Diagramme mit TikZ: Ein Beispiel, Teil II

Uwe Ziegenhagen

Nachdem ich beim letzten Mal gezeigt hatte, wie man ein Blockdiagramm mit TikZ zeichnen kann, soll es heute noch um Verbesserungen und alternative Herangehensweisen gehen.

Unsere Ausgangsbasis

Abbildung zeigt die Ausgangsbasis für den heutigen Artikel, das Endergebnis des letzten Artikels. Vergleicht man die Abbildung mit der Originalgrafik, die ebenfalls im letzten Artikel abgedruckt war, so fällt auf:

- Der Pfeil von `offset` zu `VCO2` ist gekrümmt und nicht waagerecht, so sieht das nicht schön aus
- Die Pfeile von `Noise` und `VCO1` zum `VCF`-Node sind im Original keine Kurven, sondern ebenso eckige Linien.

Man könnte jetzt natürlich anmerken, dass die Grafik ja schon »gut genug« sei, aber dann könnte man ja auch gleich Word benutzen. :-)

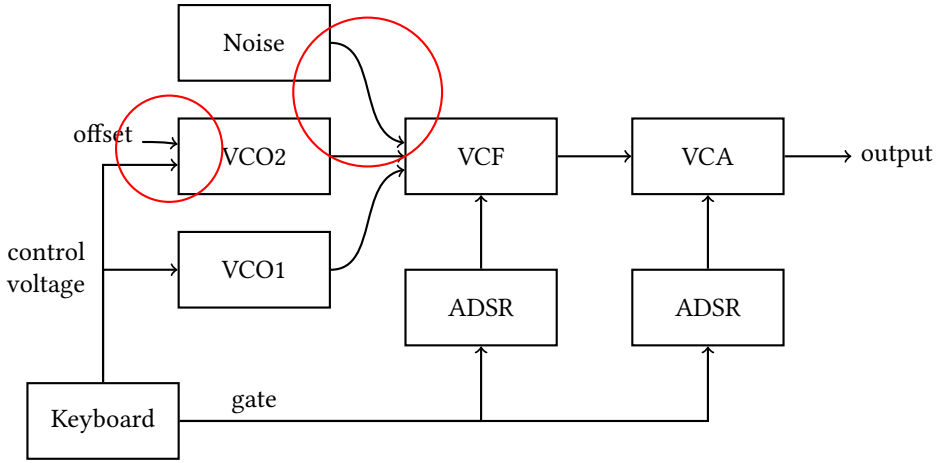

Abb. 1: Das Ergebnis des letzten Artikels mit den »Problemstellen«

Beim letzten Mal hatte ich die Positionen der einzelnen Nodes auch über ihre Koordinaten festgelegt. Dieser Ansatz ist natürlich völlig legitim, TikZ unterstützt jedoch auch noch die Positionierung von Nodes über die relativen Beziehungen zu den anderen Nodes. Wie dies geht und wie wir die oben erwähnten Kurven in ansprechende Linien-Züge umwandeln können, das möchte ich heute zeigen. Wer mitmachen möchte, der findet den Quellcode in Listing 1, über meine Webseite `www.uweziegenhagen.de` werde ich ihn auch zur Verfügung stellen.

```
\begin{tikzpicture}[
box/.style={rectangle,thick,draw=black,
minimum width=20mm, minimum height=10mm,align=center}]
\node at (0,1) [box] (keyboard) {Keyboard};

\node at (2,6) [box] (noise) {Noise};
\node at (2,4.5) [box] (vco2)  {VCO2};
\node at (2,3) [box] (vco1) {VCO1};

\node at (5,4.5) [box]  (vcf) {VCF};
\node at (5,2.5) [box] (adsr1) {ADSR};

\node at (8,4.5) [box] (vca) {VCA};
\node at (8,2.5) [box] (adsr2) {ADSR};

\node at (10.5,4.5) (output) {output};
\node at (0,4.81) (offset) {offset};
\node at (2,1.25) (gate) {gate};
\node at (-0.75,3) (cv) {\parbox{\widthof{control}}{
    control \\ voltage }};

\draw [thick,->] (vco1) to [out=0,in=190]  (vcf);
\draw [thick,->] (vco2) -- (vcf);
\draw [thick,->] (adsr1) -- (vcf);
\draw [thick,->] (adsr2) -- (vca);
\draw [thick,->] (keyboard) -| (adsr1);
\draw [thick,->] (keyboard) -| (adsr2);
\draw [thick,->] (keyboard) |- (vco1);
\draw [thick,->] (keyboard) |- (vco2.base west);
\draw [thick,->] (noise) to [out=0,in=170]  (vcf);
\draw [thick,->] (vcf) -- (vca);
\draw [thick,->] (vca) -- (output);
\draw [thick,->] (offset.base east) to [out=0,in=171] (vco2);
\end{tikzpicture}
```

