltfläche. • **Icon only** – nur Symbol • **Text only** – nur Text • **Text beside icon** – Text neben dem Symbol • **Text under icon** – Text unter dem Symbol • **Follow style** – wie in dem Stylesheet definiert
Input: text	Widget-Eingang - Auswahl eines Parameters, der den auf der Schaltfläche angezeigten Text aktualisieren wird
Output: clicked	Widget-Ausgang - Auswahl einer Aktion, die beim Drücken der Schaltfläche aufgerufen wird
Output: pressed	Widget-Ausgang - Auswahl einer Aktion, die beim Drücken der linken Maustaste auf der Schaltfläche aufgerufen wird
Output: released	Widget-Ausgang - Auswahl einer Aktion, die beim Loslassen der linken Maustaste auf der Schaltfläche aufgerufen wird

3.2. Progress Bar

Fortschrittsbalken. Kann verwendet werden, um verschiedene Werte wie FRO, SRO usw. grafisch darzustellen.



Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Text	Text, der auf dem Widget angezeigt wird
Display format	Zeichenkette, die das Format der angezeigten Werte definiert. Kompatibel mit dem „printf"-Standard: https://en.wikipedia.org/wiki/Printf_format_string Z.B. **„%.3f"** zeigt einen Fließkommawert mit drei Dezimalstellen an.
Font	Einstellen der Schriftart für das Widget
Horizontal alignment	Horizontale Position des Inhalts (Text) des Widgets • **Left** – linksbündig • **Right** – rechtsbündig • **Center** – mittig • **Justify** – Ausrichten
Vertical alignment	Vertikale Position des Inhalts (Text) des Widgets • **Top** – obenbündig • **Top** – untenbündig • **Center** – mittig • **Baseline** – an Basislinie ausrichten
Read only	Wenn diese Option markiert wird, ist der Inhalt des Widgets schreibgeschützt und kann nicht bearbeitet werden.
Input: value	Widget-Eingang - Auswahl eines Parameters, der den Wert des Fortschrittsbalkens aktualisieren wird

3.3. Line Edit

Feld zur Anzeige und Bearbeitung von Text.

Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Text	Angezeigter Text - Inhalt des Widgets
Display format	Zeichenkette, die das Format der angezeigten Werte definiert. Kompatibel mit dem „printf"-Standard: https://en.wikipedia.org/wiki/Printf_format_string Z.B. **„%.3f**" zeigt einen Fließkommawert mit drei Dezimalstellen an.
Font	Einstellen der Schriftart für das Widget
Horizontal alignment	Horizontale Position des Inhalts (Text) des Widgets • **Left** – linksbündig • **Right** – rechtsbündig • **Center** – mittig • **Justify** – Ausrichten
Vertical alignment	Vertikale Position des Inhalts (Text) des Widgets. • **Top** – obenbündig • **Top** – untenbündig • **Center** – mittig • **Baseline** – an Basislinie ausrichten
Read only	Wenn diese Option markiert wird, ist der Inhalt des Widgets schreibgeschützt und kann nicht bearbeitet werden.
Input: text	Widget-Eingang - Auswahl eines Parameters, der den im Bearbeitungsfeld angezeigten Text aktualisieren wird
Output: returnPressed	Widget-Ausgang - Auswahl der Aktion, die beim Drücken der Return-Schaltfläche am Ende der Bearbeitung des Widget-Inhalts aufgerufen wird

3.4. Dial

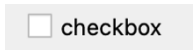
Drehknopf - zum Einstellen numerischer Werte innerhalb eines bestimmten Bereichs.

Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Wrapping	Einstellen des Drehbereiches auf volle 360° ohne Totzone.
Notches visible	Aktiviert die Anzeige der Teilung
Notch target	Teilungshub
Value	Aktueller Wert
Minimum	Minimaler Wert
Maximum	Maximaler Wert
Single step	Normaler Verstellschritt (Pfeil nach oben oder unten drücken)
Page step	Schnellverstellschritt (page up oder page down drücken)
Inverted appearance	Umgekehrte Anzeige des Drehknopfes. Wechsel von Minimum gegen Maximum.
Inverted controls	Umkehrung der Verstellrichtung
Input: value	Widget-Eingang - Auswahl eines Parameters, der die Position des Drehknopfes aktualisieren wird
Output: valueChanged	Widget-Ausgang - Auswahl einer Aktion, die bei der Änderung der Position des Drehknopfes durch den Benutzer aufgerufen wird

3.5. Checkbox

Auswahlschaltfläche. Dient zum Aktivieren/Deaktivieren von Optionen, z. B. zum Ein-/Ausschalten von Programmgrenzen.



Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Text	Anzeigetext
Checkbox state	Setzt den Markierungsstatus
Display format	Zeichenkette, die das Format der angezeigten Werte definiert. Kompatibel mit dem „printf"-Standard: https://en.wikipedia.org/wiki/Printf_format_string Z.B. **„%.3f"** zeigt einen Fließkommawert mit drei Dezimalstellen an.
Font	Einstellen der Schriftart für das Widget
Input: checkbox state	Widget-Eingang - Auswahl eines Parameters, der den Widget-Status aktualisieren wird
Input: text	Widget-Eingang - Auswahl eines Parameters, der den Widget-Text aktualisieren wird
Output: stateChanged	Widget-Ausgang - Auswahl einer Aktion, die bei der Änderung des Widget-Status aufgerufen wird

3.6. Label

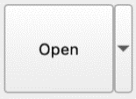
Widget-**Label** (Etikett) dient zum Anzeigen von Text oder Grafik. Ruft keine Aktion auf.

Eigenschaften:

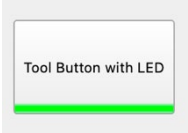
Bezeichnung der Eigenschaften	Beschreibung
Text	Anzeigetext
Display format	Zeichenkette, die das Format der angezeigten Werte definiert. Kompatibel mit dem „printf"-Standard: https://en.wikipedia.org/wiki/Printf_format_string Z.B. **„%.3f"** zeigt einen Fließkommawert mit drei Dezimalstellen an.
Font	Einstellen der Schriftart für das Widget
Word wrap	Aktiviert Worttrennung
Horizontal alignment	Horizontale Position des Inhalts (Text) des Widgets • **Left** – linksbündig • **Right** – rechtsbündig • **Center** – mittig • **Justify** – Ausrichten
Vertical alignment	Vertikale Position des Inhalts (Text) des Widgets • **Top** – obenbündig • **Top** – untenbündig • **Center** – mittig • **Baseline** – an Basislinie ausrichten
Pixmap	Auswahl des anzuzeigenden Bitmaps
Scale mode	Auswahl der Art der Bitmap-Skalierung • **Normal** - ohne Erhaltung der Proportion • **KeepAspect** – mit Erhaltung der Proportion
Input: text	Widget-Eingang - Auswahl eines Parameters, der den Widget-Text aktualisieren wird

3.7. Open File Button

Spezielle Schaltfläche zum Laden von GCode-Dateien. Ähnlich wie **Tool Button**, zeigt aber zusätzlich eine Liste der zuletzt geöffneten Dateien an. Es müssen keine Ein- und Ausgabeaktionen definiert werden, um zu funktionieren. Die Eigenschaften sind identisch wie bei **Tool Button**.



3.8. Tool Button with LED

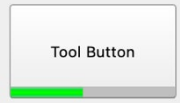


Variante der Schaltfläche Typ **Tool Button**. Zusätzlich ermöglicht es die Anzeige einer Statusleuchte, für die separat eine Aktion definiert werden kann und die den **LED**-Status aktualisieren wird.

Unterstützt alle Eigenschaften von Tool Button sowie die unten aufgeführten:

Bezeichnung der Eigenschaften	Beschreibung
LED visible	Ermöglicht es, die Anzeige der LED-Leuchte auf der Schaltfläche ein-/auszuschalten
LED state	LED-Zustand - ein/aus
LED interval	Wenn der Wert größer als Null ist, blinkt die LED. Der Wert wird in Millisekunden angegeben und definiert die Hälfte der Periode. Die Eingabe von 500 in dieses Feld bedeutet zum Beispiel *T = 500x2 = 1s ; f = 1/T = 1Hz*
Color	Auswahl der Farbe der LED-Leuchte
Input: LED state	Widget-Eingang - Auswahl eines Parameters, der den auf der Schaltfläche der LED-Leuchte angezeigten Status aktualisieren wird
Input: LED interval	Widget-Eingang - Auswahl eines Parameters, der das Blinkintervall der angezeigten LED-Leuchte aktualisieren wird

3.9. Tool Button with Progress Bar

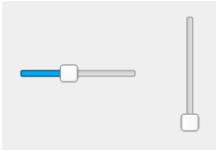


Variante der Schaltfläche Typ **Tool Button**. Zusätzlich ermöglicht es die Anzeige des Fortschrittsbalkens, für den separat eine Aktion definiert werden kann und die den Wert aktualisieren wird.

Unterstützt alle Eigenschaften von Tool Button sowie die unten aufgeführten:

Bezeichnung der Eigenschaften	Beschreibung
Value	Aktueller Wert
Minimum	Minimaler Wert
Maximum	Maximaler Wert
Color	Auswahl der Farbe der Leiste
Input: value	Widget-Eingang - Auswahl eines Parameters, der den Wert des Fortschrittsbalkens aktualisieren wird

3.10. Horizontal Slider und Vertical Slider



Zwei Varianten - horizontaler und vertikaler Schieberegler zur Einstellung des Wertes in einem bestimmten Bereich.

Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Value	Aktueller Wert
Minimum	Minimaler Wert
Maximum	Maximaler Wert
Single step	Normaler Verstellschritt (Pfeil nach oben oder unten drücken)
Page step	Schnellverstellschritt (page up oder page down drücken)
Inverted appearance	Umgekehrte Anzeige des Schiebers. Wechsel von Minimum gegen Maximum.
Inverted controls	Umkehrung der Verstellrichtung
Input: value	Widget-Eingang - Auswahl eines Parameters, der die Position des Schiebers aktualisieren wird
Output: valueChanged	Widget-Ausgang - Auswahl einer Aktion, die bei der Änderung der Position des Schiebers durch den Benutzer aufgerufen wird

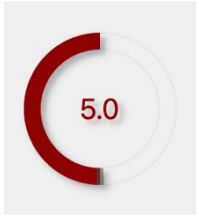
3.11. Digital IO indicator

Leuchte zur Anzeige des Status der logischen Werte (0, 1). Wird z.B. verwendet, um den Status von Hardware-Eingängen/Ausgängen zu signalisieren. Es kann auch eine Aktion auslösen, wenn es angeklickt wird.

Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Text	Anzeigetext
State	Setzt den Status der Leuchte
Display format	Zeichenkette, die das Format der angezeigten Werte definiert. Kompatibel mit dem „printf“-Standard: https://en.wikipedia.org/wiki/Printf_format_string Z.B. „%.3f“ zeigt einen Fließkommawert mit drei Dezimalstellen an.
Clickable	Zustimmung für die Steuerung des Status der Leuchte durch Mausklick
Color	Auswahl der Farbe der Leuchte
Input: text	Widget-Eingang - Auswahl eines Parameters, der den Widget-Text aktualisieren wird
Input: state	Widget-Eingang - Auswahl eines Parameters, der den Widget-Status aktualisieren wird
Output: stateChanged	Widget-Ausgang - Auswahl einer Aktion, die bei der Änderung des Widget-Status durch den Benutzer aufgerufen wird

3.12. Analog IO indicator

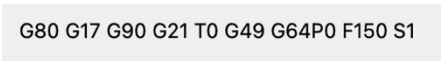


Leuchte zur Anzeige von Zahlenwerten. Wird z.B. verwendet, um den Wert von analogen Hardware-Eingängen/Ausgängen zu präsentieren. Kann auch Werte vergeben.

Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Text	Anzeigetext
Display format	Zeichenkette, die das Format der angezeigten Werte definiert. Kompatibel mit dem „printf“-Standard: https://en.wikipedia.org/wiki/Printf_format_string Z.B. „%.3f“ zeigt einen Fließkommawert mit drei Dezimalstellen an.
Value	Aktueller Wert
Clickable	Zustimmung für die Steuerung des Status der Leuchte durch Mausklick
Color	Auswahl der Farbe der Leuchte
Maximum	Maximaler Wert
Input: text	Widget-Eingang - Auswahl eines Parameters, der den Widget-Text aktualisieren wird
Input: value	Widget-Eingang - Auswahl eines Parameters, der den Faktorwert aktualisieren wird
Output: valueChanged	Widget-Ausgang - Auswahl einer Aktion, die bei der Änderung des Faktorwertes durch den Benutzer aufgerufen wird

3.13. Current G-Codes



Liste des aktuellen Status modaler Befehle (Maschinenzustand).

Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Font	Einstellen der Schriftart für das Widget.
Horizontal alignment	Horizontale Position des Inhalts (Text) des Widgets. • Left – linksbündig • Right – rechtsbündig • Center – mittig • Justify – Ausrichten
Vertical alignment	Vertikale Position des Inhalts (Text) des Widgets. • Top – obenbündig • Top – untenbündig • Center – mittig • Baseline – an Basislinie ausrichten

3.14. MDI Line

Feld für schnelle Befehlseingabe für die Maschine (MDI).

3.15. Python Console

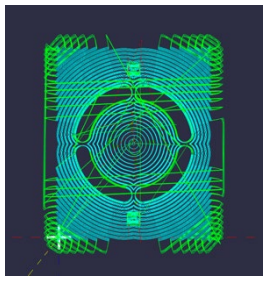
Textfeld, das Meldungen von Python-Makros sowie andere Systeminformationen- und Warnmeldungen anzeigt.

3.16. GCode List

Zeigt den Inhalt der aktuell geladenen gcode-Datei in Textform und die aktuell ausgeführte Linie während der Bearbeitung an. Außerdem kann man mit einem Doppelklick die Linie auswählen, ab der die Bearbeitung beginnen soll.

3.17. Path View

Zeigt den Inhalt der aktuell geladenen gcode-Datei und die aktuelle Position der Maschinenachse in grafischer Form (3D) an.

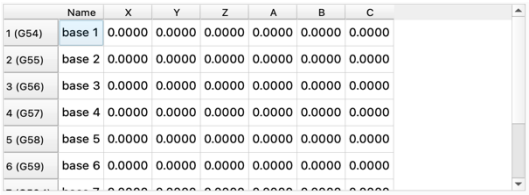


Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Perspective view	Aktivierung der Ansichtsperspektive
Soft limits visible	Aktiviert die Visualisierung von Programmlimits
Default view	Standard-Ansichtstyp: • **Top View** – Draufsicht • **Bottom View** – Ansicht von unten • **Right View** – Ansicht rechts • **Left View** – Ansicht links • **Front View** – Vorderansicht • **Rear View** – Rückansicht • **Isometric view** – Isometrische Ansicht
Standard mit z height mapping	Standardtyp von Höhenmapping (Z) mit Farben: • **None** – ohne Farben • **Color** – Farbmapping • **Grayscale** – Graustufenkartierung

3.18. Offset Table

Widget, das eine Tabelle von Arbeitsoffsets anzeigt und deren Bearbeitung ermöglicht.



	Name	X	Y	Z	A	B	C
1 (G54)	base 1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2 (G55)	base 2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3 (G56)	base 3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4 (G57)	base 4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5 (G58)	base 5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6 (G59)	base 6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

3.19. Group Box

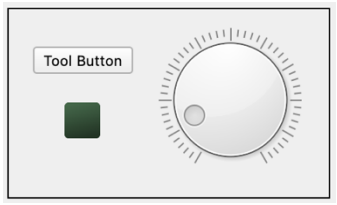
Einer der grundlegenden Container, die für die Gruppierung von Widgets verwendet werden.

Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Layout type	Typ der Autopositionierung in der Gruppe
Title	Angezeigte Bezeichnung der Gruppe

3.20. Frame

Rahmen - einer der grundlegenden Container, die für die Gruppierung von Widgets verwendet werden. Er hat ein etwas anderes Erscheinungsbild als die Group Box (der Titel wird nicht angezeigt).

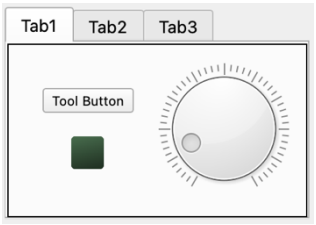


Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Layout type	Typ der Autopositionierung im Rahmen
Shadow	Typ des Rahmenschattens • **Plain** – normal • **Raised** – angehoben • **Sunken** – eingelassen
Shape	Typ der Rahmenform • **No frame** – ohne Rahmen • **Box** – Rechteck • **Panel** – angehobenes oder eingelassenes Bedienfeld • **Styled panel** – Bedienfeld entsprechend dem aktuellen Gui-Stil • **Horizontal line** – horizontale Linie • **Vertical line** – vertikale Linie • **Windows styled panel** – Bedienfeld im Windows 2000-Stil
Line width	Linienbreite
Mid line width	Breite der mittleren Linie

3.21. Tab Box

Container mit Registerkarten für die Gruppierung von Widgets.



Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Tabs quantity	Anzahl der Registerkarten
Current tab	Aktuelle Registerkarte (Standard)

3.22. Scroll Area

Container mit Scrollbars. Nützlich, wenn wir nur begrenzten Platz haben und mehr Elemente auf der Schnittstelle platzieren möchten. Es ist ratsam, ihn zu verwenden, wenn das Schinnstellendesign auf Bildschirmen mit geringerer Auflösung korrekt angezeigt werden soll.

Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Spacing	Abstand zwischen Widgets
Stretch	Skalierungsfaktoren der einzelnen Elemente in einem Container
Layout type	Typ der Autopositionierung • **Free Layout** – ohne Autopositionierung (Layout) • **Horizontal Layout** – horizontale Positionierung • **Vertical Layout** – vertikale Positionierung • **Grid Layout** – Positionierung im Raster • **Form Layout** – vertikale Positionierung von Paaren
Horizontal scroll bar policy	Strategie zur Anzeige des horizontalen Inhalt-Scrollbalkens • **As Needed** – nur wenn nötig • **Always Off** – immer ausgeschaltet • **Always On** – immer sichtbar
Vertical scroll bar policy	Strategie zur Anzeige des vertikalen Inhalt-Scrollbalkens • **As Needed** – nur wenn nötig • **Always Off** – immer ausgeschaltet • **Always On** – immer sichtbar
Shadow	Typ des Rahmenschattens • **Plain** – normal • **Raised** – angehoben • **Sunken** – eingelassen
Shape	Typ der Rahmenform • **No frame** – ohne Rahmen • **Box** – Rechteck • **Panel** – angehobenes oder eingelassenes Bedienfeld • **Styled panel** – Bedienfeld entsprechend dem aktuellen Stil • **Horizontal line** – horizontale Linie • **Vertical line** – vertikale Linie • **Windows styled panel** – Bedienfeld im Windows 2000-Stil
Line width	Linienbreite
Mid line width	Breite der mittleren Linie

3.23. Horizontal Layout

Container für das horizontale Autopositionierungssystem. Gruppiert die Widgets und legt gleichzeitig fest, wie sie positioniert und skaliert werden. Das Autopositionierungssystem ist in einem separaten Kapitel beschrieben.

Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Spacing	Abstand zwischen Widgets
Stretch	Skalierungsfaktoren der einzelnen Elemente in einem Container

3.24. Vertical Layout

Container für das vertikale Autopositionierungssystem. Gruppiert die Widgets und legt gleichzeitig fest, wie sie positioniert und skaliert werden. Das Autopositionierungssystem ist in einem separaten Kapitel beschrieben.

Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Spacing	Abstand zwischen Widgets
Stretch	Skalierungsfaktoren der einzelnen Elemente in einem Container

3.25. Grid Layout

Container des Autopositionierungssystems im Raster. Gruppiert die Widgets und legt gleichzeitig fest, wie sie positioniert und skaliert werden. Das Autopositionierungssystem ist in einem separaten Kapitel beschrieben.

Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Horizontal spacing	Horizontaler Abstand zwischen Widgets
Vertical spacing	Vertikaler Abstand zwischen Widgets
Column stretch	Skalierungsfaktoren der Spalten in einem Container
Row stretch	Skalierungsfaktoren der Zeilen in einem Container

3.26. Form Layout

Container des Autopositionierungssystems im Formular. Gruppiert die Widgets und legt gleichzeitig fest, wie sie positioniert und skaliert werden. Das Autopositionierungssystem ist in einem separaten Kapitel beschrieben.

Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Horizontal spacing	Horizontaler Abstand zwischen Widgets
Vertical spacing	Vertikaler Abstand zwischen Widgets

3.27. Splitter

Container des Autopositionierungssystems mit der Möglichkeit, die Aufteilung und Größe der Elemente dynamisch zu bestimmen. Gruppiert die Widgets und legt gleichzeitig fest, wie sie positioniert und skaliert werden. Das Autopositionierungssystem ist in einem separaten Kapitel beschrieben.

Eigenschaften:

Bezeichnung der Eigenschaften	Beschreibung
Orientation	Methode der Positionierung der Elemente • **Horizontal** – horizontal • **Vertical** - vertikal
Shadow	Typ des Rahmenschattens • **Plain** – normal • **Raised** – angehoben • **Sunken** – eingelassen
Shape	Typ der Rahmenform • **No frame** – ohne Rahmen • **Box** – Rechteck • **Panel** – angehobenes oder eingelassenes Bedienfeld • **Styled panel** – Bedienfeld entsprechend dem aktuellen Stil • **Horizontal line** – horizontale Linie • **Vertical line** – vertikale Linie • **Windows styled panel** – Bedienfeld im Windows 2000-Stil
Line width	Linienbreite
Mid line width	Breite der mittleren Linie

4. Autopositionierungssystem

Das Autopositionierungssystem ist ein Schlüsselelement der simCNC-Benutzerschnittstelle. Es macht die Schnittstelle komfortabel, dynamisch und kann sich weitgehend automatisch an unterschiedliche Bildschirmgrößen anpassen.

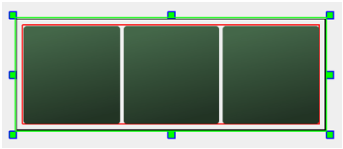
Es lohnt sich, einige Zeit zu widmen, um sich mit den Regeln vertraut zu machen und die Funktionsweise der Elemente dieses Systems zu üben. Dadurch wird es möglich, schnell und bequem attraktive und funktionale Schnittstellen zu gestalten.

4.1. Arten von Containern

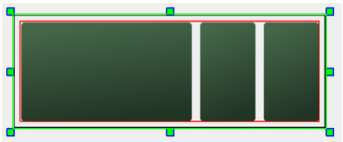
Das Grundelement des Autopositionierungssystems ist der so genannte Container, in dem wir Widgets und andere Container platzieren. Die Art des Containers bestimmt das allgemeine Prinzip der Positionierung der Elemente, die sich darin befinden.

4.1.1. Horizontal Layout

Container mit horizontaler Positionierung. Wie man leicht erraten kann, dient er dazu, Widgets zu gruppieren und sie horizontal anzuordnen.



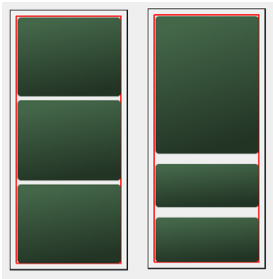
In der obigen Abbildung sehen wir drei Widgets für **Digitale IO Indicator**, die horizontal im Container angeordnet sind.
Es sind zwei Eigenschaften des Containers zu beachten, die den Abstand zwischen den Widgets (**spacing**) und die Aufteilung des Raums für die einzelnen Elemente (**stretch**) kontrollieren.