Listing 1: Quellcode für Abbildung 1

Grundlagen der Positionierung

Kümmern wir uns zunächst um die alternative Positionierung der Nodes, bevor wir uns den Koordinatenberechnungen widmen. Das folgende Beispiel wurde einem TSX-Artikel von Gonzalo Medina (`https://tex.stackexchange.com/questions/69439/how-can-i-achieve-relative-positioning-in-tikz`) entlehnt, es zeigt, wie man mit der TikZ `positioning`-Library einzelne Nodes recht einfach relativ zueinander positionieren kann.

Wir beginnen wir mit einem einfachen Node, den wir an die Koordinaten (0,0) setzen. Über Angaben wie `below = of Nodename` können wir TikZ anweisen, die neuen Nodes zu positionieren. Neben »left«, »right«, »above« und »below« gibt es für »above« und »below« noch »left« und »right«-Varianten. Abbildung 2 zeigt die entspechenden Positionen.

```
\begin{tikzpicture}[
box/.style={rectangle,thick,draw=black,minimum width=20mm, minimum height=10mm,
 ↪align=center}]
\node at (0,0) [box] (a) {a};
\node [below = of a,box] (b)  {below};
\node [above = of a,box] (c)  {above};
\node [left = of a,box] (d)  {left};
\node [right = of a,box] (e)  {right};
\node [below left = of a,box] (f)  {below left};
\node [below right= of a,box] (g)  {below right};
\node [above left = of a,box] (h)  {above left};
\node [above right= of a,box] (i)  {above right};
\end{tikzpicture}
```

Listing 2: Quellcode für Abbildung 2

Zusätzlich zu diesen relativen Angaben lassen sich noch numerische Anpassungen vornehmen, wie das Beispiel in Listing 3 und Abbildung 3 zeigt.

Hier werden die Label `A1`–`A4` nicht nur rechts von `a` gesetzt, sondern mit einem definierten Abstand von ein bis vier Zentimeter.

Dies funktioniert sogar für die X- und Y-Richtung getrennt, siehe `A5` und `A6`. `A5` entspricht dabei »below right«, bei `A6` wurde der `below`-Abstand auf zwei Zentimeter, der `right`-Abstand auf einen Zentimeter gesetzt. Die Angabe `below right=2cm and 1cm of a` entspricht also nicht einfach den X- und Y-Koordinaten, hier ist es stattdessen eher ein (Y,X)-Pärchen.

```
\begin{tikzpicture}[
box/.style={rectangle,thick,draw=black,
minimum width=20mm, minimum height=10mm,
```

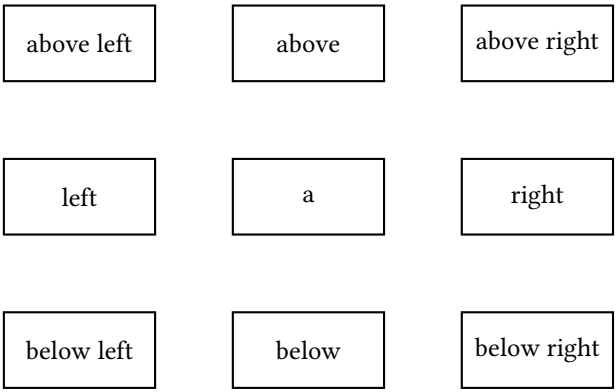

Abb. 2: Node-Positionierung mit der `positioning`-Library

```
align=center}]
\node at (0,0) [box] (a) {a};
\node (A1) [right=1cm of a] {A1};
\node (A2) [right=2cm of a] {A2};
\node (A3) [right=3cm of a] {A3};
\node (A4) [right=4cm of a] {A4};
\node (A5) [below right=0cm and 0cm of a] {A5};
\node (A6) [below right=2cm and 1cm of a] {A6};
\end{tikzpicture}
```