Oben derselbe Container, aber der Abstand (**spacing**) wurde von „1“ auf „5“ geändert und die Raumaufteilung (**stretch**) wurde auf „3,1,1“ gesetzt. Die Notation „**3,1,1**“ bedeutet, dass die empfohlene Größe des ersten Widgets 3x größer ist als die der anderen. Bitte beachten Sie das Wort **empfohlen**. Die Skalierung der Elemente wird auch durch die **size policy**-Parameter beeinflusst, die in einem separaten Unterabschnitt beschrieben sind. Im obigen Beispiel ist **size policy** für alle drei Elemente im Container auf **Preffered** eingestellt.

4.1.2. Vertical Layout

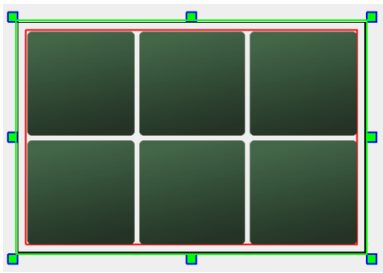
Container mit vertikaler Positionierung. Das Funktionsprinzip dieses Containers ist identisch mit dem von **Horizontal Layout**. Der einzige Unterschied besteht in der vertikalen Positionierung der darin befindlichen Elemente.



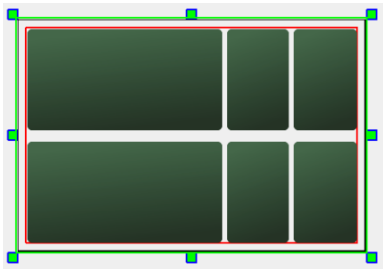
Oben sehen wir zwei Container **Vertikal Layout** mit den gleichen Widgets und zwei Einstellungsvarianten wie bei der Beschreibung des Containers **Horizontal Layout**.

4.1.3. Grid Layout

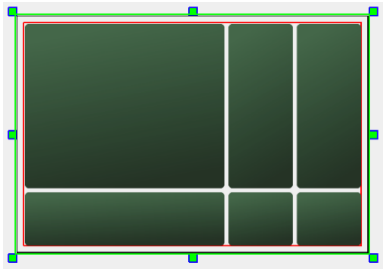
Container mit Positionierung im Raster. In diesem Fall sind die Elemente sowohl horizontal als auch vertikal als Zeilen und Spalten positioniert.



In der obigen Abbildung sehen wir sechs Widgets **Digital IO Indicator**, die im Container **Grid Layout** platziert sind. Wenn es sich um die Kontrolle der Abstände zwischen den Elementen (**spacing**) und die Skalierungsfaktoren (**stretch**) handelt, gilt das gleiche Prinzip wie beim zuvor beschriebenen (**Horizontal**) und (**Vertical Layout**), mit dem Unterschied, dass wir diese Eigenschaften für Spalten und Zeilen getrennt definieren können.



Das nächste Beispiel zeigt die Änderung von **column spacing** auf „1“ und **column stretch** auf „3,1,1“. Dies hat erwartungsgemäß zur Folge, dass der horizontale Abstand zwischen den Elementen kleiner wird und die erste Spalte dreimal so breit ist wie die anderen.

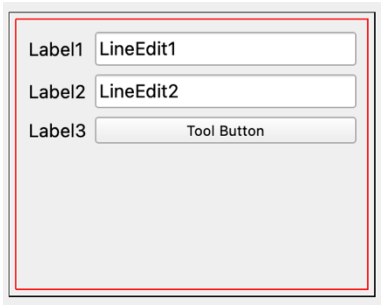


Hier wurden identische Einstellungen für **row spacing** (1) und **row stretch** (3,1,1) vorgenommen. Jetzt sind die Zeilen- und Spaltenabstände gleich und die erste Zeile ist dreimal so breit wie die anderen.

Ein häufiger Fehler, der von weniger erfahrenen Benutzern gemacht wird, ist die übermäßige Verwendung des (**Grid Layout**)-Containers und der Versuch, alle Elemente der Schnittstelle darin zu platzieren. Dieser Ansatz führt zu Problemen, wenn die Anzahl der Elemente wächst. Wenn wir viele Spalten und Zeilen haben, wird es immer schwieriger, die Größe und die Positionierung der Elemente zu kontrollieren. Ein viel besserer Ansatz besteht darin, kleinere Schnittstellenblöcke zu erstellen und diese zu größeren Blöcken zu verbinden. Man sollte nicht vergessen, dass ein Containerelement auch ein anderer Container sein kann.

4.1.4. Form Layout

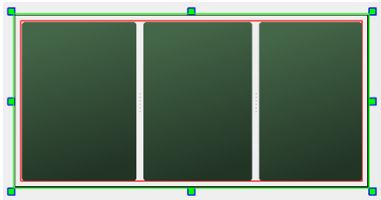
Container mit Positionierung im Formular. Es handelt sich im gewissen Sinne um eine besondere Art der Rasterpositionierung (**Grid Layout**), die speziell für die Erstellung von Formularen wie dem untenstehenden geeignet ist.



(**Form Layout**) positioniert Elementpaare in Reihen an, es ist also eigentlich ein Raster mit zwei Spalten und einer beliebigen Anzahl von Zeilen. Es eignet sich hervorragend für Anwendungen wie im obigen Beispiel, bei dem sich Zeilen vom Typ „**Beschreibung→Widget**“ wiederholen. Sie können dafür natürlich das **Grid Layout** verwenden, aber das Form Layout verfügt über für Formulare optimierte Regeln zur Skalierung von Elementen und bietet oft einen besseren visuellen Effekt mit weniger Aufwand.

4.1.5. Splitter

Dieser Container ähnelt dem **Horizontal** und **Vertikal Layout**, weist aber einen wichtigen Unterschied auf: Während wir z.B. bei **Horizontal Layout** das Größenverhältnis der Elemente dauerhaft festlegen, kann der Bediener bei **Splitter** die Größe der Elemente auch später modifizieren. Mit **Splitter** können wir die Art der Positionierung (Layout) (vertikal/horizontal) durch Einstellen der Eigenschaft „**Orientation**“ auswählen.



Oben sehen wir drei „**Digital IO Indicator**“-Widgets, die horizontal im Splitter positioniert sind.

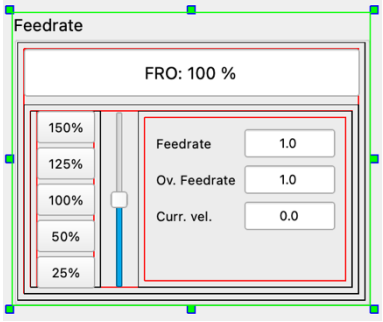
Oben: derselbe Container, aber nach dem Schließen des Editors und der Änderung der Größe seiner Elemente durch Klicken auf den Bereich zwischen den Widgets und Bewegen des Mauszeigers.

Mit diesem Container können wir dem Bediener mehr Kontrolle über die Größe der Schnittstellengruppen geben, was häufig zu einem besseren Eindruck und einem bequemeren Arbeitsablauf führt.

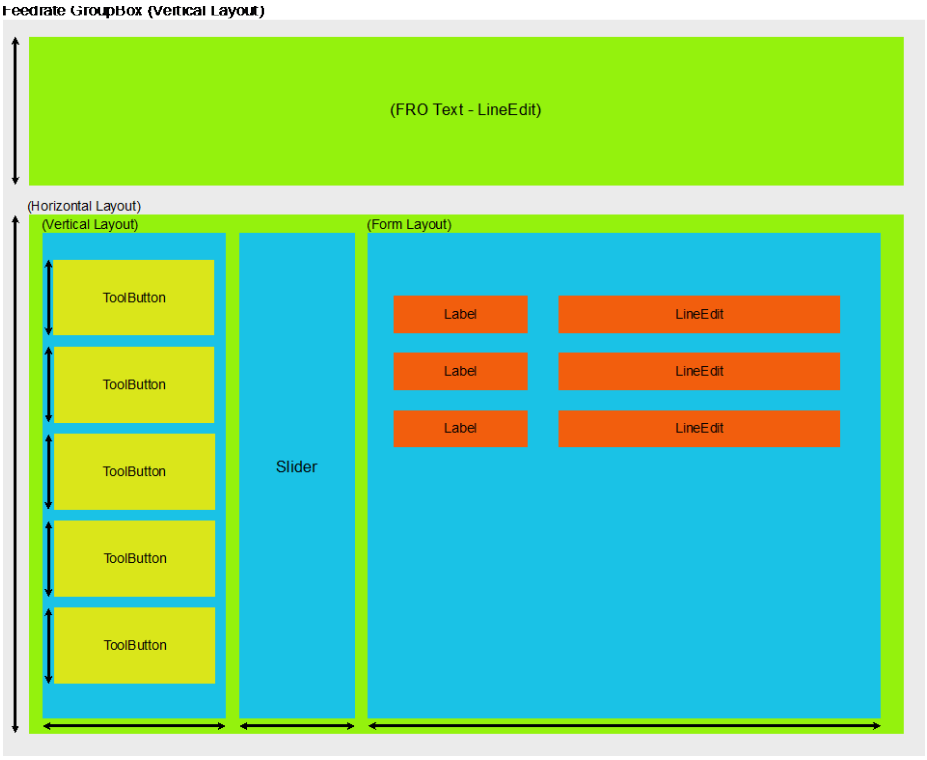
4.2. Verknüpfung von Containern - hierarchische Struktur

Das Verknüpfen von Containern ist die wichtigste Methode, um eine vollberechtigte Schnittstelle zu erstellen, und gibt mehr Kontrolle über die Größe der Elemente im Autopositionierungssystem.

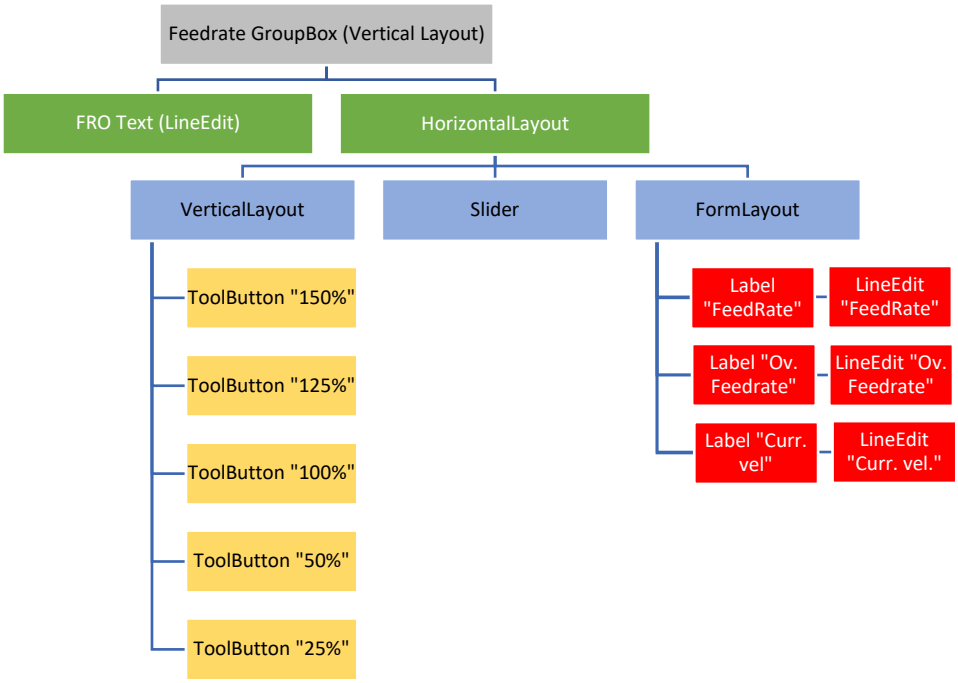
Nehmen wir zum Beispiel einen Teil der Schnittstelle wie den unten angegebenen:



Bei der Gestaltung dieser Art von Blöcken ist es am Anfang schwierig zu entscheiden, welche Art der Positionierung verwendet werden soll. Die Antwort ist die Verwendung mehrerer Container, wie in der folgenden Abbildung dargestellt:



In Baumform sieht das wie folgt aus:



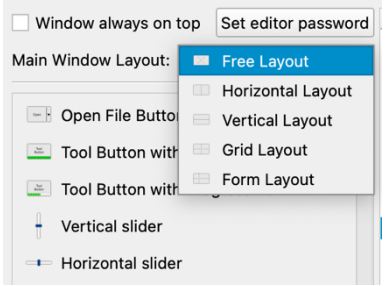
Der Hauptcontainer ist ein **GroupBox**-Widget mit vertikaler Positionierung (**Vertical Layout**). Es enthält zwei grün markierte Elemente: ein **LineEdit**-Widget (Text „FRO: 100 %“) und einen weiteren Container mit horizontaler Positionierung (**Horizontal Layout**). Darin befinden sich wiederum drei blau markierte Elemente: ein Container mit Schaltflächen 25 %-100 % (**Vertical Layout**), ein Schieber für die Geschwindigkeitseinstellung (**Slider**) und ein Container mit einem Formular zur Anzeige und Bearbeitung von Parametern (**Form Layout**).

Es mag zunächst etwas kompliziert erscheinen, aber diese hierarchische Struktur hat viele Vorteile. Unter anderem sorgt sie für Ordnung im Projekt und ermöglicht die einfache Verwendung ganzer Blöcke an verschiedenen Stellen oder sogar in anderen Projekten. Wenn wir beispielsweise eine Gruppe von Schaltflächen aus dem obigen Beispiel in einem anderen Projekt verwenden möchten, sollten wir einfach den entsprechenden Container auswählen und auf CTRL-C klicken. Dann schließen wir den Editor, wechseln den Bildschirm, kehren in den Bearbeitungsmodus zurück und klicken auf CTRL-V.
Mit ein wenig Übung wird das Denken in Block- und Gruppenkategorien im Hinblick auf die Schnittstelle zur Selbstverständlichkeit, und das Projekt beginnt, bequem und schnell zu laufen.

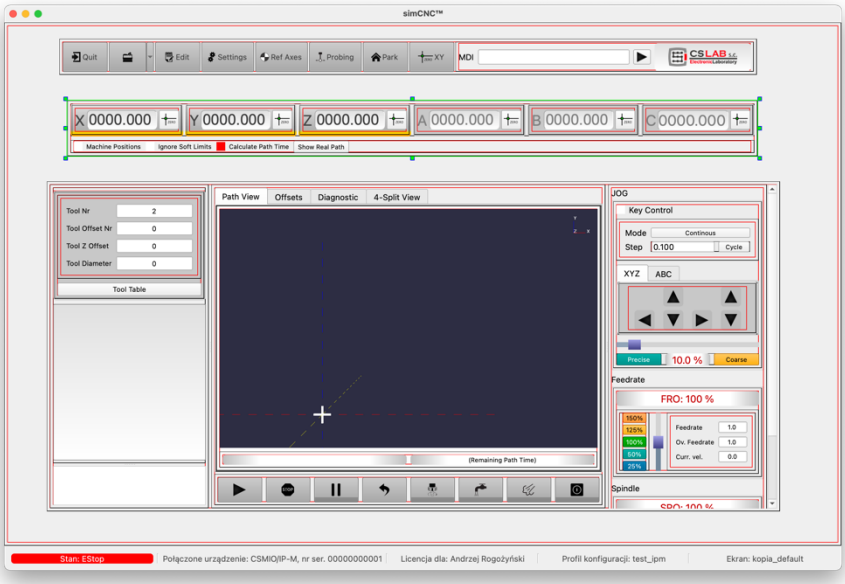
4.3. Positionierung von Elementen im Hauptfenster

Das Hauptfenster des simCNC-Programms ist auch ein Container, für den man die Art der Elementpositionierung auswählen kann. Unter Anlehnung an die im vorigen Unterabschnitt besprochene hierarchische Struktur ist dies der Hauptcontainer, der an der Spitze der Hierarchie steht.

Die Art der Positionierung für das Hauptfenster wird im Editorfenster ausgewählt:



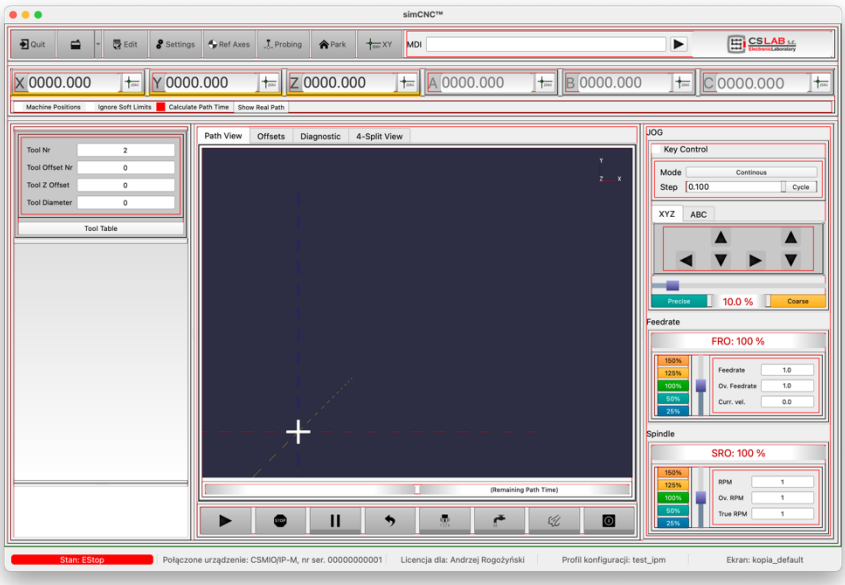
Während **Horizontal**, **Vertikal**, **Grid** und **Form Layout** bereits bekannt sind, wurde „**Free Layout**“ noch nicht erwähnt. „**Free Layout“** bedeutet das Ausschalten der Autopositionierung. Das Ausschalten der Autopositionierung ist in der Anfangsphase der Planung von Bildschirmelementen oder wenn erhebliche Änderungen am Projekt vorgenommen werden sollen, von Vorteil. Die Auswahl der Art der Positionierung für ein Fenster wird getroffen, wenn wir die Hauptgruppen von Elementen entworfen haben.



In der obigen Abbildung sehen wir ein Entwurf der Standard-SimCNC-Schnittstelle mit deaktivierter Autopositionierung des Haupt-Containers.
In diesem Entwurf gibt es drei Hauptgruppen der Schnittstelle:

- Eine Gruppe mit den Untergruppen Schaltflächenleiste, MDI und Logo
- Eine Gruppe mit Untergruppen von Achsenpositions-Widgets und Optionen
- Splitter mit Untergruppen mit den restlichen Elementen

Das Gestaltungskonzept sieht vor, die Hauptgruppen vertikal zu positionieren. Unten ist eine Abbildung nach der Einstellung „**Vertical Layout**“ für den Hauptcontainer zu sehen. Ab diesem Zeitpunkt wird die Änderung der Fenstergröße die automatische Anpassung des gesamten Inhalts an die neue Größe zur Folge haben.



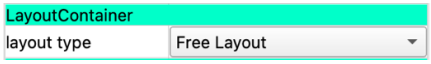
4.4. Widgets mit Containern

Die folgenden Widgets im Schnittstellen-Editor dienen der Gruppierung von Elementen und enthalten Container mit der Möglichkeit, die Art der Positionierung festzulegen:

- **GroupBox**
- **Frame**
- **TabBox**
- **ScrollArea**

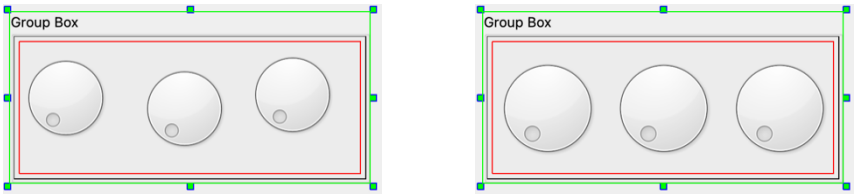
Die Eigenschaften dieser Widgets und ihr Aussehen sind im Kapitel über Widgets beschrieben.

Wie beim Hauptcontainer kann man auch bei den oben genannten Widgets die Auto-Positionierung deaktivieren, indem man „**Free Layout**“ wählt.



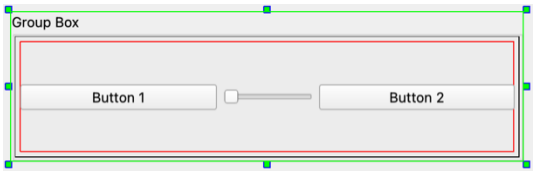
Manchmal kann dies kann nützlich sein, wenn wir den Inhalt eines Widgets vorplanen möchten.

Die folgende Abbildung zeigt ein Beispiel für ein GroupBox-Widget mit drei Dial-Widgets vor und nach der Aktivierung der horizontalen Autopositionierung (**Horizontal Layout**):

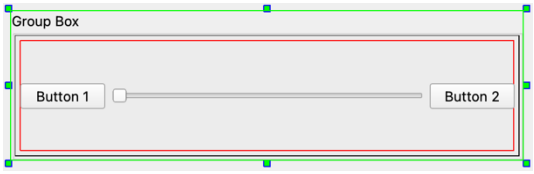


4.5. Raumaufteilung in Containern

Sehr oft ist es notwendig zu definieren, wie das Autopositionierungssystem den verfügbaren Platz in einem Container den einzelnen Elementen zuweisen soll. Nehmen wir die folgende Gruppe als Beispiel:



Hier sehen wir zwei Schaltflächen und einen Schieber in horizontaler Positionierung. Es scheint alles richtig zu sein, aber die Bedienung des Schiebers wäre präziser, wenn er breiter wäre. Der simCNC-Bildschirmeditor ermöglicht es, diesen Aspekt genau zu kontrollieren, indem man die Raumaufteilung im Container (**stretch**) und die Skalierungspolitik der Elemente (**size policy**) definiert.



Oben sehen wir dieselbe Gruppe, aber für den Schieber wurde die horizontale Skalierungspolitik (**horizontal size policy**) auf „**Expanding**“ gesetzt - d.h. der verfügbare Platz wird maximal genutzt.

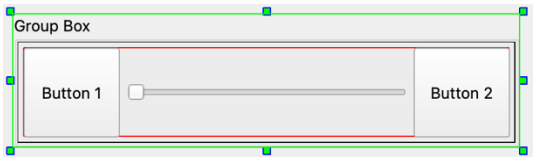
4.5.1. Einstellung der Skalierungsregeln für ein Element (size policy)

Für jedes Element können die folgenden vertikalen (**vertical size policy**) und horizontalen (**horizontal size policy**) Skalierungsregeln definiert werden:

Art	Beschreibung
Fixed	Größe des Elements, die über die Eigenschaften **minimum width** und **minimum height** fest definiert wurde. Werden diese Eigenschaften auf Null gesetzt, wird die Standard-Mindestgröße verwendet.
Minimum	Die Standardgröße des Widgets ist seine Mindestgröße. Das Widget kann größer werden, aber das wird nicht erzwungen.
Maximum	Die Standardgröße des Widgets ist seine maximale Größe. Das Widget kann verkleinert werden, jedoch nicht unter eine Größe, die seine Nutzung unmöglich machen würde.
Preferred	Das Widget kann vergrößert und verkleinert werden, jedoch nicht unter eine Größe, die seine Nutzung unmöglich machen würde. Die Vergrößerung des Elements wird nicht erzwungen.
Expanding	Erzwingen, so viel Raum wie möglich für das Element einzunehmen. Das Widget kann auch verkleinert werden, jedoch nicht unter eine Größe, die seine Nutzung unmöglich machen würde.
Minimum Expanding	Erzwingen, so viel Raum wie möglich für das Element einzunehmen.
Ignored	Dem Element wird so weit wie möglich der erforderliche Raum zugewiesen. Ist der Container zu klein, wird das Element möglicherweise ausgelassen und überhaupt nicht angezeigt.

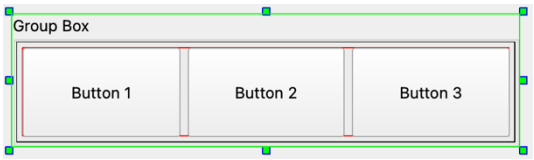
Die in der Praxis am häufigsten verwendeten Einstellungen sind **Fixed**, **Preferred** und **Expanding**.

Nachfolgend ein Beispiel - dieselbe Gruppe wie früher, aber vertical size policy für Schaltflächen wurde von **Fixed** auf **Preferred** geändert. Wie erwartet, wurde die Höhe der Schaltflächen angepasst, um den verfügbaren Raum zu nutzen. Der Schieber nimmt hingegen horizontal den meisten Platz ein, da für die Schaltflächen der Parameter **horizontal size policy** auf **Preferred** und der Schieber auf Expanding eingestellt ist.

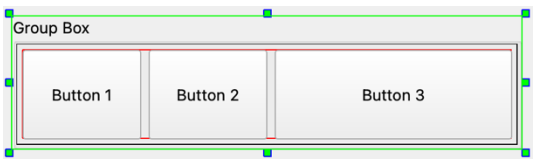


4.5.2. Größenverhältnis der Elemente in einem Container (stretch)

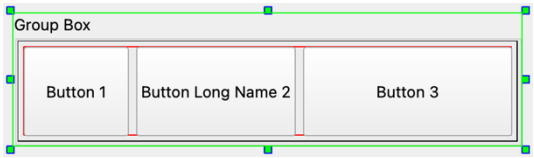
Diese Eigenschaft von Containern wurde bereits bei der Besprechung von Positionierungstypen erwähnt, aber zur Erinnerung - der **stretch**-Parameter kann verwendet werden, um schnell und bequem zu kontrollieren, wie der Raum für die Elemente im Container zugewiesen werden soll. Damit dieser Parameter richtig funktioniert, ist es am besten, die **size policy** für Elemente auf **Preffered** einzustellen (**Fixed** erzwingt beispielsweise immer die Standardgröße des Elements).
Unten ist eine Gruppe von drei Schaltflächen in der horizontalen Positionierung (**Horizontal Layout**) zu sehen. Die Parameter **size policy** für die Schaltflächen sind auf **Preffered** gesetzt und das **stretch**-Feld für den Container ist auf „1,1,1“ gesetzt.



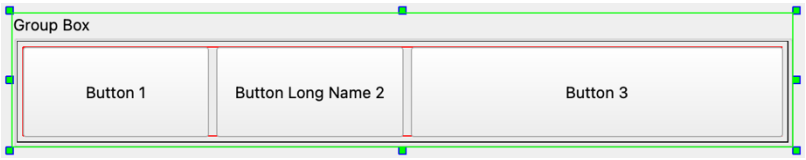
Der Wert „1,1,1“ bedeutet, dass die empfohlene Größe jeder der drei Schaltflächen gleich sein sollte. Beachten wir, was passiert, wenn wir den **stretch**-Parameter des Containers auf „1,1,2“ ändern:



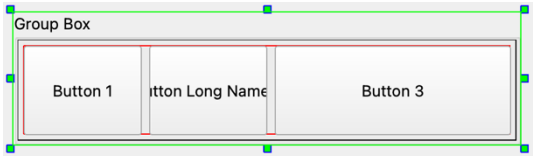
Wie ersichtlich, ist jetzt die dritte Schaltfläche doppelt so groß. Es ist erwähnenswert, dass es bei dem **stretch**-Parameter auf das Verhältnis der Werte ankommt, und nicht auf die absoluten Werte - wenn man also „10,10,20“ angibt, wird der Effekt identisch sein. Die Verwendung größerer Werte kann sinnvoll sein, wenn man die Aufteilung genauer definieren will. Wenn wir „10,10,15“ eingeben, wird die dritte Schaltfläche 1,5x größer als die anderen.
Ein weiterer Punkt ist, dass es sich bei den angegebenen Größen um empfohlene Größen handelt. Sehen Sie sich an, was passiert, wenn der **stretch**-Parameter auf „1,1,2“ eingestellt ist, wir aber die Beschreibung der zweiten Schaltfläche auf eine viel längere ändern.



Wie man sieht, hat das Autopositionierungssystem trotz der empfohlenen Aufteilung mehr Platz für die zweite Schaltfläche bestimmt, um den darin angezeigten Text zu platzieren. Die eingestellte Aufteilung wird nur angewendet, wenn die Containergröße ausreichend ist.



Eine andere Möglichkeit, wenn wir die gewünschte Aufteilung unbedingt beibehalten wollen, besteht darin, die **horizontal size policy** für die zweite Schaltfläche auf **Ignored** zu setzen.



Wie wir sehen, hat die Änderung der Skalierungsregeln für die zweite Schaltfläche dazu beigetragen, dass die auf Basis des **stretch**-Parameters eingestellte Aufteilung beibehalten wurde, allerdings auf Kosten dessen, dass der Text in der Schaltfläche abgeschnitten wurde.

5. Verknüpfung der grafischen Schnittstelle mit dem simCNC-Programm

Widgets können in ihren Eigenschaften Eingänge (**input**) und Ausgänge (**output**) haben. Diese Eigenschaften werden verwendet, um die grafische Schnittstelle mit Funktionen und Werten aus dem simCNC-Programm zu verknüpfen.

Die Eingänge eines Widgets ermöglichen es, seinen Zustand oder Inhalt zu aktualisieren, wenn sich der jeweilige Wert in simCNC ändert. Als Beispiel kann die Anzeige der aktuellen Position einer Achse dienen: Wir erstellen ein **Label**-Widget und wählen in der Liste als „**Input: Text**“ die „**Axis X display position**“ aus. Wenn sich die Position der Achse (in diesem Fall der X-Achse) ändert, wird der Widget-Text aktualisiert.

Die Widget-Ausgänge ermöglichen es, der grafischen Schnittstelle, Aktionen aufzurufen oder Parameter im simCNC-Programm zu ändern. Ein Beispiel dafür ist die Schaltfläche Start der Bearbeitung: Wir erstellen ein **ToolButton**-Widget und als „**Output**: **Clicked**“ wählen wir in seinen Eigenschaften „**Start trajectory**“ aus.

5.1. Widget-Eingangssignale

Bezeichnung	Beschreibung
Anti-dive delay	Verzögerungswert der Anti-dive-Funktion bei automatischer Plasmabrenner-Höhenkontrolle (THC)
Anti-dive velocity	Geschwindigkeitswert der Anti-dive-Funktion bei automatischer Plasmabrenner-Höhenkontrolle (THC)
Axis (…) abs position	Aktuelle absolute Position der Achse (Maschinenkoordinaten)
Axis (…) current work offset	Aktueller Arbeitsoffset der Achse
Axis (…) display position	Aktuelle Position der Achse, mit der möglichen Umschaltung zwischen Maschinen- und Programmkoordinaten
Axis (…) prog position	Aktuelle Programmposition der Achse
Axis (…) tool offset	Werkzeug-Offset in der jeweiligen Achse
Axis (…) tool wear offset	Werkzeugverschleiß-Offset in der jeweiligen Achse
Axis (…) velocity	Bewegungsgeschwindigkeit in der jeweiligen Achse
Calculated path time	Berechnete geschätzte Ausführungszeit der gcode-Datei
Current spindle speed	Aktuelle Spindeldrehzahl
Current torch voltage	Aktuelle Spannung des Plasmabrenner-Lichtbogens
Current velocity	Aktuelle Bewegungsgeschwindigkeit
Feedrate	Soll-Vorschubgeschwindkigkeit
Feedrate override	Soll-Vorschubgeschwindigkeit unter Berücksichtigung des Reglers (FRO)
FRO	Wert des Reglers der Vorschubgeschwindigkeit
GCode file path	Pfad der aktuell geladenen gcode-Datei
GCode line number	Nummer der ausgeführten Linie der gcode-Datei
IO pin value	E/A-Hardware-Pin-Wert
JOG mode	Aktueller JOG-Modus
JOG speed	Soll-JOG-Geschwindigkeit
JOG step	Sprung für JOG-Schrittbetrieb
Machine param	Wert des Maschinenparameters mit der vergebenen Nummer
Override spindle speed	Soll-Spindeldrehzahl unter Berücksichtigung des Reglers (SRO)
Remain path time	Prognostizierte verbleibende Zeit bis zum Abschluss der Ausführung der gcode-Datei
Screen name	Name des aktuell geladenen Bildschirms
Selected tool nr	Nummer des ausgewählten Werkzeugs
Signal value	E/A-Signalwert
Spindle CCW percent	Prozentualer Wert der aktuellen Spindeldrehzahl (Linkslauf)
Spindle CW percent	Prozentualer Wert der aktuellen Spindeldrehzahl (Rechtslauf)
Spindle speed	Soll-Spindeldrehzahl
Spindle tool nr	Nummer des in der Spindel geladenen Werkzeugs
SRO	Wert des Reglers der Spindeldrehzahl

THC init position	Gespeicherte „Z“-Koordinate, bei der die Funktion der automatischen Brennerhöhenkontrolle aktiviert wurde
THC max deviation positive	Maximaler Bewegungsbereich in **positiver** Richtung für die automatische Plasmabrenner-Höhenkontrolle
THC max deviation negative	Maximaler Bewegungsbereich in **negativer** Richtung für die automatische Plasmabrenner-Höhenkontrolle
THC mode	Ausgewählte Betriebsart der automatischen Brennerhöhenkontrolle
THC position deviation	Aktueller Ausgangswert der Funktion der automatischen Brennerhöhenkontrolle – Abstand von THC init position
THC smart analog amplification	Verstärkungswert für den smart-analog-Modus der Funktion der automatischen Brennerhöhenkontrolle
THC velocity	Bewegungsgeschwindigkeit für die Funktion der automatischen Brennerhöhenkontrolle
THC voltage deadband	Spannungsbereich der fehlenden Regelung für automatische Brennerhöhenkontrolle
Tool diameter	Durchmesser des aktuell ausgewählten Werkzeugs
Tool diameter wear	Werkzeugverschleiß-Offset (Durchmesser)
Tool offset number	Nummer des ausgewählten Werkzeug-Offsets
Torch on mode	Detektionsmodus des Plasmabrenner-Lichtbogens
Torch on voltage max	Obere Spannungsgrenze für die Funktion der Detektion des Plasmabrenner-Lichtbogens
Torch on voltage min	Untere Spannungsgrenze für die Funktion der Detektion des Plasmabrenner-Lichtbogens
Torch voltage division factor	Teiler für die Messung der Spannung des Plasmabrenner-Lichtbogens
Torch voltage potentiometer max	Lichtbogenspannungswert für die maximale Position des Potentiometers zur Regelung der Schnitthöhe
Torch voltage potentiometer min	Lichtbogenspannungswert für die minimale Position des Potentiometers zur Regelung der Schnitthöhe
Torch voltage threshold	Schwellenwert der Lichtbogenspannung für die Aufwärts-/Abwärtsbewegung für die Modi „analog“ und „smart-analog“ der Funktion der automatischen Brennerhöhenkontrolle.
Work offset number	Nummer des aktuell ausgewählten Arbeitsoffsets

5.2. Widget-Ausgangssignale

Bezeichnung	Beschreibung
Close simCNC	Schließen des simCNC-Programms
Edit G-Code	Öffnen des Bearbeitungsfenster für die gcode-Datei
Execute probing script	Aktivierung von Python-Makro zur Werkzeugmessung
JOG (…)+ pressed	Start der JOG-Bewegung in positiver Richtung für die Achse
JOG (…)+ released	Stopp der JOG-Bewegung in positiver Richtung für die Achse
JOG (…)- pressed	Start der JOG-Bewegung in negativer Richtung für die Achse
JOG (…)- released	Stopp der JOG-Bewegung in negativer Richtung für die Achse
Open settings window	Öffnen des Fensters für Einstellungen des simCNC-Programms
Path simulation	Simulation der gcode-Datei, Analyse von Geschwindigkeiten, Beschleunigungen u.ä.
Ref (…) axis	Aktivierung des Achsenreferenzverfahrens
Ref all axes	Aktivierung der Achsenreferenzsequenz, die in den simCNC-Einstellungen für die automatische Referenzierung ausgewählt wurden.
Rewind trajectory	Zurückspulen der gcode-Datei an den Anfang
Run script	Aktiviert das angegebene Python-Makro
Run spindle clockwise	Einschalten der Spindeldrehung im Uhrzeigersinn
Run spindle counter-clockwise	Einschalten der Spindeldrehung gegen den Uhrzeigersinn
Set anti-dive delay	Einstellen des anti-dive-Verzögerungswerts für die Funktion der automatischen Plasmabrenner-Höhenkontrolle
Set anti-dive velocity	Einstellen der anti-dive-Geschwindigkeit für die Funktion der automatischen Brennerhöhenkontrolle
Set axis (…) current work offset	Einstellen des Werts des Arbeitsoffsets in der jeweiligen Achse

Set axis (...) prog position	Modifizierung des Arbeitsoffsetwerts in der jeweiligen Achse durch Angabe des aktuellen Werts der Programmposition
Set current tool diameter	Einstellen des Durchmessers für das aktuell ausgewählte Werkzeug
Set current tool diameter wear	Einstellen des Verschleißwerts für das aktuell ausgewählte Werkzeug
Set feedrate	Einstellen des Vorschub-Sollwerts
Set flood on/off	Schaltet das Kühlmittel ein/aus
Set FRO	Setzt den aktuellen Wert des Vorschubgeschwindigkeitsreglers
Set IO pin value	Einstellen des Zustands des (Ausgangs-)Pins
Set JOG mode	Einstellen der Betriebsart der JOG-Funktion
Set JOG speed	Einstellen der JOG-Bewegungsgeschwindigkeit (0 - 100 %)
Set JOG step	Einstellen des Sprungs für den JOG-Schrittmodus
Set machine param	Einstellen des Maschinenparameterwerts
Set mist on/off	Schaltet die Nebelkühlung ein/aus
Set pause on/off	Stoppt/setzt fort die Ausführung der gcode-Datei
Set selected tool number	Setzt die Nummer des aktuell ausgewählten Werkzeugs
Set spindle speed	Setzt die Soll-Spindeldrehzahl
Set spindle tool number	Setzt die Nummer des in der Spindel geladenen Werkzeugs
Set SRO	Setzt den aktuellen Wert des Spindeldrehzahlreglers
Set THC max deviation negative	Setzt den maximalen Bewegungsbereich in negativer Richtung für die Funktion der automatischen Brennerhöhenkontrolle
Set THC max deviation positive	Setzt den maximalen Bewegungsbereich in positiver Richtung für die Funktion der automatischen Brennerhöhenkontrolle
Set THC off	Schaltet die Funktion der automatischen Brennerhöhenkontrolle aus
Set THC on	Schaltet die Funktion der automatischen Brennerhöhenkontrolle ein
Set THC smart-analog amplification	Setzt den Verstärkungswert für den „smart-analog“-Modus der Funktion der automatischen Brennerhöhenkontrolle
Set THC velocity	Setzt die Bewegungsgeschwindigkeit für die Funktion der automatischen Brennerhöhenkontrolle
Set THC voltage deadband	Setzt den zulässigen Bereich von Spannungsschwankungen des Brennerlichtbogens, für den die Höhenkorrektur nicht ausgeführt wird.
Set tool number offset	Setzt die Nummer des Arbeitsoffsets
Set torch on voltage max	Setzt die obere Spannungsgrenze des Lichtbogens für die Funktion der Detektion des Plasmabrenner-Lichtbogens
Set torch on voltage min	Setzt die untere Spannungsgrenze des Lichtbogens für die Funktion der Detektion des Plasmabrenner-Lichtbogens
Set torch voltage division factor	Setzt den Wert des Teilers für die Messung der Spannung des Plasmabrenner-Lichtbogens
Set torch voltage potentiometer max	Setzt den Sollwert der Lichtbogenspannung für die maximale Position des Potentiometers
Set torch voltage potentiometer min	Setzt den Sollwert der Lichtbogenspannung für die minimale Position des Potentiometers
Set torch voltage threshold	Einstellen des Schwellenwerts für die Aufwärts-/Abwärtsbewegung des Brenners („analog“- und „smart-analog“-Modus der Funktion der automatischen Brennerhöhenkontrolle)
Set work offset number	Setzt die Nummer des Arbeitsoffsets
Show real trajectory	Schaltet die Ansicht der realen Trajektorie der Bewegung ein, unter Berücksichtigung der Optimierung und der konstanten Schnittgeschwindigkeit.
Start trajectory	Aktivierung des Bearbeitung - startet die aktuell geladene gcode-Datei
Stop trajectory	Stoppen der Bearbeitung
Switch to EStop/Idle state	Umschalten des simCNC-Programms zwischen Stopp- und Bereitschaftsstatus
Tool table	Zeigt das Fenster mit der Werkzeugliste an

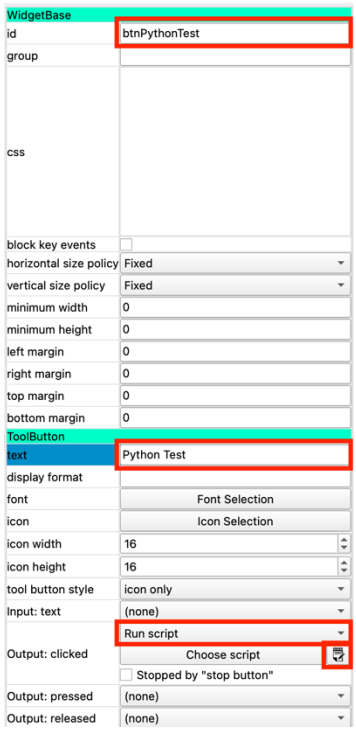
6. Verbindung der grafischen Schnittstelle mit Python-Skripten

Das System der grafischen simCNC-Benutzerschnittstelle ermöglicht Folgendes:

- Ausführen von Benutzerskripten auf Schaltflächendruck
- Referenzieren auf Widgets vom Skriptniveau aus
    - Aktivierung von Widget-Aktionen
    - Lesen und Ändern von Widget-Eigenschaften (z.B. Text)

6.1. Aufrufen eines Skripts durch Klicken auf eine Schaltfläche auf dem Bildschirm

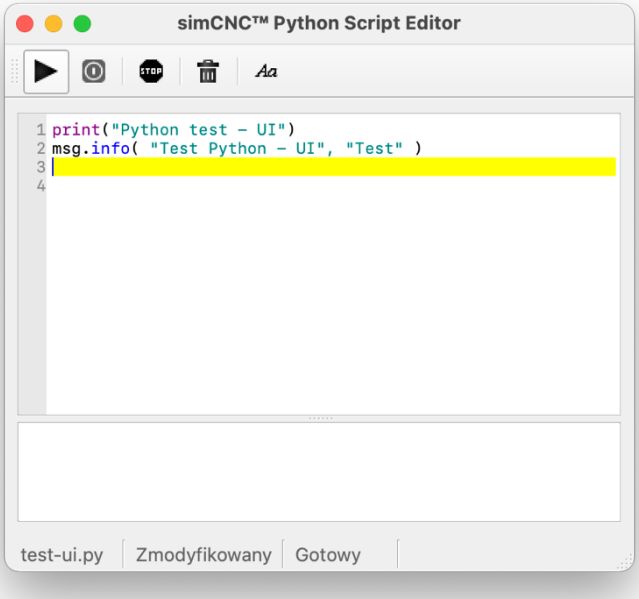
Der vorherige Abschnitt enthielt eine Liste möglicher Widget-Ausgangssignale. Häufig besteht jedoch die Notwendigkeit, eine benutzerdefinierte Funktion zu implementieren, die vom Bediener über die grafische Bedienerschnittstelle aufgerufen wird. Ein Beispiel dafür, wie dies realisiert werden kann, ist unten dargestellt:



Wir erstellen eine Schaltfläche (**ToolButton**) im Schnittstelleneditor. Wir geben ihr einen Identifizierer - z.B. **btnPythonTest**, man kann auch ein Etikett für die Schaltfläche festlegen, z.B. „Python Test“.

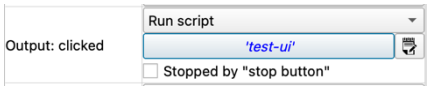
Anschließend setzen wir **Run script** für die Eigenschaft **Output:clicked** der Schaltfläche. Es wird das Skriptauswahlfenster geöffnet. Wir schließen es, weil wir gleich eine neue Datei erstellen.

Wir öffnen den Skript-Editor (das Symbol neben der Schaltfläche **Choose script**).



Wir geben den Skriptcode in den Editor ein (wie oben). Unser Beispielskript zeigt eine Meldung in der Konsole und ein Informationsfenster an.

Wir zeichnen eine Datei auf, vorzugsweise im Verzeichnis mit den Daten unseres entworfenen Bildschirms. In unserem Beispiel hat der Bildschirm den Namen „my_test_screen“. Wir gehen also in das Unterverzeichnis „screens/my_test_screen/scripts“ in Bezug auf den Installationsort von simCNC und geben der Datei einen Namen - zum Beispiel „test-ui“.



Dann schließen wir das Python-Editorfenster, klicken in den Widget-Eigenschaften auf **Choose script** und wählen die Datei „test-ui“ aus. Wir zeichnen die Änderungen im Projekt auf, indem wir auf **Save** klicken, und schließen den Schnittstellen-Editor - so haben wir eine Schaltfläche erstellt, die beim Anklicken unser Python-Makro aktiviert wird. Man kann seine Funktion durch Klicken auf die Schaltfläche überprüfen. In der Python-Konsole (sofern wir dieses Widget im Projekt haben) sollte der Text „Python test - UI“ erscheinen und zusätzlich sollte ein Fenster mit der Meldung „Test Python - UI“ erscheinen.

6.2. Verweis auf Schnittstellenelemente vom Python-Skriptniveau aus

Das System der grafischen simCNC-Schnittstelle ermöglicht es, auf Widgets vom Python-Skriptniveau aus zu verweisen. Wann ist dies notwendig? Stellen wir uns vor, dass wir eine Werkzeugmaschine mit automatischem Werkzeugwechsel haben und den Status oder Fehlercode des Werkzeugwechslers auf dem Bildschirm anzeigen lassen möchten. Der Werkzeugwechsel wird am häufigsten durch das **M6**-Maschinenmakro realisiert. Der **M6**-Makrocode muss daher in der Lage sein, den Widgettext auf dem Bildschirm zu aktualisieren, um die gewünschten Informationen anzuzeigen.

Bei der Erstellung des Makrocodes im Editor, der in simCNC integriert ist, wird automatisch eine Klasse mit dem Namen **gui** importiert, die alle grafischen Elemente des geladenen Bildschirms enthält. Auf ein konkretes Element wird über seinen Widget-Identifizierer (seine **id**-Eigenschaft) verwiesen:

gui.<id>.<Bezeichnung der Methode>(<Argumente>)

Wenn wir das Widget-Beispiel aus dem vorherigen Unterabschnitt verwenden und den Text auf der Schaltfläche ändern möchten, gehen wir wie folgt vor:

gui.btnPythonTest.setText("Some New Text")

6.2.1. Methoden der Widget-Klasse

Die Widget-Klasse stellt dem Programmierer die folgenden Methoden zur Verfügung:

Bezeichnung der Methode	Beschreibung
executeClickedOutput	Aktiviert die auf Mausklick der linken Maustaste definierte Aktion Argumente: (keine) Rückgabewert: (keine)
executePressedOutput	Aktiviert die Aktion, die auf Mausklick der linken Maustaste auf einem Widget definiert wurde Argumente: (keine) Rückgabewert: (keine)
executeReleasedOutput	Aktiviert die Aktion, die auf Loslassen der linken Maustaste auf einem Widget definiert wurde Argumente: (keine) Rückgabewert: (keine)
executeOutput	Aktiviert die für den angegebenen Widget-Ausgang definierte Aktion. Argumente: • Name des Widget-Ausgangs • Rückgabewert: (keine)
getAttribute	Übernahme des Wertes einer Widget-Eigenschaft. Argumente: • Bezeichnung der Eigenschaften Rückgabewert: • Eigenschaftswert
getAttributes	Übernahme der Liste von Widget-Eigenschaften Argumente: (keine) Rückgabewert: • Liste der Namen von Widget-Eigenschaften
getOutputs	Übernahme der Liste der Widget-Ausgangssignale Argumente: (keine) Rückgabewert: • Liste der Namen der Widget-Ausgangssignale

getText	Übernahme der Eigenschaft **Text** des Widgets. Argumente: (keine) Rückgabewert: • Inhalt der Eigenschaft **Text** des Widgets
setAttribute	Einstellen der Widget-Eigenschaften Argumente: • Bezeichnung der Eigenschaften • Eigenschaftswert Rückgabewert: (keine)
setText	Einstellen der Eigenschaft **Text** des Widgets Argumente: • Wert für die Eigenschaft **Text** des Widgets Rückgabewert: (keine)

6.2.2. Änderung der Gestaltung des Widgets

Mit der **setAttribute**-Methode können wir das Stylesheet (**css**) für das Widget dynamisch definieren. Der Name der Eigenschaft lautet **styleSheet**. Die folgenden Beispiele verwenden **btnPythonTest** als Widget-Identifizierer aus dem Unterabschnitt 6.1.

Beispiel 1 (Wechsel der Hintergrundfarbe auf rot):

gui.btnPythonTest.setAttribute(„styleSheet", „background-color: red;")

Beispiel 2 (Wechsel der Hintergrundfarbe auf rot und der Schriftfarbe auf gelb):

gui.btnPythonTest.setAttribute(„styleSheet", „background-color: red; color: yellow;")

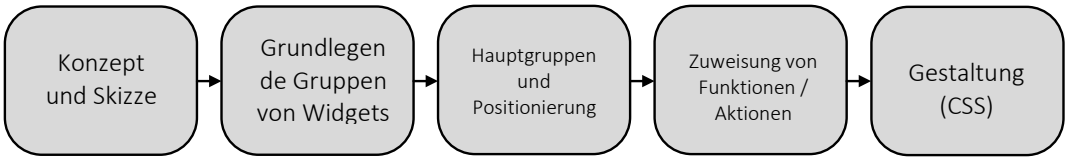
Beispiel 3 (Änderung der Schriftgröße):

gui.btnPythonTest.setAttribute(„styleSheet", „font-size: 24px;")

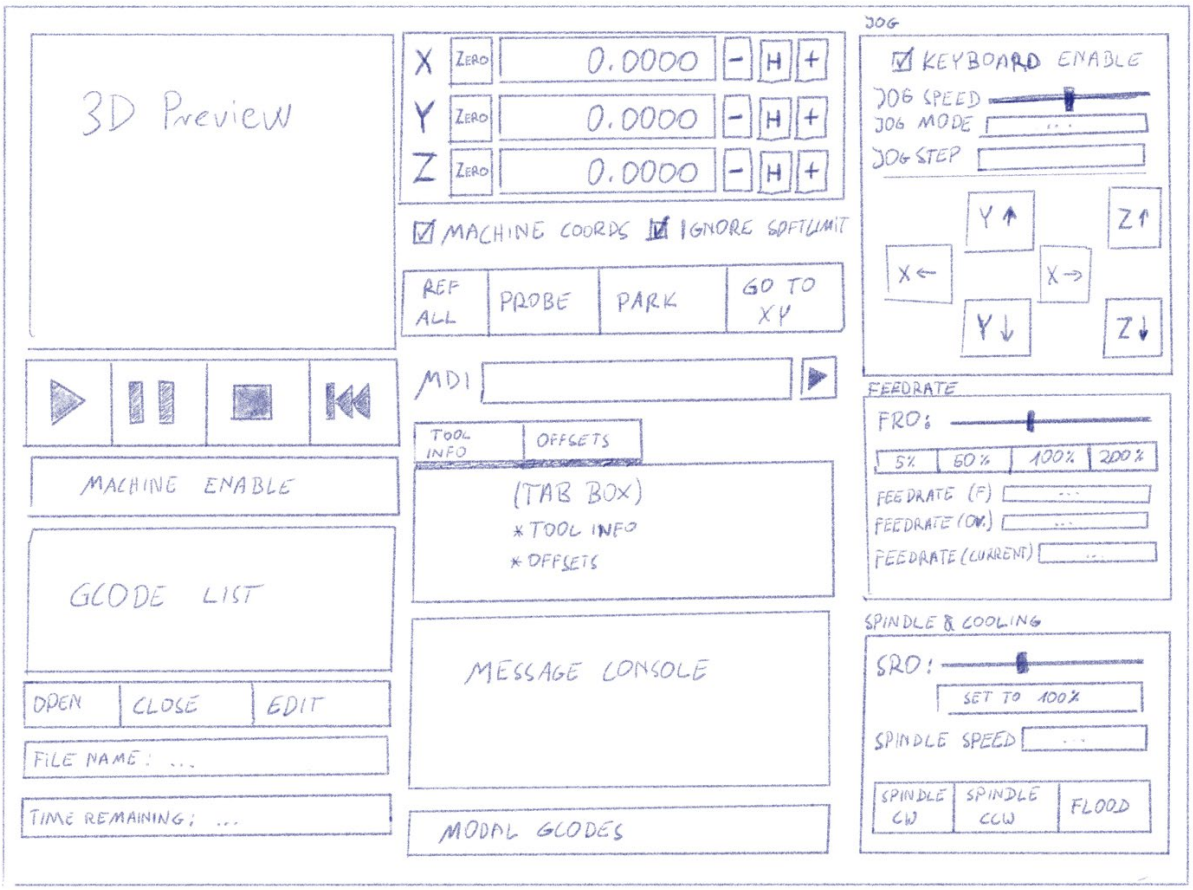
Anhang - Schritt-für-Schritt-Schnittstellengestaltung

In diesem Anhang wurde der Aufbau einer vollständigen Schnittstelle Schritt für Schritt dargestellt - unter Verwendung der in den vorangegangenen Abschnitten enthaltenen Informationen.

Zur Erinnerung - so wird die Reihenfolge der Aufgaben während des Bauprozesses sein:



Konzept und Skizze



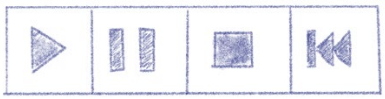
Wir gehen von dem Aufbau einer kompakten Schnittstelle für eine dreiachsige Fräsmaschine aus, wie in der obigen Skizze dargestellt. Wie man sieht, reicht sogar eine Freihandzeichnung aus. Falls die Bestimmung mancher Elemente nicht klar ist, können wir sicher sein, dass im folgenden Abschnitt alles der Reihe nach erklärt wird.

Erstellen eines neuen Schnittstellenentwurfs und Beginn der Bearbeitung

- Wir wählen eine Position aus dem folgenden Menü aus: **Konfiguration→Bildschirm einstellen**
- Im Bildschirmauswahlfenster klicken wir auf die Schaltfläche **Neu erstellen**
- Wir geben ihm einen Namen, zum Beispiel „ui_example“.
- Wir wählen den Namen der neu erstellten Schnittstelle in der Liste aus und klicken auf die Schaltfläche **Laden**
- Wir wählen eine Position aus dem folgenden Menü aus: **Konfiguration→Schnittstelleneditor öffnen**

Grundlegende Gruppen von Widgets

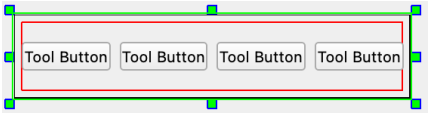
Gruppe der Schaltflächen „Start“, „Pause“, „Stop“ und „Scrollen“



Da wir hier vier Schaltflächen in der horizontalen Positionierung haben, werden wir den Container **Horizontal Layout** verwenden.

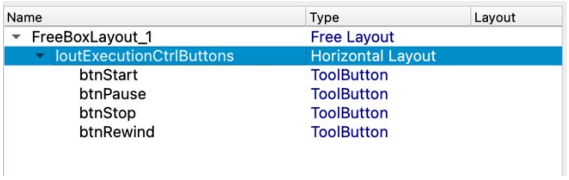
- Wir ziehen aus der Widget-Liste im Editorfenster **Horizontal Layout** in das simCNC-Fenster
- Aus der gleichen Liste ziehen wir die vier **Tool-Button**-Widgets in den Container

Im Ergebnis sollten wir mehr oder weniger Folgendes erhalten:

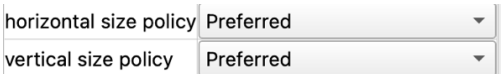


- Wir ändern die Standardnamen der Elemente so, dass man später leicht erkennen kann, welchen Zweck sie haben. Hier wurde Folgendes angenommen: **loutExecutionCtrlButtons** für den Container und **btnStart**, **btnPause**, **btnStop** und **btnRewind** für die Schaltflächen. Der Name wird im **id**-Feld der Widget-Eigenschaft definiert.

Der Objektbaum im Editorfenster sollte wie folgt aussehen:

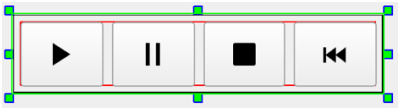


- Anschließend ändern wir die Skalierungsregeln für die Schaltflächen, damit sich ihre Größe an die Containergröße anpassen kann. Wir klicken auf **btnStart** und mit gedrückt gehaltener Shift-Schaltfläche auf **btnRewind**. Wenn alle vier Schaltflächen ausgewählt sind, ändern wir die Eigenschaften **horizontal** und **vertical size policy** auf **Preffered**.

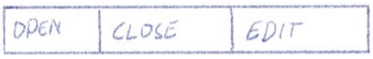


- Wir legen Symbole für die Schaltflächen fest, indem wir das Objekt auswählen und auf die Schaltfläche **Icon Selection** neben der Eigenschaft **Icon** des Widgets klicken
  (die in diesem Beispiel verwendeten Symbole können von https://soft.cs-lab.eu/ui_example_icons.zip heruntergeladen werden)
  - **btnStart** → „icon_play.png”
  - **btnPause** → „icon_pause.png”
  - **btnStop** → „icon_stop.png”
  - **btnRewind** → „icon_rewind.png”

Das Aussehen der Gruppe entspricht jetzt dem, was wir erreichen wollten:



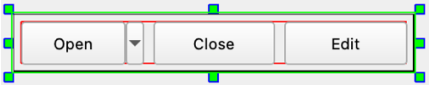
Gruppe von Schaltflächen „Open“, „Close“ und „Edit“



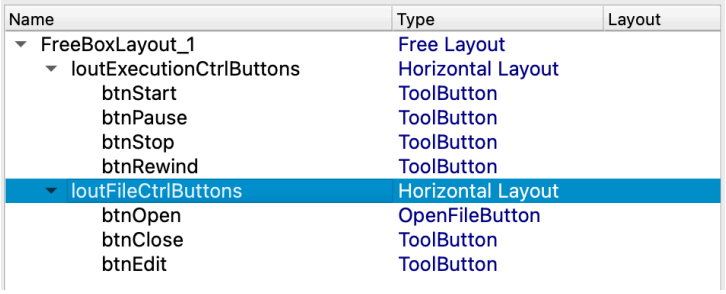
Eine sehr ähnliche Situation wie die vorherige - drei Schaltflächen in horizontaler Positionierung, mit dem Unterschied, dass wir für die Schaltfläche **Open** ein dediziertes Widget benutzen, in dem die Liste der zuletzt geöffneten Dateien gespeichert ist.

- Wir ziehen ein weiteres Objekt **Horizontal Layout** aus dem Editorfenster in das simCNC-Fenster
- Wir ziehen ein **Open File Button**-Widget und zwei **Tool Button**-Widgets in den Container
- Wir geben dem Container und den Widgets die Namen (Eigenschaft **id**)
    - **loutFileCtrlButtons** für den Container
    - **btnOpen**, **btnClose** und **btnEdit** für die Schaltflächen
- Wir wählen alle drei Schaltflächen aus und ändern **horizontal** und **vertical size policy** auf **Preffered**, damit sich die Größen der Schaltflächen an die Containergröße anpassen können
- Wir legen das Schaltflächen-Etikett fest, indem wir die Eigenschaft **Text** der Widgets modifizieren
    - „Open”, „Close” und „Edit” entsprechend für **btnOpen**, **btnClose** und **btnEdit**

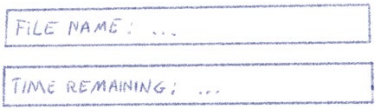
Die Gruppe ist fertig und sieht wie folgt aus:



Der Objektbaum unserer Schnittstelle wiederum sollte jetzt wie folgt aussehen:



Gruppe Dateiname und Bearbeitungszeit



Die Skizze lässt vermuten, dass es sich hier um separate Gruppen handelt, aber wir sollten nicht vergessen, dass die Skizze nur ein allgemeines Konzept ist. Häufig werden während des Planungsprozesses kleinere Änderungen vorgenommen, sei es, um das Design zu verbessern oder die Konstruktion zu vereinfachen. Das ist kein Problem, solange wir das Konzept nicht komplett ändern. In diesem Fall wäre es sinnvoll, eine neue Skizze zu erstellen.

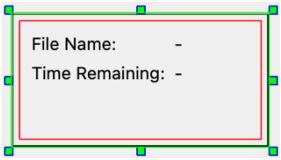
- Wir ziehen das Objekt **Form Layout** aus dem Editorfenster in das simCNC-Fenster
- Wir geben den Namen dem hinzugefügten Container (**id**-Eigenschaft) auf **loutFileInfo**
- Für den Container setzen wir die Eigenschaft **vertical size policy** auf **Maximum** - dadurch wird verhindert, dass das Widget beim Skalieren des Fensters unnötig vergrößert wird
- Wir ziehen die vier **Label**-Elemente in den hinzugefügten Container und geben ihnen Namen (**id**) gemäß der folgenden Tabelle:

lbFileNameDesc	lbFileName
lbTimeRemainingDesc	lbTimeRemaining

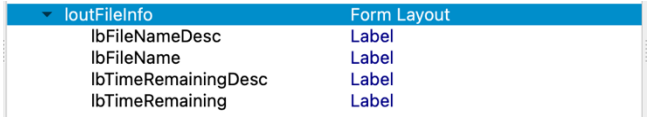
- Wir ändern die Eigenschaft **Text** der hinzugefügten Etiketten gemäß der folgenden Tabelle:

File Name	-
Time Remaining	-

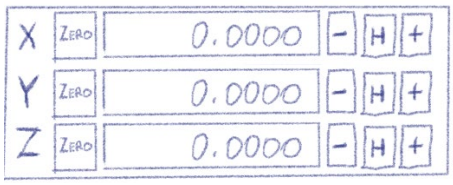
Die Gruppe ist fertig und sollte wie folgt aussehen:



Und so sollte sie im Objektbaum aussehen:



Gruppe der Achsenpositionsanzeiger



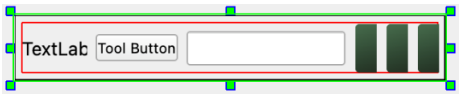
Für jede Achse haben wir hier eine Gruppe von sechs Widgets in horizontaler Positionierung:

- Achsenbezeichnung (**Label**)
- Schaltfläche zum Zurücksetzen der Programmposition (**Tool Button**)
- Positionsanzeige- und Editierfeld (**Line Edit**)
- Drei End- und Basisschalter-Kontrollleuchten (**Digital IO indicator**)

Wir beginnen mit der Erstellung einer Gruppe für eine Achse:

- Wir ziehen den Container **Horizontal Layout** aus dem Editorfenster in das simCNC-Fenster
- Wir ziehen die Widgets eines nach dem anderen in den Container und ordnen sie von links nach rechts im Container an:
    - **Label**
    - **Tool Button**
    - **Line Edit**
    - 3x **Digital IO Indicator**

Wir sollten mehr oder weniger Folgendes erhalten:



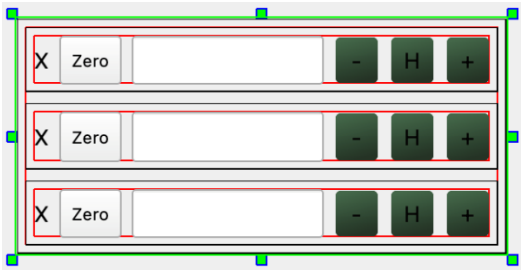
- Wir setzen **horizontal** und **vertical size policy** für die Schaltfläche (**Tool Button**) auf **Preffered**.
- Wir setzen die Eigenschaft **Text** für die Schaltfläche auf „Null“.

- Wir setzen **vertical size policy** für das Bearbeitungsfeld (**Line Edit**) auf **Preffered**
- Wir setzen die Eigenschaft **Text** des Label-Objekts auf **X**
- Wir setzen die Eigenschaft **Text** der Kontrollleuchten **Digital IO indicator** nacheinander auf (von links) „-“, „H“ und „+“.

Die Gruppe der einen Achse ist praktisch fertig:

- Wir ziehen den Container **Vertical Layout** aus dem Editorfenster in das simCNC-Fenster
- Wir wählen den Container aus, dem gerade Widgets hinzugefügt wurden, und drücken CTRL-X (Ausschneiden)
- Wir wählen den leeren Container **Vertical Layout**, den gerade hinzugefügt wurde, und drücken dreimal STRG-V (Einfügen)

Der Effekt sollte wie folgt aussehen:



- Wir legen die Namen (**id**-Eigenschaft) der Widgets fest
  - Gruppe der X-Achse – Name des Containers: **loutAxisXDRO** und Widgets nacheinander, von links:
    - **lbAxisXName**
    - **btnAxisXZero**
    - **edAxisXPosition**
    - **ioAxisXLimitNeg**
    - **ioAxisXHoming**
    - **ioAxisXLimitPos**
  - Gruppe der Y- und Z-Achsen – identisch wie X-Achse, wobei in dem Namen ersetzen wir überall „AxisX“ entsprechend durch „AxisY“ und „AxisZ“
  - Name des übergeordneten Containers (**Vertical Layout**): **loutAxesDROs**

So sollte der Objektbaum aussehen:

Name	Type	Layout
FreeBoxLayout_1	Free Layout	
loutExecutionCtrlButtons	Horizontal Layout	
loutFileCtrlButtons	Horizontal Layout	
loutAxesDROs	Vertical Layout	
loutAxisXDRO	Horizontal Layout	
lbAxisXName	Label	
btnAxisXZero	ToolButton	
edAxisXPosition	LineEdit	
ioAxisXLimitNeg	DigitalIOControl	
ioAxisXHoming	DigitalIOControl	
ioAxisXLimitPos	DigitalIOControl	
loutAxisYDRO	Horizontal Layout	
lbAxisYName	Label	
btnAxisYZero	ToolButton	
edAxisYPosition	LineEdit	
ioAxisYLimitNeg	DigitalIOControl	
ioAxisYHoming	DigitalIOControl	
ioAxisYLimitPos	DigitalIOControl	
loutAxisZDRO	Horizontal Layout	
lbAxisZName	Label	
btnAxisZZero	ToolButton	
edAxisZPosition	LineEdit	
ioAxisZLimitNeg	DigitalIOControl	
ioAxisZHoming	DigitalIOControl	
ioAxisZLimitPos	DigitalIOControl	

Man sollte nicht vergessen, das Projekt von Zeit zu Zeit zu speichern, indem man STRG-S drückt oder im Editorfenster auf **Save** klickt.

Gruppe von Checkboxen „Machine coords“ und „Ignore Soft Limit“

Gruppe von zwei Widgets Typ **Checkbox** in horizontaler Positionierung:

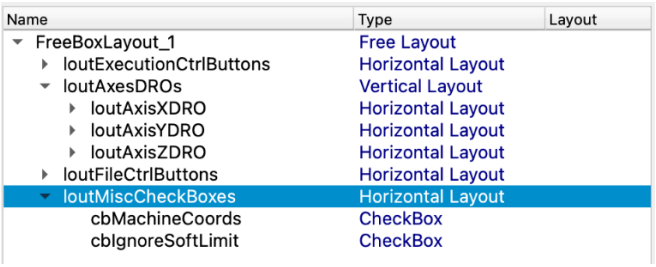
- Umschalten zwischen der Anzeige von Maschinen- und Programmkoordinaten (**Machine Cords**)
- Deaktivierung von Programm-Grenzwerten (**Ignore SoftLimit**)

Wir beginnen wie üblich mit dem Hinzufügen eines Containers (**Horizontal Layout**):

- Wir ziehen den Container **Horizontal Layout** aus dem Editorfenster in das simCNC-Fenster
- Wir ziehen zwei **Checkbox**-Widgets in den Container
- Wir legen die Namen (**id**-Eigenschaft) fest
  - Container: **loutMiscCheckboxes**
  - **cbMachineCoords** und **cbIgnoreSoftLimit** für die Widgets
- Wir legen den angezeigten Text (Eigenschaft **Text**) für die **Checkbox**-Widgets fest
  - „Machine Coords“ und „Ignore SoftLimit“

Im Effekt sollte die Gruppe wie folgt aussehen:

Und so sollte die Gruppe im Objektbaum des Projektes aussehen:



Gruppe der Schaltflächen „Ref All“, „Probe“, „Park“ und „Go To XY“

Wir haben hier eine Gruppe von vier Schaltflächen (**Tool Button**) in horizontaler Positionierung (**Horizontal Layout**).

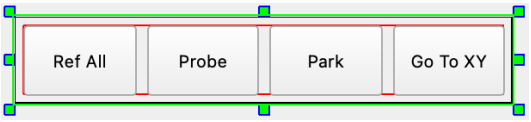
- „Ref All“ → Aufrufen der Achsenreferenzierung
- „Probe“ → Ausführen der Werkzeugmessung
- „Park“ → Bewegung der Maschinenachse in die Parkposition
- „Go To XY“ → Bewegung der X-, Y-Achse zum Programm-Nullpunkt

Wir beginnen wieder mit dem Hinzufügen eines Containers:

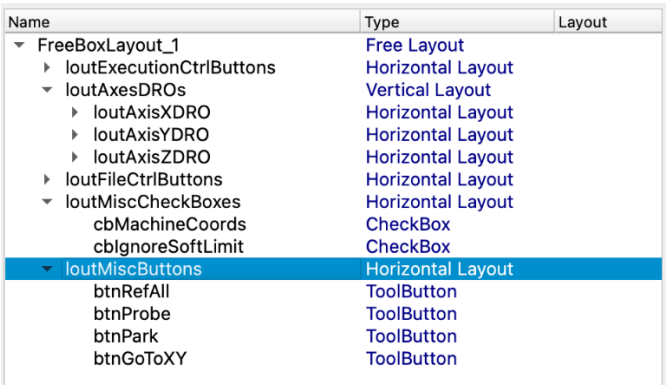
- Wir ziehen den Container **Horizontal Layout** aus dem Editorfenster in das simCNC-Fenster
- Wir ziehen die vier Schaltflächen **Tool Button** in den Container
- Wir wählen (bei gedrückter Shift-Schaltfläche) die Schaltflächen-Widgets aus und ändern die Eigenschaften **horizontal** und **vertical size policy** auf **Preffered**, so dass sich ihre Größe automatisch an den Container anpassen kann
- Wir legen die Namen der Objekte (**id**-Eigenschaft) fest:
  - Name des Containers: **loutMiscButtons**
  - Widget-Namen: **btnRefAll**, **btnProbe**, **btnPark**, **btnGoToXY**

- Wir setzen die Eigenschaft **Text** für die Schaltflächen:
  - Entsprechend (von links): „Ref All“, „Probe“, „Park“, „Go To XY“

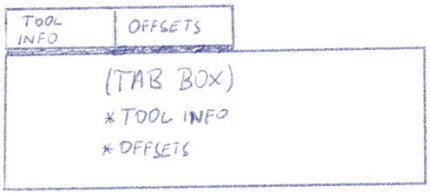
Der Effekt sollte wie folgt aussehen:



Und so sollte die Gruppe im Objektbaum aussehen:

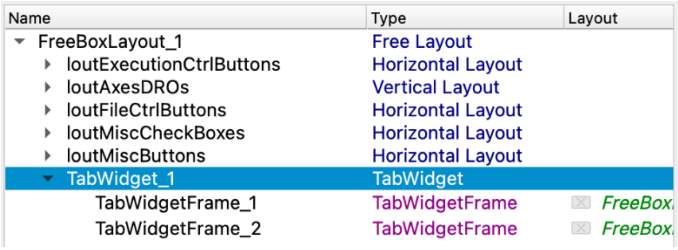


Widget mit den Registerkarten „Tool Info“ und „Offsets“

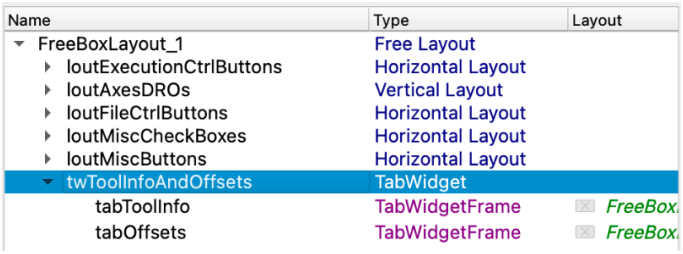


- Wir ziehen das Widget **Tab Box** aus dem Editorfenster in das simCNC-Fenster
- Wir wählen das Widget aus und setzen die Anzahl der Registerkarten (Eigenschaft **tabs quantity**) auf „2“.

Wir sehen drei neue Objekte im Objektbaum: **TabWidget**, das die Anzeige und das Umschalten der Registerkarten kontrolliert, und zwei **TabWidgetFrame**-Objekte, die Container für die einzelnen Registerkarten sind – hier werden wir die Widgets platzieren.

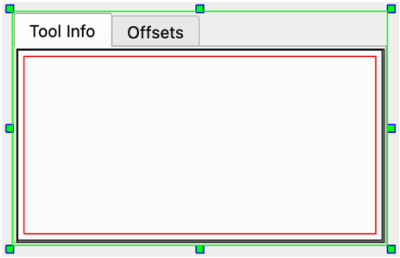


- Wir vergeben die Namen (**id**-Eigenschaft):
  - **TabWidget**-Objekt: **twToolInfoAndOffsets**
  - Registerkarten: **tabToolInfo** und **tabOffsets**

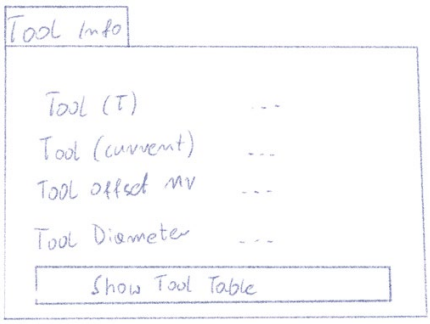


- Wir legen Etiketten für die Registerkarten fest (Eigenschaft **title**) - man muss zunächst die entsprechende Registerkarte im Objektbaum auswählen, indem z.B. auf **tabToolInfo** geklickt wird:
    - „Tool Info“ für **tabToolInfo** und „Offsets“ für **tabOffsets**

Unser Widget sollte jetzt wie folgt aussehen:



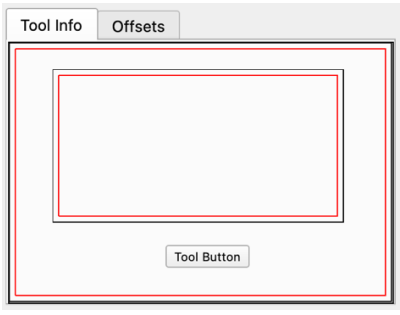
Jetzt können wir mit der Erstellung von Registerkarteninhalten beginnen. Nachstehend ist das Konzept der Registerkarte **Tool Info** dargestellt:



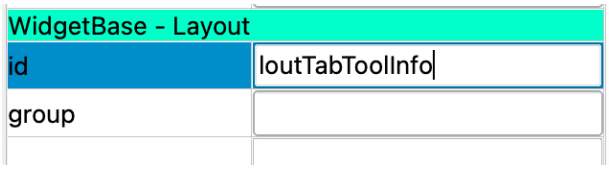
Hier sehen wir einige Objekte Typ **Label** – Beschreibungen und angezeigte Werte (drei Punkte in der Skizze) und eine Schaltfläche **Show Tool Table** zum Öffnen des Fensters mit Werkzeugtabelle. Die **Label**-Objekte werden im Formular (**Form Layout**) positioniert, während die Registerkarte eine vertikale Positionierung (**Vertical Layout**) mit einem Formular und einer Schaltfläche (**Tool Button**) enthält.

- Wir ziehen **Form Layout** aus dem Editorfenster in die Registerkarte **Tool Info**
- In die Registerkarte **Tool Info** ziehen wir **Tool Button** aus dem Editorfenster und platzieren es unter dem Container **Form Layout**

Dies sollte wie folgt aussehen:

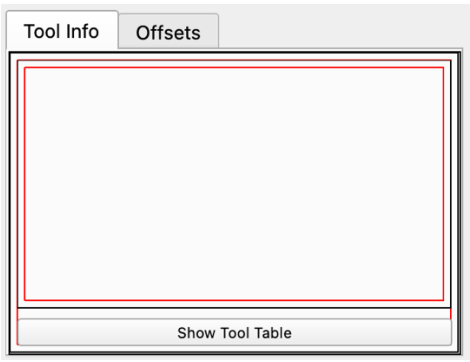


- Wir klicken auf **tabToolInfo** im Objektbaum und setzen die Eigenschaft **layout type** auf **Vertikal**. Damit wird die Positionierung der Objekte auf der Registerkarte definiert.
- Wir setzen den Namen (**id**) des Containers der Registerkarte auf **loutTabToolInfo**

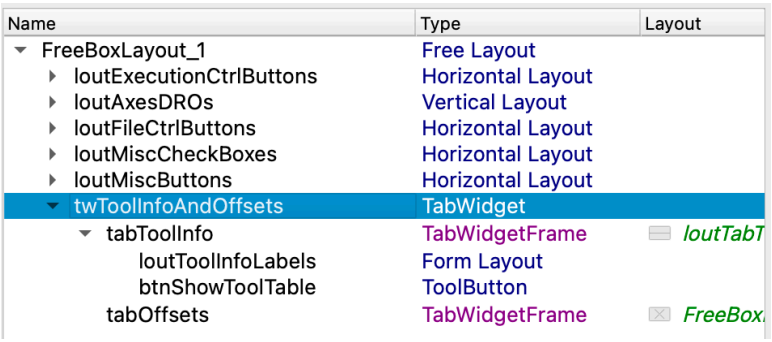


- Wir vergeben Namen (**id**) für den Container **Form Layout** und die Schaltfläche
  - Container: **loutToolInfoLabels**
  - Schaltfläche: **btnShowToolTable**
- Wir setzen die Eigenschaft **horizontal size policy** für die Schaltfläche auf **Preffered**
- Wir setzen die Eigenschaft **Text** für die Schaltfläche auf „Show Tool Table“.

Zu diesem Zeitpunkt sollte das Widget wie folgt aussehen:

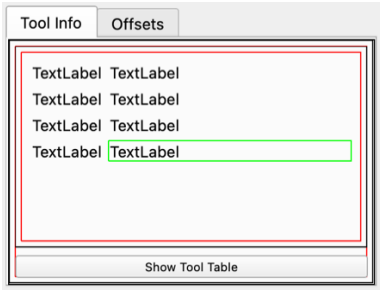


Und so sollte es im Objektbaum aussehen:



Wir können jetzt die **Label**-Objekte in den Container **loutToolInfoLabels** positionieren.

- Wir ziehen acht **Label**-Objekte in den Container **loutToolInfoLabels**. Wir platzieren sie auf eine der Skizze entsprechende Weise, d.h. vier Reihen mit je zwei Widgets.



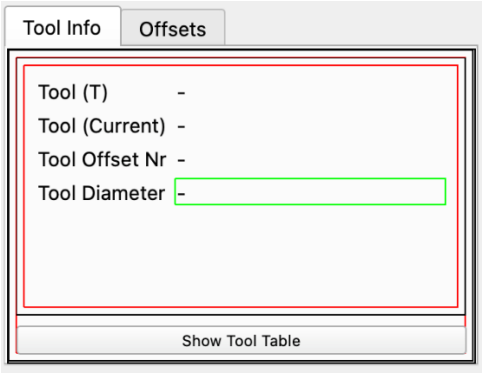
- Wir vergeben den **Label**-Widgets Namen (**id**), wie in der folgenden Tabelle dargestellt

lbSelectedToolDesc	lbSelectedTool
lbCurrentToolDesc	lbCurrentTool
lbToolOffsetNrDesc	lbToolOffsetNr
lbToolDiameterDesc	lbToolDiameter

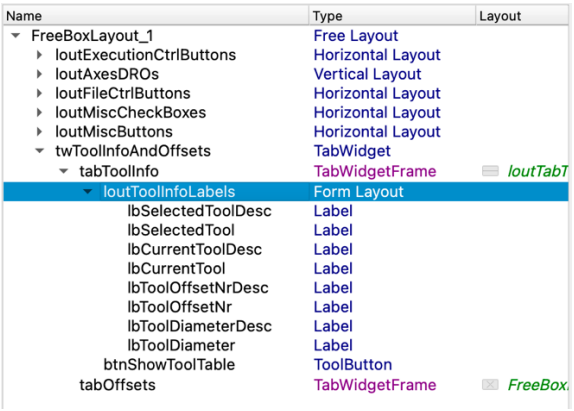
- Wir vergeben den **Label**-Widgets die Eigenschaft **Text**, wie in der folgenden Tabelle dargestellt

Tool (T)	-
Tool (Current)	-
Tool Offset Nr	-
Tool Diameter	-

Das Widget sollte nun wie folgt aussehen:



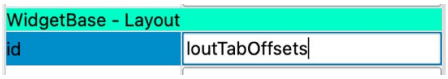
Und so sollte der Objektbaum aussehen:



Es bleibt die Registerkarte „Offsets“ übrig. Hier ist die Sache einfacher, da die Registerkarte nur ein Widget Typ **Offsets Table** enthält.

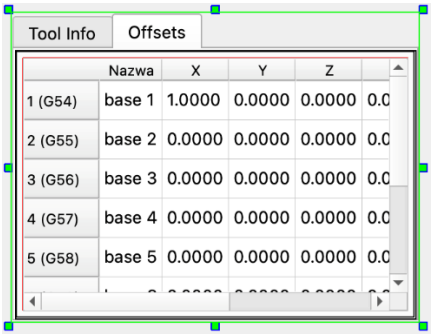
- Wir klicken im Objektbaum auf **tabOffsets**
- Wir setzen die Positionierung der Registerkarte (**layout type**) auf **Vertikal** (die Art der Positionierung hat hier keine große Bedeutung, da nur ein Widget im Container positioniert wird. Irgendeiner muss aber ausgewählt werden, damit sich die Größe des Widgets automatisch an die Containergröße anpasst).

- Wir setzen den Namen (**id)** für den Container der Registerkarte auf **loutTabOffsets**

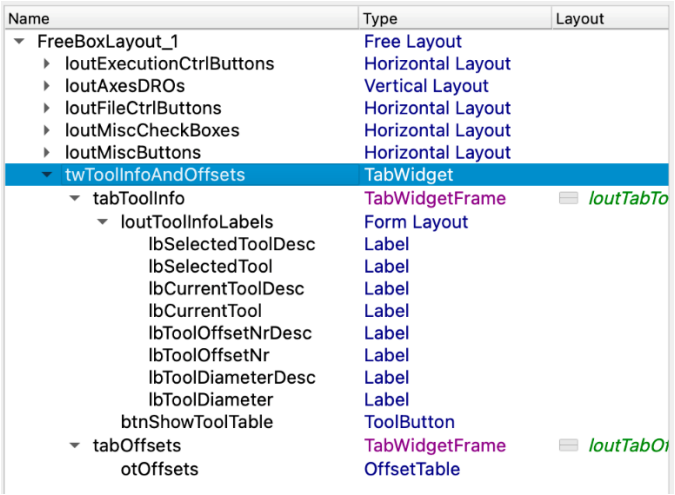


- Wir klicken im simCNC-Fenster auf die Registerkarte „Offsets“, um sie zu aktivieren
- Wir ziehen das Widget **Offset-Table** aus dem Editorfenster in die Registerkarte „Offsets“.
- Wir geben einen Namen (**id**) für das Widget **Offsets Table** in **otOffsets**

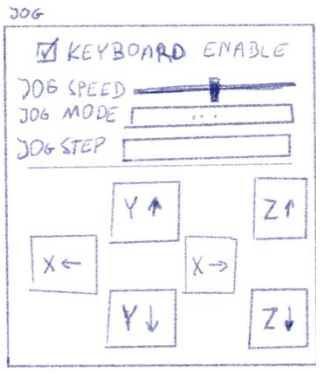
Das Widget ist von der visuellen Seite her fertig und sollte jetzt wie unten dargestellt aussehen:



Ein komplettes Widget sollte im Objektbaum wie folgt aussehen:



„JOG“-Gruppe



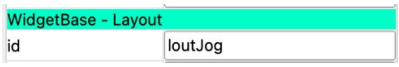
In der Skizze sehen wir von oben:

- Checkbox (**Check Box**), um die Achsenkontrolle über die Tastatur zu ermöglichen
- Etikett (**Label**) und Schieber (**Horizontal Slider**) zur Einstellung der JOG-Bewegungsgeschwindigkeit
- Etikett (**Label**) und Schaltfläche (**Tool Button**) zur Änderung des JOG-Modus
- Etikett (**Label**) und Schaltfläche (**Tool Button**) zur Auswahl des JOG-Schrittes
- Trennlinie (**Frame**)
- Gruppe von Schaltflächen (**Tool Button**), die in einem Raster (**Grid Layout**) positioniert sind, um einzelne Achsen zu kontrollieren

Wir benutzen hier den Widget-Container **Group Box**. Nachstehend sind nacheinander die Planungstätigkeiten dargestellt:

- Wir ziehen das Widget **Group Box** aus dem Editorfenster in das simCNC-Fenster
- Wir legen den Widget-Namen (**id**) auf **gbJog** fest
- Wir setzen die vertikale Positionierung - die Eigenschaft **layout type** auf **Vertikal**

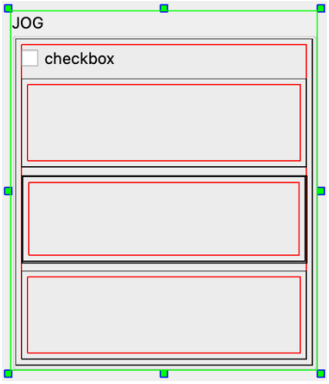
- Wir setzen den Namen (**id**) des Containers auf **loutJog**



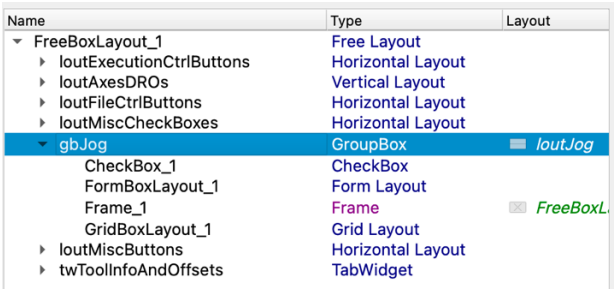
- Wir setzen das Gruppenetikett (Eigenschaft **title**) auf „JOG“.
- Wir ziehen die folgenden Widgets aus dem Editor in die erstellte Gruppe und positionieren sie nacheinander von oben:
  - **Check Box**
  - **Form Layout**
  - **Frame**
  - **Grid Layout**

  Bei der Positionierung von Widgets im Container mit eingeschalteter Autopositionierung ist auf die angezeigten Tags zu achten, die zeigen, wo das neue Objekt platziert wird.

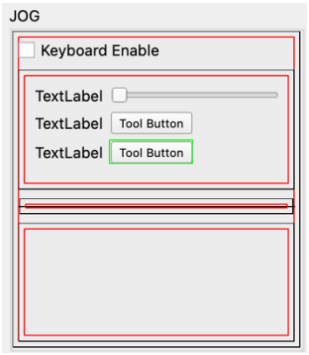
So sollte das Widget jetzt aussehen:



Und so sollte der Objektbaum aussehen:



- Wir setzen die Namen (**id**) der hinzugefügten Objekte nacheinander, von oben: **cbKeyboardJogEnable**, **loutJogConfig**, **frJogLine** und **loutJogButtons**
- Wir setzen die Eigenschaft **Text** des Widgets **cbKeyboardJogEnable** auf „Keyboard Enable“.
- Wir setzen die Eigenschaft **shape** des **frJogLine**-Widgets auf **Horizontal Line**
- Wir setzen **vertical size policy** des **frJogLine**-Widgets auf **Maximum**
- In den Container **loutJogConfig** ziehen wir aus dem Editor drei Widgets **Label** und **Horizontal Slider** und zwei Schaltflächen (**Tool Button**), und positionieren sie wie in der Skizze dargestellt



- Wir vergeben Namen für die neuen Widgets (**id**) gemäß der folgenden Tabelle:

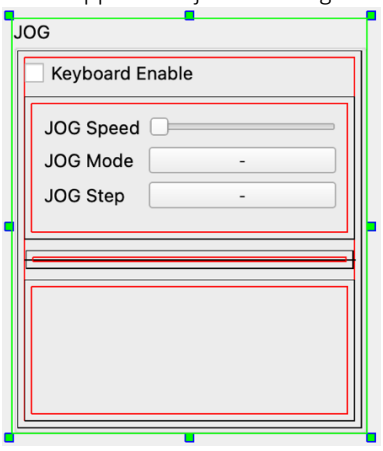
lbJogSpeedDesc	slJogSpeed
lbJogModeDesc	btnJogMode
lbJogStepDesc	btnJogStep

- Wir setzen die Eigenschaft **Text** für die Widgets gemäß der folgenden Tabelle:

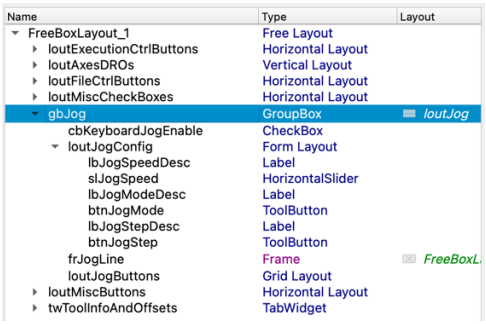
JOG Speed	(nicht anwendbar)
JOG Mode	-
JOG Step	-

- Für die Widgets **btnJogMode** und **btnJogStep** ändern wir die Eigenschaft **horizontal size policy** auf **Preffered**, damit sie sich horizontal an die Containergröße anpassen können.

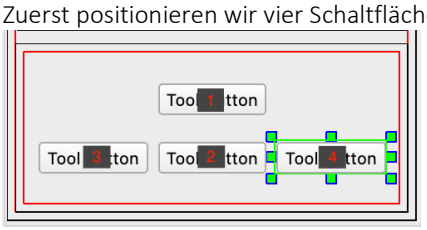
Die Gruppe sollte jetzt wie folgt aussehen:

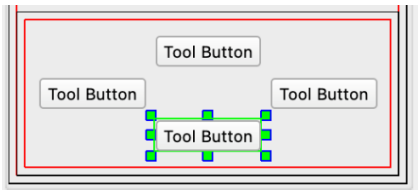


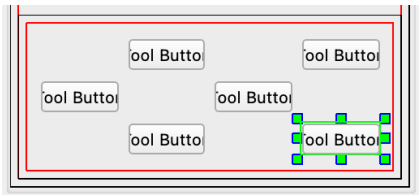
Und so sollte der Objektbaum aussehen:



- Wir fügen sechs Schaltflächen **Tool Button** zu der Gruppe **loutJogButtons** hinzu und positionieren sie wie in der Skizze gezeigt. Es ist erneut auf die Tags zu achten, die anzeigen, wo das hinzugefügte Widget „abgelegt“ wird. Bei dieser Gruppe ist es am einfachsten, in der folgenden Reihenfolge vorzugehen:
  - Zuerst positionieren wir vier Schaltflächen







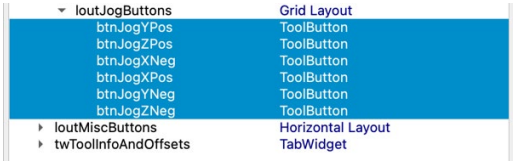
- Wir geben den Schaltflächen Namen (**id**) gemäß der folgenden Tabelle

	btnJogYPos		**btnJogZPos**
btnJogXNeg		**btnJogXPos**	
	btnJogYNeg		**btnJogZNeg**

- Wir setzen die Eigenschaft **Text** für die Widgets gemäß der folgenden Tabelle:
  (Pfeile sind *Unicode*-Zeichen, der Einfachheit halber können sie kopiert und eingefügt werden)

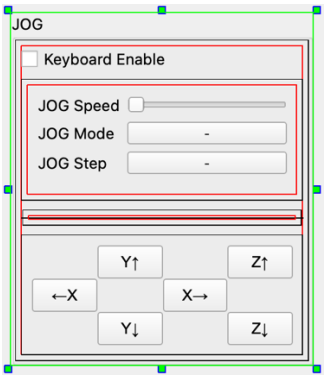
	Y↑		Z↑
←X		X→	
	Y↓		Z↓

- Wir wählen alle **btnJog...**-Schaltflächen im Objektbaum aus und setzen die Eigenschaften **horizontal** und **vertical size policy** auf **Preffered**

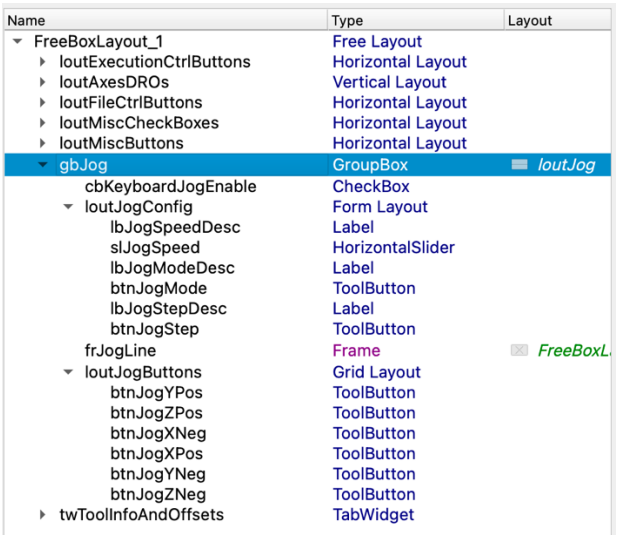


- Wir setzen für die **btnJog**...-Schaltflächen eine etwas größere Schriftart (Eigenschaft **font**) - z.B. „Arial 14“.
- Wir wählen das Objekt **loutJogButtons** und setzen die folgenden Eigenschaften auf „0“:
  - **left, right, top** und **bottom margin**
  - **horizontal** und **vertical spacing**

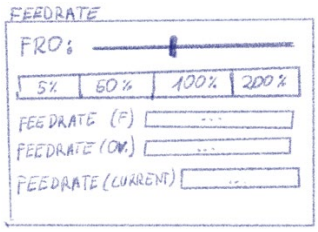
Visuell ist die JOG-Gruppe fertig und sollte wie folgt aussehen:



Und so sollte die JOG-Gruppe im Objektbaum aussehen:



Gruppe „Feedrate“



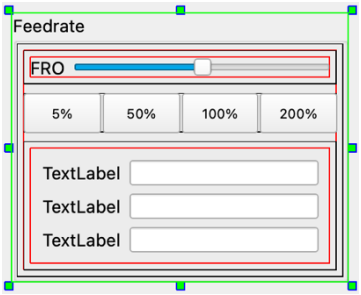
In der obigen Skizze haben wir folgende Elemente (von oben):

- Etikett (**Label**) und Schieber (**Horizontal Slider**) zur Korrektur der Bearbeitungsgeschwindigkeit (FRO)
- Gruppe (**Horizontal Layout**) von Schaltflächen (**Tool Button**) zur schnellen Einstellung häufig verwendeter FRO-Werte
- Etikett (**Label**) und Editierfeld (**Line Edit**) zum Anzeigen und Ändern der Soll-Bearbeitungsgeschwindigkeit
- Etikett (**Label**) und Editierfeld (**Line Edit**) zum Anzeigen der Bearbeitungsgeschwindigkeit mit der verwendeten FRO-Korrektur
- Etikett (**Label**) und Editierfeld (**Line Edit**) zum Anzeigen der aktuellen resultierenden Geschwindigkeit der Maschinenachsen

Wie zuvor werden wir den Widget-Container **Group Box** verwenden.

- Wir ziehen das Widget **Group Box** aus dem Editorfenster in das simCNC-Fenster
- Wir setzen die vertikale Positionierung und ändern die Eigenschaft **layout type** auf **Vertikal**
- Wir geben dem Widget und dem darin befindlichen Container Namen (**id**): **gbFeedrate** und **loutFeedrate**

- Wir ändern das Gruppenetikett (Eigenschaft **title**) auf „Feedrate“.
- Dem Container fügen wir die folgenden Container von oben nach unten hinzu:
  - 2x **Horizontal Layout**
  - **Form Layout**
- Wir geben ihnen Namen (**id**): **loutFroSlider**, **loutFroButtons**, **loutFeedrateInfo**
- Dem Container **loutFroSlider** fügen wir die Widgets **Label** und **Horizontal Slider** hinzu und nennen sie **lbFroDesc** und **slFro**
- Wir ändern die Eigenschaft **Text** des **lbFroDesc**-Widgets auf „FRO“
- Dem Container **loutFroButtons** fügen wir vier Schaltflächen (**Tool Button**) hinzu und geben ihnen Namen (**id**): **btnFro5**, **btnFro50**, **btnFro100**, **btnFro200**
- Wir ändern die Eigenschaft **Text** für die Schaltflächen entsprechend auf „5 %“, „50 %“, „100 %“ bzw. „200 %“.
- Wir ändern die Eigenschaft **horizontal size policy** für die Schaltflächen auf **Preffered**
- Wir ändern die Eigenschaft **vertical size policy** für den Containers „**loutFroSlider**“ auf „**Maximum**“ (dadurch wird der Container nicht vertikal gestreckt)
- Wir ändern die Eigenschaft **vertical size policy** für den Container **loutFroButtons** auf **Maximum**
- Dem Container **loutFeedrateInfo** fügen wir drei **Label**- und drei **Line-Edit**-Widgets hinzu und positionieren sie wie in der Skizze dargestellt: **Label** auf der linken Seite, **Line Edit** auf der rechten Seite



- Wir geben den Widgets Namen (**id**) gemäß der folgenden Tabelle:

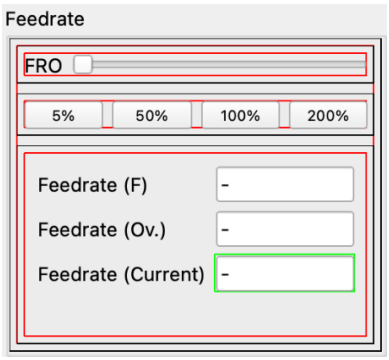
lbFeedrateDesc	edFeedrate
lbFeedrateOvDesc	edFeedrateOv
lbFeedrateCurrentDesc	edFeedrateCurrent

- Wir ändern die Eigenschaft **Text** für die Widgets gemäß der folgenden Tabelle

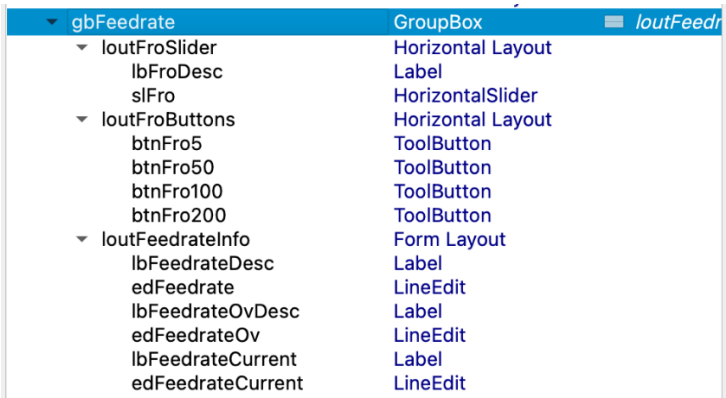
Feedrate (F)	-
Feedrate (Ov.)	-
Feedrate (Current)	-

- Wir setzen die Eigenschaft als schreibgeschützt (**read only**) für die Widgets **edFeedrateOv** und **edFeedrateCurrent**

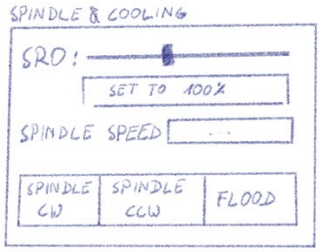
Die „Feedrate“-Gruppe ist visuell fertig und sollte wie folgt aussehen:



Und so sollte sie im Objektbaum aussehen:



Gruppe „Spindle & Cooling“

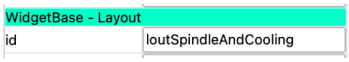


In der obigen Skizze haben wir folgende Elemente (von oben):

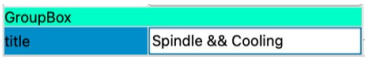
- Etikett (**Label**) und Schieber (**Horizontal Slider**) zur Korrektur der Spindeldrehzahl (SRO)
- Schaltfläche zum Zurücksetzen der Spindeldrehzahlkorrektur (**Tool Button**)
- Etikett (**Label**) und Editierfeld (**Line Edit**) schreibgeschützt und zum Ändern der Soll-Spindeldrehzahl
- Drei Schaltflächen (**Tool Button**) zum Einschalten der Spindel und des Kühlmittels

Wie zuvor, werden wir den Widget-Container **Group Box** verwenden.

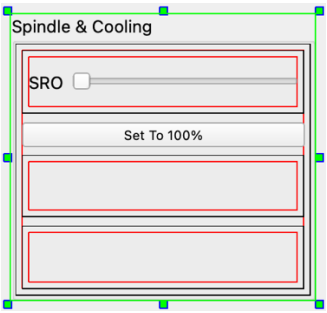
- Wir ziehen das Widget **Group Box** aus dem Editorfenster in das simCNC-Fenster
- Wir setzen die vertikale Positionierung und ändern die Eigenschaft **layout type** auf **Vertikal**
- Wir geben dem Widget und dem darin befindlichen Container Namen (**id**): **gbSpindleAndCooling** und **loutSpindleAndCooling**



- Wir ändern das Etikett der Gruppe (Eigenschaft **title**) auf „Spindle && Cooling“ (das „&“-Zeichen ist ein Sonderzeichen und muss daher doppelt eingegeben werden, damit es korrekt angezeigt wird)



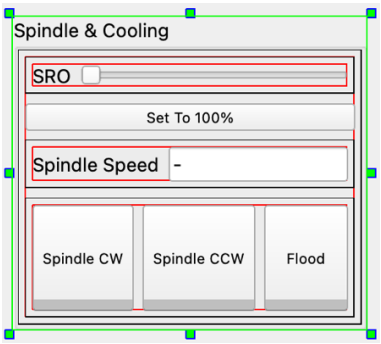
- Dem Container fügen wir die folgenden Elemente von oben nach unten hinzu:
  - **Horizontal Layout**
  - **Tool Button**
  - 2x **Horizontal Layout**
- Wir vergeben die folgenden Namen (**id**): **loutSroSlider**, **btnSroReset**, **loutSpindleInfo**, **loutSpindleAndCoolingCtrl**
- Dem Container **loutSroSlider** fügen wir Label und **Horizontal Slider** hinzu
- Wir geben Namen (**id**): **lbSroDesc** und **slSro**
- Wir setzen die Eigenschaft **Text** des **lbSroDesc**-Widgets auf „SRO“
- Wir setzen die Eigenschaft **Text** des **btnSroReset**-Widgets auf „Set to 100%“.
- Wir setzen die Eigenschaft **horizontal size policy** für das Widget **btnSroReset** auf **Preferred**



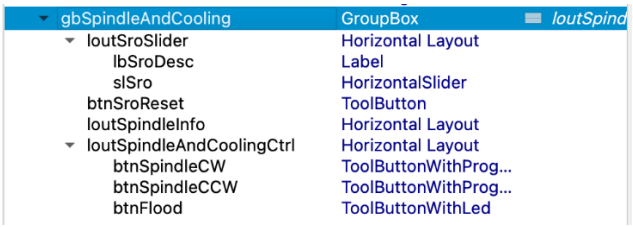
- Dem Container **loutSpindleInfo** fügen wir **Label** und **Line Edit** hinzu
- Wir geben ihnen Namen (**id**): **lbSpindleSpeedDesc** und **edSpindleSpeed**
- Wir setzen die Eigenschaft **Text** für die hinzugefügten Widgets auf „Spindle Speed“ und „-“.

- Dem unteren Container **loutSpindleAndCoolingCtrl** fügen wir von links nacheinander die folgenden Elemente hinzu:
  - 2x **Tool Button with Progress Bar**
  - **Tool Button with LED**
- Den hinzugefügten Schaltflächen geben wir die folgenden Namen (**id**): **btnSpindleCW**, **btnSpindleCCW** und **btnFlood**
- Für die Schaltflächen setzen wir die Eigenschaft **text**: „Spindle CW“, „Spindle CCW“und „Flood“
- Für die Schaltflächen ändern wir die Eigenschaften **horizontal** und **vertical size policy** auf **Preffered**.
- Wir markieren die Eigenschaft **LED visible** für die Schaltfläche **btnFlood**
- Wir ändern die Eigenschaft **vertical size policy** für die Container **loutSroSlider** und **loutSpindleInfo** auf **Maximum**

Die „Spindle & Cooling“-Gruppe ist visuell fertig und sollte wie folgt aussehen:



Und so sollte sie im Objektbaum aussehen:

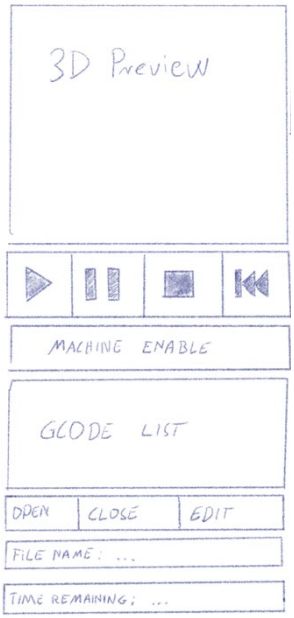


Hauptgruppen und Positionierung

Nachdem alle erforderlichen grundlegenden Widget-Gruppen erstellt worden sind, können wir mit der Planung der Hauptgruppen beginnen. In der Skizze vom Anfang des Abschnitts kann man bemerken, dass der Entwurf aus drei Widgets-Spalten besteht, wobei die Elemente in jeder Spalte vertikal angeordnet sind.

Linke Spalte

In der Skizze sehen wir die linke Hauptgruppe der Widgets. Sie enthält nacheinander (von oben):



- 3D-Ansicht des Pfades (**Path View**)
- Schaltflächengruppe **loutExecutionCtrlButtons**
- Schaltfläche zum Versetzen der Maschine in den Bereitschaftsmodus (**Tool Button with LED**)
- Dateiansicht in Textform – G-Code-Liste (**GCode List**)
- Schaltflächengruppe **loutFileCtrlButtons**
- Gruppe Dateiname und prognostizierte Bearbeitungszeit - **loutFileInfo**

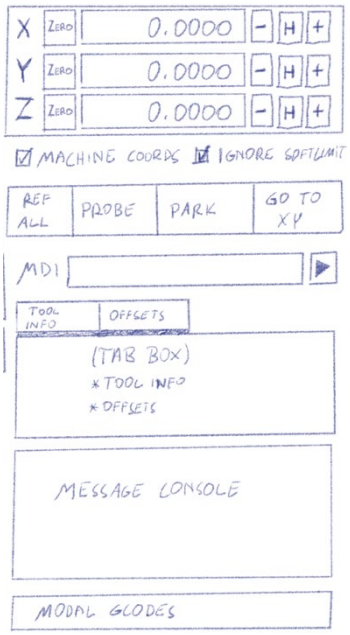
Liste der Tätigkeiten bei der Planung:

- Wir ziehen den Container der vertikalen Positionierung **Vertical Layout** aus dem Editorfenster in das simCNC-Fenster
- Wir geben den Namen (**id**) für den Container: **loutLeftColumn**
- Auf den Container ziehen wir die folgenden Elemente und positionieren sie von oben nach unten:
    - **PathView**
    - Die zuvor entworfene Gruppe **loutExecutionCtrlButtons**
    - **Tool Button with LED**
    - **GCode List**
    - Die zuvor entworfene Gruppe **loutFileCtrlButtons**
    - Die zuvor entworfene Gruppe **loutFileInfo**
- Wir geben den Widgets **PathView**, **ToolButton with LED** und **GCode List** die folgenden Namen (**id**): **view3D**, **btnCtrlEnable** und **gcodeList**
- Wir definieren die Eigenschaft **LED visible** für die Schaltfläche **btnCtrlEnable**
- Wir setzen die Eigenschaften **horizontal** und **vertical size policy** für die Schaltfläche **btnCtrlEnable** auf **Preffered**.
- Wir setzen die Eigenschaft **Text** derselben Schaltfläche auf „Machine Enable“

Die linke Spalte der Widgets ist fertig und sollte wie folgt aussehen:

loutLeftColumn	Vertical Layout
view3D	SimGLWidget
loutExecutionCtrlButtons	Horizontal Layout
btnStart	ToolButton
btnPause	ToolButton
btnStop	ToolButton
btnRewind	ToolButton
btnCtrlEnable	ToolButtonWithLed
gcodeList	GCodeList
loutFileCtrlButtons	Horizontal Layout
btnOpen	OpenFileButton
btnClose	ToolButton
btnEdit	ToolButton
loutFileInfo	Form Layout
lbFileNameDesc	Label
lbFileName	Label
lbTimeRemainingDesc	Label
lbTimeRemaining	Label

Zentrale Spalte



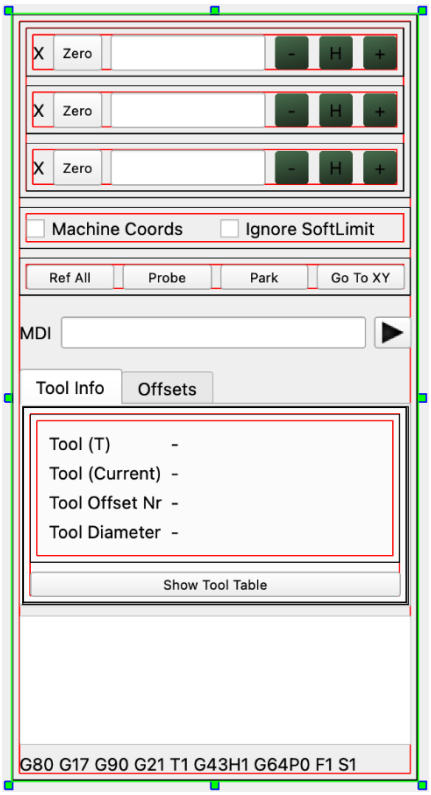
In der Skizze sehen wir die zentrale Hauptgruppe der Widgets. Sie enthält nacheinander (von oben):

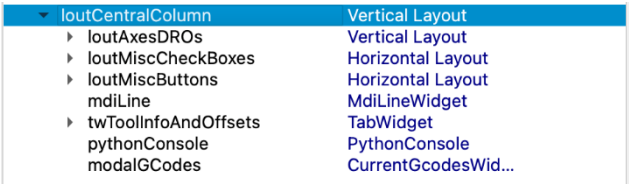
- Achsenzeigergruppe **loutAxesDROs**
- Checkbox-Liste **loutMiscCheckBoxes**
- Schaltflächengruppe **loutMiscButtons**
- Widget für die manuelle Maschinenbefehlseingabe **MDI line**
- Widget mit Registerkarten **twToolInfoAndOffsets**
- Konsole für Meldungen **Python Console**
- Widget der modalen G-Code-Liste **CurrentGCodesWidget**

Liste der Tätigkeiten bei der Planung:

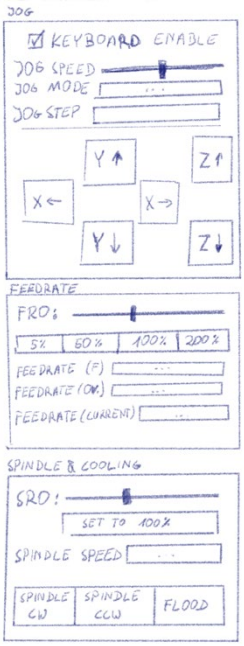
- Wir ziehen den Container der vertikalen Positionierung **Vertical Layout** aus dem Editorfenster in das simCNC-Fenster
- Wir geben den Namen (**id**) für den Container: **loutCentralColumn**
- Auf den Container ziehen wir die folgenden Elemente und positionieren sie von oben nach unten:
  - Die zuvor entworfene Gruppe **loutAxesDROs**
  - Die zuvor entworfene Gruppe **loutMiscCheckBoxes**
  - Die zuvor entworfene Gruppe **loutMiscButtons**
  - **MDI line**
  - Das zuvor entworfene Widget **twToolInfoAndOffsets**
  - **Python Console**
  - **Current g-codes**
- Wir geben Namen (**id**) für Widgets **MDI-Line**, **Python Console** und **Current g-codes** entsprechend auf: **mdiLine**, **pythonConsole**, **modalGCodes**

Die zentrale Hauptgruppe der Widgets ist fertig und sollte wie folgt aussehen:





Rechte Spalte



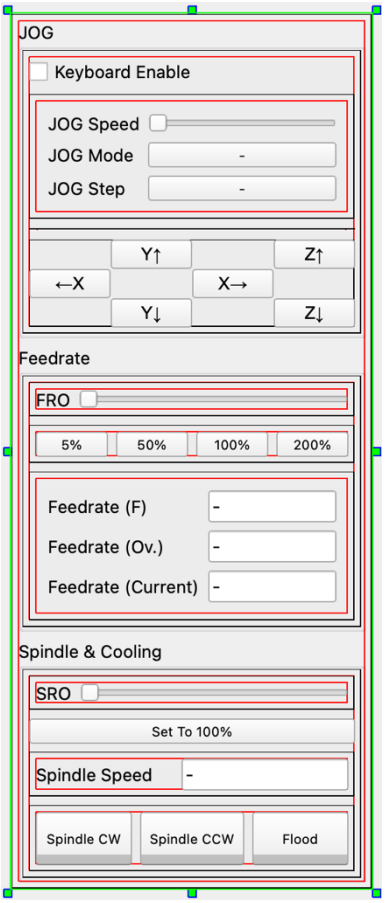
In der Skizze sehen wir die rechte Hauptgruppe der Widgets. Sie enthält nacheinander (von oben):

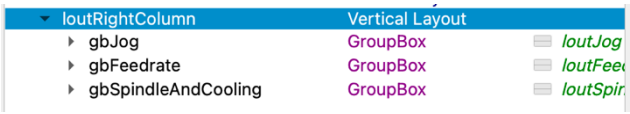
- Gruppe der manuellen Steuerung **gbJog**
- Gruppe der Vorschubkontrolle **gbFeedrate**
- Gruppe der Spindel- und Kühlmittelkontrolle **gbSpindleAndCooling**

Liste der Tätigkeiten bei der Planung:

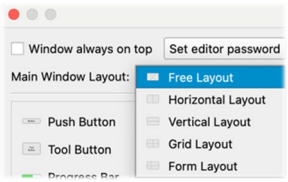
- Wir ziehen den Container der vertikalen Positionierung **Vertical Layout** aus dem Editorfenster in das simCNC-Fenster
- Wir geben die Eigenschaft **id** für den Container: **loutRightColumn**
- Dem Container fügen wir die folgenden Elemente von oben nach unten hinzu:
  - Die zuvor entworfene Gruppe **gbJog**
  - Die zuvor entworfene Gruppe **gbFeedrate**
  - Die zuvor entworfene Gruppe **gbSpindleAndCooling**

Die rechte Hauptgruppe ist fertig und sollte wie folgt aussehen:





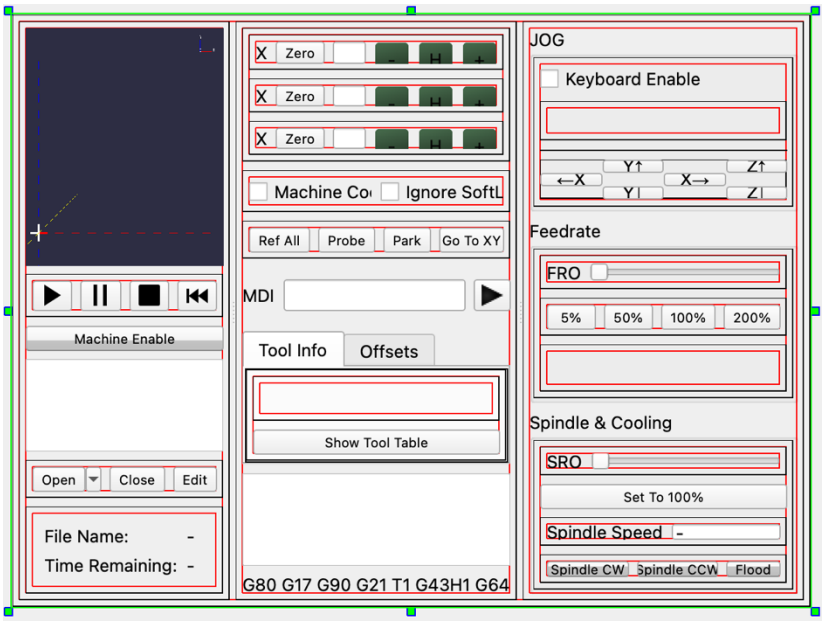
Positionierung im Hauptcontainer



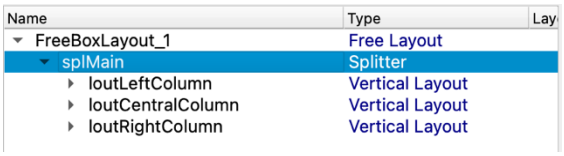
Nachdem wir die drei Hauptgruppen unserer Schnittstelle vorbereitet haben, sollte jetzt die Positionierung im Hauptfenster ausgewählt werden. Wie wir in der Abbildung rechts sehen können, haben wir für den Hauptcontainer die Wahl zwischen vier Arten der Positionierung: horizontal, vertikal, im Raster und im Formular. In diesem Fall werden wir jedoch **Splitter** verwenden, damit der Bediener die Aufteilung des Platzes zwischen den drei Spalten der Schnittstelle problemlos ändern kann. **Splitter** ist in den verfügbaren Optionen nicht vorhanden, aber man kann ihn sehr einfach umgehen.

Führen wir die folgenden Tätigkeiten aus:

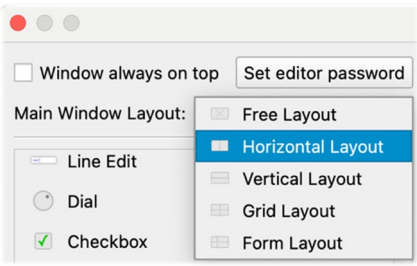
- Wir ziehen ein **Splitter**-Objekt aus dem Editorfenster in das simCNC-Fenster
- Wir geben ihm den Namen (**id**): splMain
- In den **splMain**-Container ziehen wir die Gruppen **loutLeftColumn**, **loutCentralColumn** und **loutRightColumn** nacheinander. Lassen wir sie am rechten Rand fallen, um die gewünschte Reihenfolge der Spalten aufrechtzuerhalten.



- Prüfen wir, ob im Objektbaum die richtige Hierarchie beibehalten wird.



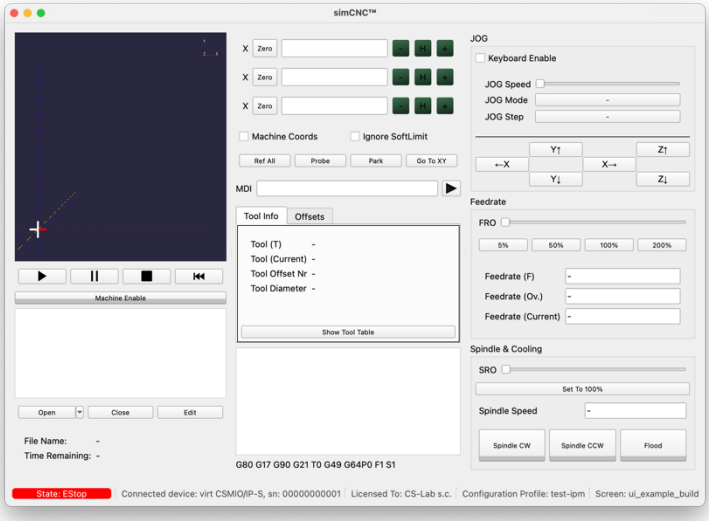
- Wir setzen die Positionierung **Horizontal Layout** für das Hauptfenster:



Da wir nach dem Hinzufügen von Splitter nur eine Gruppe (**splMain**) im Hauptfenster haben, hat es keine Bedeutung, ob wir die horizontale oder vertikale Positionierung auswählen. Die **splMain**-Gruppe wird sowieso einfach an die Fenstergröße angepasst.

- Im Objektbaum wählen wir das Objekt Typ **Horizontal Layout** ganz oben aus und geben ihm den Namen (**id**): **loutMain**

Wir können jetzt das Editorfenster für einen Moment schließen (nicht vergessen, die Änderungen beizubehalten) und uns die Effekte der bisherigen Arbeit anschauen. Das Programmfenster sollte wie folgt aussehen:



Wie man sieht, stimmt das Aussehen des Entwurfs mit unserem Konzept überein. Er kann visuell etwas attraktiver gestaltet werden, aber dazu kommen wir später. Jetzt werden wir uns mit der funktionalen Seite befassen, d.h. man sollte verursachen, dass die Schnittstelle ihre grundlegende Aufgabe erfüllt, nämlich die Darstellung von Informationen und die Annahme von Befehlen des Bedieners.

Zuweisung von Funktionen / Aktionen

Die meisten der Schnittstellenobjekte sind in dieser Phase inaktiv. Damit die Widgets ihre Aufgaben erfüllen können, müssen wir ihre Eingangs- und Ausgangsfunktionen definieren. Wir öffnen den Schnittstelleneditor erneut (Menü **Konfiguration→Schnittstelleneditor öffnen**). Nachstehend befindet sich eine Tabelle der Projekt-Widgets samt den Eigenschaften der Ein- und Ausgänge, die eingestellt werden sollen.
Zum Beispiel für die Schaltfläche Start der G-Code-Ausführung (Schaltfläche mit dem „Play"-Symbol in der linken Spalte unter der 3D-Ansicht) - **btnStart** - setzen wir die Eigenschaft **Output: clicked** auf **Start trajectory**).

Widget-„Id"	Typ der Eigenschaft	Wert	Beschreibung
btnStart	Output: clicked	Start trajectory	Start von G-Code nach Anklicken der Schaltfläche
btnPause	Output: clicked	Set Pause On/Off	Ein-/Ausschalten der Pause der G-Code-Ausführung nach Anklicken der Schaltfläche
btnStop	Output: clicked	Stop trajectory	Stoppen der G-Code-Ausführung nach Anklicken der Schaltfläche
btnRewind	Output: clicked	Rewind trajectory	Scrollen des G-Codes zum Anfang nach Anklicken der Schaltfläche
btnCtrlEnable	Output: clicked	Switch to EStop/idle state	Umschalten zwischen EStop-Status und Bereitschaft nach Anklicken der Schaltfläche
btnCtrlEnable	Input: LED state	CSMIO enable	Der LED-Status auf der Schaltfläche ändert sich mit dem Bereitschaftsstatus des CSMIO-Controllers.
btnOpen	-	-	Es wurde ein Widget zum Öffnen von G-Code-Dateien verwendet, das keine Konfiguration erfordert
btnClose	Output: clicked	Run script	In der Aktionsliste ist das Schließen der Datei nicht verfügbar, aber dies kann von dem Python-Makro-Niveau aus gemacht werden. Eine Liste der Makros für Widgets ist dem folgenden Unterabschnitt zu entnehmen.
btnEdit	Output: clicked	Edit G-Code	Öffnen des G-Code-Datei-Editors
lbFileName	Input: text	GCode file path	Anzeige des Pfades der geladenen G-Code-Datei
lbTimeRemaining	Input: text	Remain path time	Anzeige der verbleibenden Bearbeitungszeit
btnAxisXZero	Output: clicked	Run Script	Nullung der Programmkoordinate für die X-Achse von dem Python-Makro-Niveau aus. Eine Liste der Makros für Widgets ist dem folgenden Unterabschnitt zu entnehmen.
btnAxisYZero	Output: clicked	Run Script	Wie oben, aber für die Y-Achse
btnAxisZZero	Output: clicked	Run Script	Wie oben, aber für die Z-Achse
edAxisXPosition	Input: text	Axis X display position	Anzeige der (Programm-/Maschinen-)Koordinate für die X-Achse
edAxisXPosition	Output: returnPressed	Set axis X prog position	Änderung der Programm-Koordinate für die X-Achse nach dem Drücken der „return"-Schaltfläche
edAxisYPosition	Input: text	Axis Y display position	Anzeige der (Programm-/Maschinen-) Koordinate für die Y-Achse
edAxisYPosition	Output: returnPressed	Set axis Y prog position	Änderung der Programm-Koordinate für die Y-Achse nach dem Drücken der „return"-Schaltfläche
edAxisZPosition	Input: text	Axis Z display position	Anzeige der (Programm-/Maschinen-) Koordinate für die Z-Achse
edAxisZPosition	Output: returnPressed	Set axis Z prog position	Änderung der Programm-Koordinate für die Z-Achse nach dem Drücken der „return"-Schaltfläche
ioAxisXLimitNeg	Input: state	Signal value (IP;0;mkit;0→limit--;0)	Status der Kontrollleuchte verbunden mit dem Status des „limit--"-Signals von MotionKit Nr. 0 CSMIO/IP
ioAxisXHoming	Input: state	Signal value (IP;0;mkit;0→Home;0)	Status der Kontrollleuchte verbunden mit dem Status des „home--"-Signals von MotionKit Nr. 0 CSMIO/IP
ioAxisXLimitPos	Input: state	Signal value (IP;0;mkit;0→limit++;0)	Status der Kontrollleuchte verbunden mit dem Status des „limit++"-Signals von MotionKit Nr. 0 CSMIO/IP
ioAxisYLimitNeg	Input: state	Signal value (IP;0;mkit;1→limit--;0)	Status der Kontrollleuchte verbunden mit dem Status des „limit--"-Signals von MotionKit Nr. 1 CSMIO/IP
ioAxisYHoming	Input: state	Signal value (IP;0;mkit;1→Home;0)	Status der Kontrollleuchte verbunden mit dem Status des „home--"-Signals von MotionKit Nr. 1 CSMIO/IP
ioAxisYLimitPos	Input: state	Signal value (IP;0;mkit;1→limit++;0)	Status der Kontrollleuchte verbunden mit dem Status des „limit++"-Signals von MotionKit Nr. 1 CSMIO/IP

Widget-„Id“	Typ der Eigenschaft	Wert	Beschreibung
ioAxisZLimitNeg	Input: state	Signal value (IP;0;mkit;2→limit--;0)	Status der Kontrollleuchte verbunden mit dem Status des „limit--“-Signals von MotionKit Nr. 2 CSMIO/IP
ioAxisZHoming	Input: state	Signal value (IP;0;mkit;2→Home;0)	Status der Kontrollleuchte verbunden mit dem Status des „home--“-Signals von MotionKit Nr. 2 CSMIO/IP
ioAxisZLimitPos	Input: state	Signal value (IP;0;mkit;2→limit++;0)	Status der Kontrollleuchte verbunden mit dem Status des „limit++“-Signals von MotionKit Nr. 2 CSMIO/IP
cbMachineCoords	Input: checkbox state	Ref position displayed	Staus der Checkbox verbunden damit, ob die Maschinenkoordinaten aktuell angezeigt werden
cbMachineCoords	Output: stateChanged	Set position display to ref	Umschalten auf die Anzeige der Maschinenkoordinaten nach der Auswahl der Checkbox
cbIgnoreSoftLimit	Input: checkbox state	Global soft limit disabled	Status der Checkbox verbunden mit dem Status der Deaktivierung von Programmlimits
cbIgnoreSoftLimit	Output: stateChanged	Disable global soft limit	Ausschalten von Programmlimits nach der Auswahl der Checkbox
btnRefAll	Output: clicked	Ref all axes	Aktivierung der Referenzierung aller Achsen nach Anklicken der Schaltfläche
btnProbe	Output: clicked	Execute probing script	Aktivierung des Standard-Python-Makros für die Werkzeugmessung nach Anklicken der Schaltfläche
btnPark	Output: clicked	Run script	Aktivierung des Python-Makros für die Fahrt zur Parkposition. Eine Liste der Makros für Widgets ist dem folgenden Unterabschnitt zu entnehmen.
btnGoToXY	Output: clicked	Run script	Aktivierung des Python-Makros für die Fahrt zur Arbeitsoffset-Position. Eine Liste der Makros für Widgets ist dem folgenden Unterabschnitt zu entnehmen.
lbSelectedTool	Input: text	Selected tool number	Anzeige der Nummer des aktuell ausgewählten Werkzeugs
lbCurrentTool	Input: text	Spindle tool number	Anzeige der Nummer des Werkzeugs, das sich aktuell in der Spindel befindet
lbToolOffsetNr	Input: text	Tool offset number	Anzeige der ausgewählten Werkzeugoffset-Nummer
lbToolDiameter	Input: text	Tool diameter	Anzeige des Durchmessers des aktuellen Werkzeugs
btnShowToolTable	Output: clicked	Tool Table	Anzeige der Liste der Werkzeuge nach Anklicken der Schaltfläche
cbKeyboardJogEnable	Input: checkbox state	Key Control	Status der Checkbox verbunden mit dem Aktivierungsstatus der Maschinenkontrolle von der Tastatur aus
cbKeyboardJogEnable	Output: state changed	Key Control	Zustimmung für die Maschinenkontrolle von der Tastatur aus, wenn die Checkbox markiert ist
slJogSpeed	Input: value	Jog speed	Schieberposition bezogen auf die JOG-Geschwindigkeit
slJogSpeed	Output: valueChanged	Set Jog Speed	Die Änderung der Schieberposition führt zur Änderung der aktuellen JOG-Geschwindigkeit
btnJogMode	Input: text	Jog mode	Der aktuelle JOG-Modus bestimmt den Text auf der Schaltfläche
btnJogMode	Output: clicked	Set Jog mode	Durch Klicken auf die Schaltfläche wird der aktuelle Jog-Modus geändert
btnJogStep	Input: text	Jog step	Der aktuelle JOG-Schritt bestimmt den Text auf der Schaltfläche
btnJogStep	Output: clicked	Run Script	Aktivierung des Python-Makros, das zwischen den Werten zyklisch umschalten wird. Eine Liste der Makros für Widgets ist dem folgenden Unterabschnitt zu entnehmen.
btnJogXNeg	Output: pressed	Jog X- pressed	Wenn die Schaltfläche gedrückt wird, wird eine JOG-Fahrt der X-Achse in die negative Richtung ausgeführt
btnJogXNeg	Output: released	JOG X- released	Wenn die Schaltfläche losgelassen wird, wird die JOG-Fahrt der X-Achse in die negative Richtung gestoppt
btnJogXPos	Output: pressed	Jog X+ pressed	Wenn die Schaltfläche gedrückt wird, wird eine JOG-Fahrt der X-Achse in die positive Richtung ausgeführt
btnJogXPos	Output: released	JOG X+ released	Wenn die Schaltfläche losgelassen wird, wird die JOG-Fahrt der X-Achse in die positive Richtung gestoppt
btnJogYNeg	Output: pressed	Jog Y- pressed	Wenn die Schaltfläche gedrückt wird, wird eine JOG-Fahrt der Y-Achse in die negative Richtung ausgeführt
btnJogYNeg	Output: released	JOG Y- released	Wenn die Schaltfläche losgelassen wird, wird die JOG-Fahrt der Y-Achse in die negative Richtung gestoppt
btnJogYPos	Output: pressed	Jog Y+ pressed	Wenn die Schaltfläche gedrückt wird, wird eine JOG-Fahrt der Y-Achse in die positive Richtung ausgeführt

Widget-„Id“	Typ der Eigenschaft	Wert	Beschreibung
btnJogYPos	Output: released	JOG Y+ released	Wenn die Schaltfläche losgelassen wird, wird die JOG-Fahrt der Y-Achse in die positive Richtung gestoppt
btnJogZNeg	Output: pressed	Jog Z- pressed	Wenn die Schaltfläche gedrückt wird, wird eine JOG-Fahrt der Z-Achse in die negative Richtung ausgeführt
btnJogZNeg	Output: released	JOG Z- released	Wenn die Schaltfläche losgelassen wird, wird die JOG-Fahrt der Z-Achse in die negative Richtung gestoppt
btnJogZPos	Output: pressed	Jog Z+ pressed	Wenn die Schaltfläche gedrückt wird, wird eine JOG-Fahrt der Z-Achse in die positive Richtung ausgeführt
btnJogZPos	Output: released	JOG Z+ released	Wenn die Schaltfläche losgelassen wird, wird die JOG-Fahrt der Z-Achse in die positive Richtung gestoppt
slFro	Input: value	Fro	Die Position des Schiebers wird zusammen mit der Änderung des FRO-Wertes angepasst.
slFro	Output: valueChanged	Set Fro	Die Änderung der Schieberposition führt zur Änderung des aktuellen FRO-Wertes
btnFro5	Output: clicked	Run Script	Aktivierung des Python-Makros, das den FRO-Wert auf 5 % setzt. Eine Liste der Makros für Widgets ist dem folgenden Unterabschnitt zu entnehmen.
btnFro50	Output: clicked	Run Script	Aktivierung des Python-Makros, das den FRO-Wert auf 50 % setzt. Eine Liste der Makros für Widgets ist dem folgenden Unterabschnitt zu entnehmen.
btnFro100	Output: clicked	Run Script	Aktivierung des Python-Makros, das den FRO-Wert auf 100 % setzt. Eine Liste der Makros für Widgets ist dem folgenden Unterabschnitt zu entnehmen.
btnFro200	Output: clicked	Run Script	Aktivierung des Python-Makros, das den FRO-Wert auf 200 % setzt. Eine Liste der Makros für Widgets ist dem folgenden Unterabschnitt zu entnehmen.
edFeedrate	Input: text	Feedrate	Anzeige der aktuell eingestellten Bearbeitungsgeschwindigkeit
edFeedrate	Output: returnPressed	Set Feedrate	Einstellen der Bearbeitungsgeschwindigkeit, wenn die return-Schaltfläche gedrückt wird.
edFeedrateOv	Input: text	Feedrate override	Anzeige der aktuell eingestellten Bearbeitungsgeschwindigkeit, einschließlich FRO
edFeedrateCurrent	Input: text	Current velocity	Anzeige der aktuellen resultierenden Geschwindigkeit – Geschwindigkeit des Werkzeugs in Bezug auf das Material.
slSro	Input: text	Sro	Die Position des Schiebers wird zusammen mit der SRO-Änderung angepasst.
slSro	Output: valueChanged	Set Sro	Die Änderung der Schieberposition führt zur Änderung des aktuellen SRO-Wertes
btnSroReset	Output: clicked	Run Script	Aktivierung des Python-Makros, das den SRO-Wert auf 100 % setzt. Eine Liste der Makros für Widgets ist dem folgenden Unterabschnitt zu entnehmen.
edSpindleSpeed	Input: text	Spindle Speed	Anzeige der aktuell eingestellten Spindeldrehzahl
edSpindleSpeed	Output: returnPressed	Set Spindle Speed	Einstellen der Spindeldrehzahl, wenn die return-Schaltfläche gedrückt wird
btnSpindleCW	Output: clicked	Run Spindle Clockwise	Einschalten des Rechtslaufs der Spindel nach dem Drücken der Schaltfläche
btnSpindleCW	Input: value	Spindle CW percent	Der Balken auf der Schaltfläche zeigt an, dass die eingestellte Drehzahl erreicht wurde (Rechtslauf)
btnSpindleCCW	Output: clicked	Run Spindle Counter-Clockwise	Einschalten des Linkslaufs der Spindel nach dem Drücken der Schaltfläche
btnSpindleCCW	Input: value	Spindle CCW percent	Der Balken auf der Schaltfläche zeigt an, dass die eingestellte Drehzahl erreicht wurde (Linkslauf)
btnFlood	Output: clicked	Set Flood On/Off	Ein-/Ausschalten des Kühlmittels
btnFlood	Input: LED state	Flood on	Die Diode auf der Schaltfläche leuchtet, wenn das Kühlmittel eingeschaltet ist.

Python-Makros für Widgets mit „Run script“-Aktion

Wie man in der obigen Tabelle sehen kann, ist einigen Widgets keine spezifische Funktion zugewiesen, sondern lediglich ein **Run script**, also Aufrufen des Python-Makros. Nachfolgend finden wir die Dateinamen zusammen mit ihren Quellcodes. Um die Dateien zu erstellen, können wir den in simCNC integrierten Editor (Menü **Makros→Makro-Editor anzeigen**) oder einen beliebigen Editor wie z.B. **VS Code** verwenden. Die Dateien sind vorzugsweise im Verzeichnis des entworfenen Bildschirms, im Unterverzeichnis scripts, zu speichern. Die Dateien müssen die Endung „.py“ haben.
Unter Windows wird dies der folgende Pfad sein: *„C:\Program Files\simCNC\screens\ui_example\scripts”*
Für Linux: *“/opt/simCNC/screens/ui_example/scripts”*
Für macOS: *“/Applications/CS-Lab/simCNC.app/Contents/MacOS/screens/ui_example/scripts/”*

*Makro für Schaltfläche **btnClose** - Datei „btnClose.py“*

```
d.closeGCodeFile( )
print(„GCode file closed.”)
```

*Makro für Schaltfläche **btnAxisXZero** - Datei „btnAxisXZero.py“*

```
d.setAxisProgPosition( Axis.X, 0 )
print("Axis X prog position set to 0.000")
```

*Makro für Schaltfläche **btnAxisYZero** - Datei „btnAxisYZero.py“*

```
d.setAxisProgPosition( Axis.Y, 0 )
print("Axis Y prog position set to 0.000")
```

*Makro für Schaltfläche **btnAxisZZero** - Datei „btnAxisZZero.py“*

```
d.setAxisProgPosition( Axis.Z, 0 )
print("Axis Z prog position set to 0.000")
```

*Makro für Schaltfläche **btnPark** - Datei „btnPark.py“*

```
d.executeGCode( "G0G53 Z0" );
d.executeGCode( "G0G53 X0 Y0" );
print("Go to park position finished")
```

*Makro für Schaltfläche **btnGoToXY** - Datei „btnGoToXY.py“*

```
d.executeGCode( "G0G53 Z0" );
d.executeGCode( "G0 X0 Y0" );
print("Go to material zero XY position finished")
```

*Makro für Schaltfläche **btnJogStep** - Datei „btnJogStep.py“*

```
currentStep = d.getJogStep( )
newStep = currentStep * 10.0
if newStep > 1.0:
        newStep = 0.001
d.setJogStep( newStep )
print("JOG step set to: {:.3f}".format(newStep))
```

*Makro für Schaltfläche **btnFro5** - Datei „btnFro5.py“*

```
d.setFRO( 5 )
print("FRO set to: {:.1f}%".format(d.getFRO( )))
```

*Makro für Schaltfläche **btnFro50** - Datei „btnFro50.py“*

```
d.setFRO( 50 )
print("FRO set to: {:.1f}%".format(d.getFRO( )))
```

*Makro für Schaltfläche **btnFro100** - Datei „btnFro100.py“*

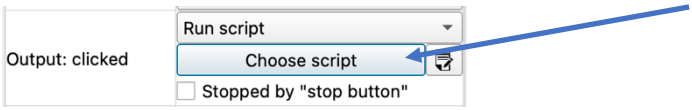
```
d.setFRO( 100 )
print("FRO set to: {:.1f}%".format(d.getFRO( )))
```

*Makro für Schaltfläche **btnFro200** - Datei „btnFro200.py“*

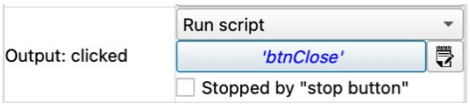
```
d.setFRO( 200 )
print("FRO set to: {:.1f}%".format(d.getFRO( )))
```

Zuweisung von Makros zu Widgets

Um einem Widget ein Makro zuzuweisen, ist ein Widget auszuwählen und in der Eigenschaftsliste unter der Aktion **Run script** auf die Schaltfläche Makroauswahl zu klicken.



So sollte es für die Schaltfläche **btnClose** aussehen, nachdem das Makro „btnClose.py“ ausgewählt wurde



Auf diese Weise weisen wir den Widgets alle oben angegebenen Makros zu.

Ergänzungen und kleinere Korrekturen

In dieser Phase haben wir bereits eine Schnittstelle, die funktioniert und verwendet werden kann. Bevor wir uns mit den visuellen Verbesserungen und der Kosmetik befassen, werden wir einige kleinere Korrekturen vornehmen.

2. Einstellen des Wertebereichs für die Schieber **slJogSpeed**, **slFro** und **slSro**
    a. Wir setzen die Eigenschaft **Maximum** des **slJogSpeed**-Widgets auf 100
    b. Wir setzen die Eigenschaft **Maximum** des **slFro**-Widgets auf 200
    c. Wir setzen die Eigenschaft **Maximum** des **slSro**-Widgets auf 200
3. Einstellen des Wertebereichs für die Visualisierungsbalken der Spindeldrehzahl auf den Schaltflächen **btnSpindleCW** und **btnSpindleCCW**
    a. Wir setzen die Eigenschaft **Maximum** des **btnSpindleCW**-Widgets auf 1
    b. Wir setzen die Eigenschaft **Maximum** des **btnSpindleCCW**-Widgets auf 1
4. Einstellen des Koordinatenanzeigeformats auf X.XXX
    a. Wir geben in das Eigenschaftsfeld **display format** des Widgets **edAxisXPosition**, **edAxisYPosition** und **edAxisZPosition** den Wert „%.3f“ ein (ohne Anführungszeichen)

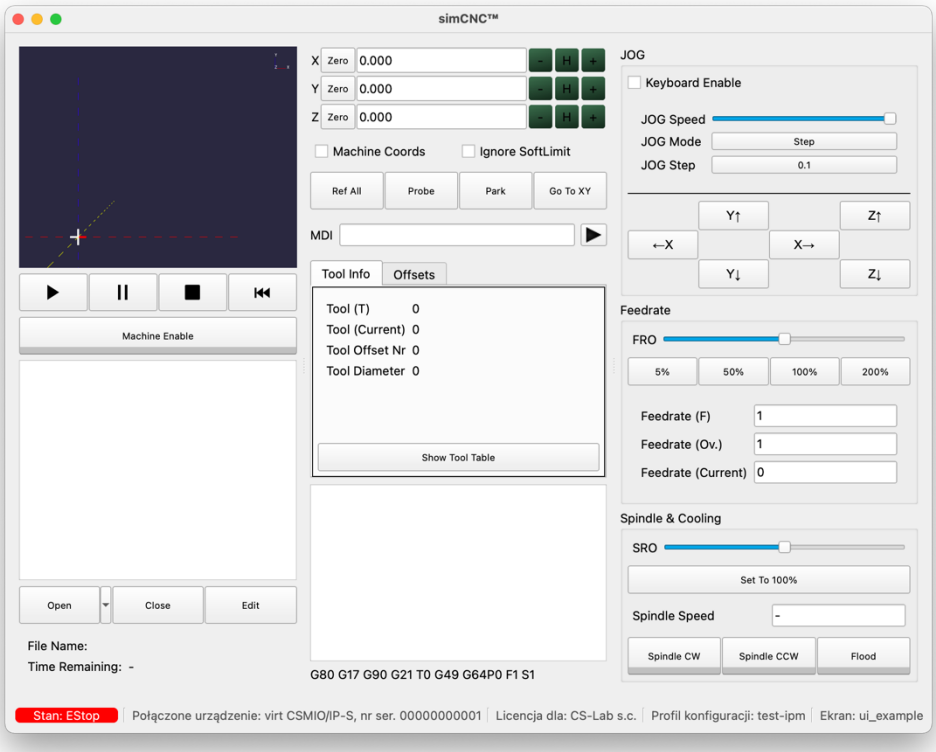
Gestaltung

Wie bereits erwähnt, ist dieser Schritt optional. Unsere Schnittstelle sollte in dieser Phase ordnungsgemäß funktionieren. Es lohnt sich jedoch, etwas mehr Arbeit zu widmen, um sein Aussehen zu verbessern.

Am Anfang werden wir die Skalierungs- und Platzaufteilungsregeln einiger Widgets und Container ein wenig „tunen“. Zu diesem Zweck definieren wir die folgenden Eigenschaften für die Objekte:

Bezeichnung („id“) des Objektes	Bezeichnung der Eigenschaften	Wert
loutLeftColumn	Stretch	1,0,0,1,0,0
loutExecutionCtrlButtons	minimum height	40
loutExecutionCtrlButtons	left, right, top, bottom margin	0
loutExecutionCtrlButtons	Spacing	1
btnCtrlEnable	minimum height	40
loutFileCtrlButtons	minimum height	40
loutFileCtrlButtons	left, right, top, bottom margin	0
loutFileCtrlButtons	Spacing	1
loutFileInfo	left, right, top, bottom margin	0
loutAxesDROs	left, right, top, bottom margin	0
loutAxesDROs	Spacing	1
loutAxisXDRO	left, right, top, bottom margin	1
loutAxisXDRO	Spacing	1
loutAxisYDRO	left, right, top, bottom margin	1
loutAxisYDRO	Spacing	1
loutAxisZDRO	left, right, top, bottom margin	1
loutAxisZDRO	Spacing	1
loutMiscButtons	minimum height	40
loutMiscButtons	left, right, top, bottom margin	0
loutMiscButtons	Spacing	1
loutToolInfoLabels	left, right, top, bottom margin	0
btnShowToolTable	minimum height	30
loutRightColumn	Stretch	1,0,0
loutJogConfig	vertical size policy	Maximum
frJogLine	vertical size policy	Maximum
loutFroButtons	minimum height	30
loutFroButtons	left, right, top, bottom margin	0
loutFroButtons	Spacing	1
btnFro5	vertical size policy	Preferred
btnFro50	vertical size policy	Preferred
btnFro100	vertical size policy	Preferred
btnFro200	vertical size policy	Preferred
btnSroReset	minimum height	30
loutSpindleAndCoolingCtrl	minimum height	40
loutSpindleAndCoolingCtrl	left, right, top, bottom margin	0
loutSpindleAndCoolingCtrl	Spacing	1

Die Schnittstelle sollte jetzt etwas übersichtlicher und kompakter sein:



Wir stellen noch die rote Farbe für die Endschalter-Statusanzeigeleuchten ein:

Bezeichnung („id“) des Objektes	Bezeichnung der Eigenschaften	Wert
ioAxisXLimitNeg	color	(rot)
ioAxisXLimitPos	color	(rot)
ioAxisYLimitNeg	color	(rot)
ioAxisYLimitPos	color	(rot)
ioAxisZLimitNeg	color	(rot)
ioAxisZLimitPos	color	(rot)

(...) sowie rechtsbündige Ausrichtung und etwas größere Schrift für die Achsennamen- und Koordinatenanzeige-Widgets:

Bezeichnung („id“) des Objektes	Bezeichnung der Eigenschaften	Wert
edAxisXPosition	horizontal alignment	Right
edAxisXPosition	font	Arial 20
lbAxisXName	font	Arial 20
edAxisYPosition	horizontal alignment	Right
edAxisYPosition	font	Arial 20
lbAxisYName	font	Arial 20
edAxisZPosition	horizontal alignment	Right
edAxisZPosition	font	Arial 20
lbAxisZName	font	Arial 20

(...) deaktivieren wir den zusätzlichen Rahmen im Widget mit den Registerkarten „Tool Info“ und „Offsets“:

Bezeichnung („id“) des Objektes	Bezeichnung der Eigenschaften	Wert
tabToolInfo	shape	Kein Rahmen
tabOffsets	shape	Kein Rahmen

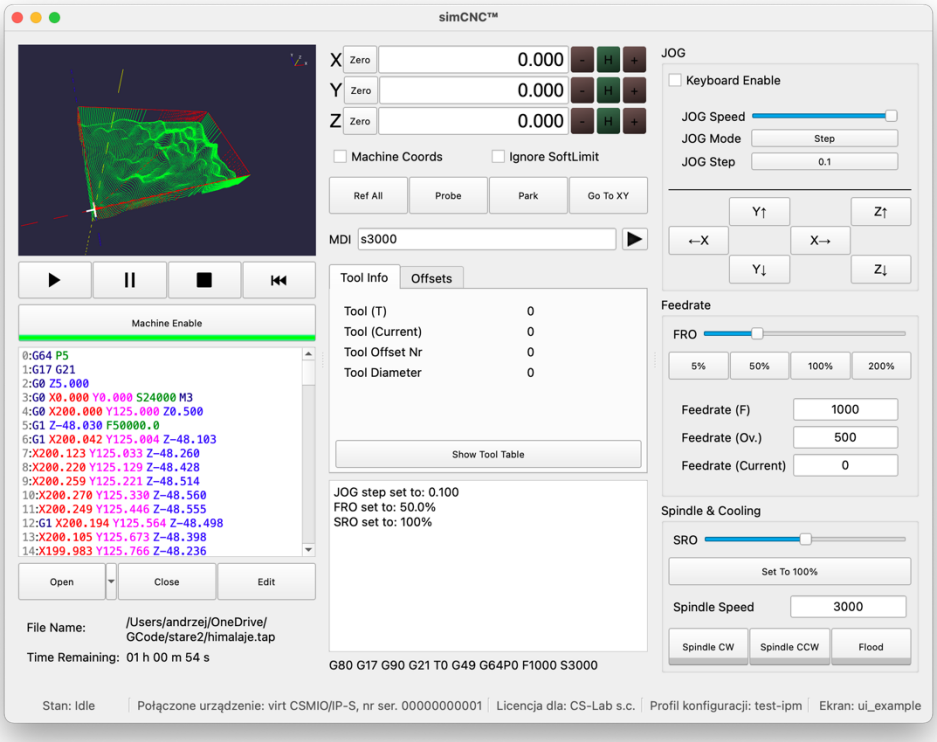
(...) zentrieren wir die Anzeige mancher Parameter

Bezeichnung („id“) des Objektes	Bezeichnung der Eigenschaften	Wert
lbSelectedTool	Horizontal alignment	Zentriert
lbCurrentTool	Horizontal alignment	Zentriert
lbToolOffsetNr	Horizontal alignment	Zentriert
lbToolDiameter	Horizontal alignment	Zentriert
edFeedrate	Horizontal alignment	Zentriert
edFeedrateOv	Horizontal alignment	Zentriert
edFeedrateCurrent	Horizontal alignment	Zentriert
edSpindleSpeed	Horizontal alignment	Zentriert

und stellen wir die Eigenschaften des lbFileName-Widgets so ein, dass längere Namen möglich sind:

Bezeichnung („id“) des Objektes	Bezeichnung der Eigenschaften	Wert
lbFileName	Word wrap	(markiert)
lbFileName	Vertical size policy	Maximum
lbFileName	Minimum height	32

Die Schnittstelle sieht jetzt wie folgt aus:



Endgültige Kosmetik mit css-Stylesheets

Die Gestaltung mit Stylesheets ist ein Werkzeug mit großen Möglichkeiten, das jedoch eher für fortgeschrittene Benutzer vorgesehen ist. Eine ausführliche Erörterung aller Eigenschaften würde über diese Anleitung hinausgehen. Weiter unten in diesem Kapitel wurde ein einfaches Beispiel präsentiert. Interessierte finden unter den folgenden Links ausführlichere Informationen zu diesem Thema.

https://www.w3schools.com/css/css_intro.asp
https://doc.qt.io/qt-5/stylesheet-syntax.html

Ich möchte Sie auch ermutigen, mit css-Dateien in Ihren Standardbildschirm-Projekten („default“) zu experimentieren. Denken Sie nur daran, eine Kopie zu erstellen und daran zu arbeiten, da alle Änderungen an den Standardbildschirmen nach der Aktualisierung der simCNC-Version überschrieben werden können.

Von simCNC übergebene Eigenschaften

simCNC übergibt einige Eigenschaften an die css-Sheets, die es ermöglichen, das Aussehen der Elemente zu ändern, z.B. in Abhängigkeit vom Programmstatus oder dem aktiven Dunkelmodus.

Bezeichnung der Eigenschaften	Werte	Beschreibung
darkTheme	true false	Gibt **true** zurück, wenn ein dunkles Motiv aktiv ist
state	estop idle homing trajectory jog mdi mpg	Gibt den aktuellen Zustand der Maschine zurück
csmioState	init disabled prepare run fault	Gibt den aktuellen Zustand des CSMIO/IP-Geräts
pause	true false	Gibt **true** zurück, wenn die Bearbeitung gestoppt ist (Pause)
Axis<X,Y,Z, …>Referenced	true false	Gibt **true** zurück, wenn die jeweilige Achse korrekt referenziert ist
Axis<X,Y,Z, …>Enabled	true false	Gibt **true** zurück, wenn die jeweilige Achse in der Konfiguration eingeschaltet ist

Annahmen für das Beispiel

Im folgenden Beispiel werden wir das folgende Gestaltungsschritte ausführen:

- Wir werden die Hintergrund- und Linienfarbe der Y-Achse in der 3D-Ansicht ändern
- Wir ändern die Farben der X-, Y- und Z-Achsennamen und machen diese vom Referenzierzustand der jeweiligen Achse abhängig
- Wir dunkeln den Hintergrund unter einigen Etiketten ab
- Wir lassen die JOG-Schaltflächen aufleuchten, wenn der Mauszeiger über ihnen schwebt
- Wir lassen die FRO- und SRO-Reset-Schaltflächen aufleuchten, wenn der Mauszeiger über ihnen schwebt

Gruppierung

Es kommt häufig vor, dass wir ähnliche Stileigenschaften für mehrere Widgets definieren möchten. Man kann sich die Arbeit erleichtern, indem man die Eigenschaft **group** im Schnittstelleneditor einstellt. Dadurch können wir uns in dem **csss**-Sheet auf eine Gruppe statt auf einzelne Elemente beziehen. Jetzt setzen wir die Eigenschaft **group** für die Widgets gemäß der folgenden Tabelle:

Bezeichnung („id“) des Widgets	Gruppenname (Eigenschaft group)
btnJogXPos	JogButtons
btnJogXNeg	JogButtons
btnJogYPos	JogButtons
btnJogYNeg	JogButtons
btnJogZPos	JogButtons
btnJogZNeg	JogButtons
btnFro100	Set100PercentButtons
btnSroReset	Set100PercentButtons
lbAxisXName	AxesNameLabels
lbAxisYName	AxesNameLabels
lbAxisZName	AxesNameLabels
lbSelectedTool	DarkerLabels
lbCurrentTool	DarkerLabels
lbToolOffsetNr	DarkerLabels
lbToolDiameter	DarkerLabels
lbFileName	DarkerLabels
lbTimeRemaining	DarkerLabels

Änderung der Farben in der 3D-Ansicht

Im Verzeichnis unserer Schnittstelle öffnen wir die Datei „colors.css“. Es kann ein beliebiger Texteditor verwendet werden, z.B. **Notepad** oder V**isual Studio Code**. Letzteres hat den Vorteil, dass es die Korrektheit der Syntax mit Farbe versieht und teilweise analysiert.

In der Datei bearbeiten wir die Hintergrundfarbe (**path_view_background**) und die Farbe der Y-Achse (**axisY**), um sie vor dem neuen Hintergrund besser sichtbar zu machen:

```
[type="path_view_background"] {
  color: #202020;
}
[type="axis_Y"] {
  color: #4646ff;
}
```

Erstellen einer neuen Stilsheet-Datei

Im Verzeichnis unserer Schnittstelle erstellen wir eine neue Textdatei mit dem Namen „widgets.css“, d.h. für unser Beispiel gilt unter Windows der folgende Pfad:

C:\Program Files\simCNC\screens\ui_example\widgets.css

Das simCNC-Programm sucht beim Laden des Bildschirms automatisch im Schnittstellenverzeichnis nach **css**-Dateien.

Änderung der Farben der Achsenetikette und Visualisierung des Referenzierstatus

In die Datei „widgets.css“ fügen wir die folgenden Anweisungen ein:

```
[group="AxesNameLabels"] {
  background-color: #e80;
  border-radius: 3px;
  margin: 2px;
  color: #444;
}

[id="lbAxisXName"][axisXReferenced="true"] {
  background-color: #a0000000;
  color: #0f0;
}
[id="lbAxisYName"][axisYReferenced="true"] {
  background-color: #a0000000;
  color: #0c0;
}
[id="lbAxisZName"][axisZReferenced="true"] {
  background-color: #a0000000;
```

Wie man sieht, beziehen wir uns hier auf die Widgets auf zwei Arten. Zuerst wird das Standardaussehen für die gesamte Gruppe festgelegt (**AxesNameLabels**), während nachstehend die bedingte Gestaltung für jede Achse separat durchgeführt wird, wenn die Flagge der Achsenreferenzierung gleich **true** ist.

Modifizierbare Eigenschaften sind:

- **background-color** – Hintergrundfarbe
- **border-radius** – abgerundete Ecken
- **margin** – Rand
- **color** – Etikett-Textfarbe

Unten sehen wir den Effekt, wenn die X-Achse referenziert ist, die Y- und Z-Achse jedoch nicht:
*(um die in css vorgenommenen Änderungen sichtbar zu machen, ist der Bildschirm neuzuladen: Menü **Konfiguration→Bildschirm neuladen**)*

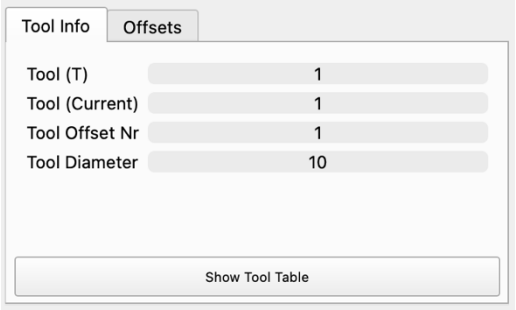
*Dunklerer Hintergrund unter den Etiketten aus der Gruppe **DarkerLabels***

```
[group="DarkerLabels"] {
  background-color: #10000000;
  border-radius: 5px;
}
```

In diesem Fall setzen wir nur zwei Parameter für die Gruppe mit dem Namen **DarkerLabels**:

- **background-color** – Hintergrundfarbe
- **border-radius** – abgerundete Ecken

Unten können wir den Effekt sehen – dunklerer Hintergrund und abgerundete Ecken unter den Etiketten:
*(um die in css vorgenommenen Änderungen sichtbar zu machen, ist der Bildschirm neuzuladen: Menü **Konfiguration→Bildschirm neuladen**)*

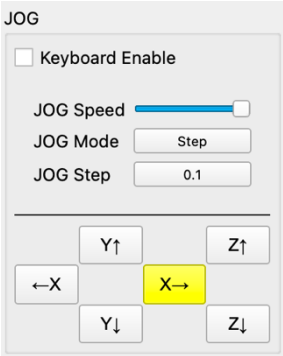


Änderung der Farbe (Hinterleuchten) der JOG-Schaltflächen

```
[group="JogButtons"]:hover {
  background-color: yellow;
}
```

Wie man oben sieht, modifizieren wir nur die Hintergrundfarbe (**background-color**), wobei wir dies bedingt machen. Das Schlüsselwort **hover** ist dafür verantwortlich, dass der Stil nur dann angewendet wird, wenn sich der Mauszeiger über dem Widget befindet. Dadurch wird der Effekt erzielt, dass die Schaltfläche „beleuchtet“ wird.

Nachstehend sehen wir den Effekt:
*(um die in css vorgenommenen Änderungen sichtbar zu machen, ist der Bildschirm neuzuladen: Menü **Konfiguration→Bildschirm neuladen**)*

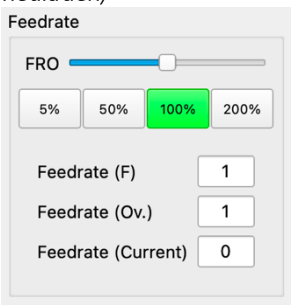


Änderung der Farbe (Hinterleuchten) der FRO- und SRO-Reset-Schaltflächen

```
[group="Set100PercentButtons"]:hover {
  background-color: #0f0;
}
```

Das Gleiche wie bei den JOG-Schaltflächen - wir modifizieren bedingt die Hintergrundfarbe (**background-color**). Das Schlüsselwort **hover** ist dafür verantwortlich, dass der Stil nur dann angewendet wird, wenn sich der Mauszeiger über dem Widget befindet. Dadurch wird der Effekt erzielt, dass die Schaltfläche „beleuchtet“ wird.

Nachstehend sehen wir den Effekt:
*(um die in css vorgenommenen Änderungen sichtbar zu machen, ist der Bildschirm neuzuladen: Menü **Konfiguration→Bildschirm neuladen**)*

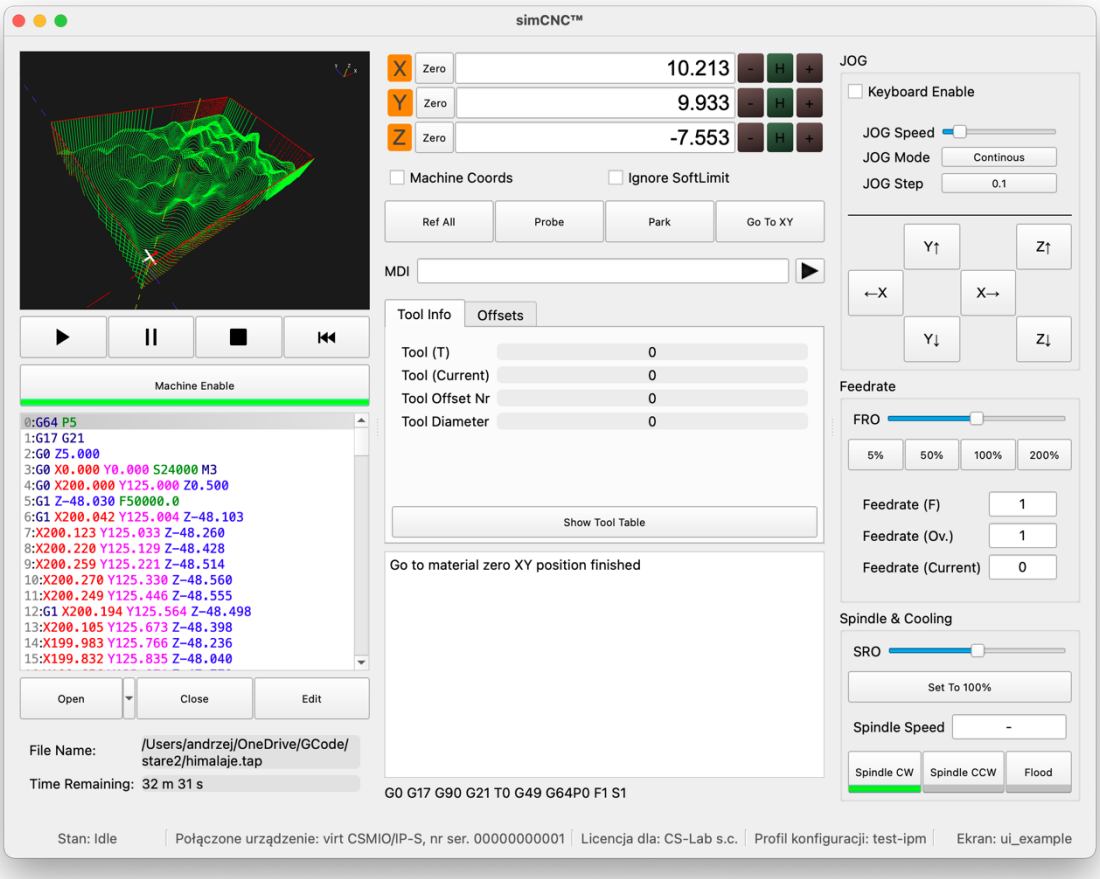


Der gesamte Inhalt der „widgets.css“-Datei

```
[group="AxesNameLabels"] {
  background-color: #e80;
  border-radius: 3px;
  margin: 2px;
  color: #444;
}

[id="lbAxisXName"][axisXReferenced="true"] {
  background-color: #a0000000;
  color: #0f0;
}
[id="lbAxisYName"][axisYReferenced="true"] {
  background-color: #a0000000;
  color: #0c0;
}
[id="lbAxisZName"][axisZReferenced="true"] {
  background-color: #a0000000;
  color: #0c0;
}

[group="DarkerLabels"] {
  background-color: #10000000;
  border-radius: 5px;
}

[group="JogButtons"]:hover {
  background-color: yellow;
}
```

Oben die gesamte Datei „widgets.css“. Wie man sieht, reichen schon wenige Zeilen aus, um den Entwurf der Schnittstelle optisch ansprechender zu gestalten und oft auch für den Benutzer bequemer und lesbarer zu machen.

Endeffekt und Zusammenfassung



So präsentiert sich der fertige Entwurf. Wie wir uns bei der Lektüre dieses Kapitels überzeugen konnten, erfordert die Erstellung einer Schnittstelle „von Grund auf“ etwas Arbeit, aber wenn wir eine solche Möglichkeit haben, kann die simCNC-Software relativ schnell an die komfortable Bedienung vieler Maschinentypen und deren Zubehör angepasst werden.

Der in diesem Kapitel beschriebene Entwurf wurde der simCNC-Standardinstallation (ab Version 3.410) hinzugefügt.

Wenn Sie Ihren eigenen Entwurf erstellt haben, das Sie mit anderen Nutzern teilen möchten, lassen Sie es uns wissen unter office@cs-lab.eu.

Das CS-Lab-Team grüßt Sie und wünscht Ihnen eine fruchtbare Arbeit und tolle Effekt ☺