Listing 3: Quellcode für Abbildung 3

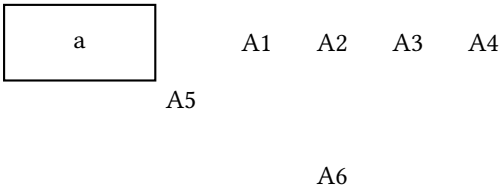

Abb. 3: Node-Positionierung mit der `positioning`-Library

Damit haben wir jetzt alle notwendigen Informationen, um unsere Synthesizer-Nodes relativ setzen zu können. Die für die weiteren Anpassungen notwendigen Pfeile habe ich auch eingezeichnet.

```
\begin{tikzpicture}[
box/.style={rectangle,thick,draw=black,
minimum width=20mm, minimum height=10mm,
align=center,node distance=0.5cm}]

\node at (0,0) [box] (noise) {Noise};
\node[box, below = of noise] (vco2) {VCO2};
\node[box, below = of vco2] (vco1) {VCO1};

\node[box, right = of vco2] (vcf) {VCF};
\node[box, right = of vcf] (vca) {VCA};

\node[box, below = of vcf] (adsr1) {ADSR};
\node[box, below = of vca] (adsr2) {ADSR};

\node[right = 0.5cm of vca] (output) {output};
\node[box, below left = 0.5cm of vco1] (keyboard) {Keyboard};
\node[left = 1.5cm of vco2](cv) {offset};

\node[left = 1.5cm of vco1](cv){\parbox{\widthof{control}}{control \\ voltage}};

\draw[thick,->] (keyboard) |- (vco2.base west);
\draw[thick,->] (vco1) to [out=0,in=190]  (vcf);
\draw[thick,->] (vco2) -- (vcf);
\draw[thick,->] (noise) to [out=0,in=170]  (vcf);
\draw[thick,->] (offset.base east) to [out=0,in=171] (vco2);
\end{tikzpicture}
```

Listing 4: Quellcode für Abbildung 4

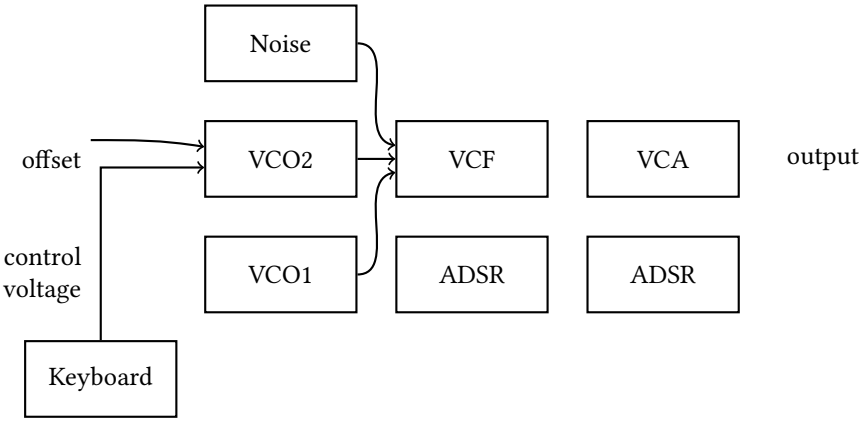

Abb. 4: Alle Nodes relativ positioniert, Ausgabe von Listing 4

Berechnungen

Der Pfeil von `offset` zu `VCO2`

Die relative Positionierung der Nodes haben wir nun umgesetzt, als nächstes geht es darum, die Punkte zu berechnen, durch die wir unsere Linien ziehen wollen; Abbildung 5 zeigt diese:

- links in `VCO2` der Eintrittspunkt für den Pfeil von `offset`
- mit blauen Punkten markiert der Pfeilweg von `Noise` zu VCF
- mit roten Sternchen markiert der Pfeilweg von `VCO1` zu `VCF`

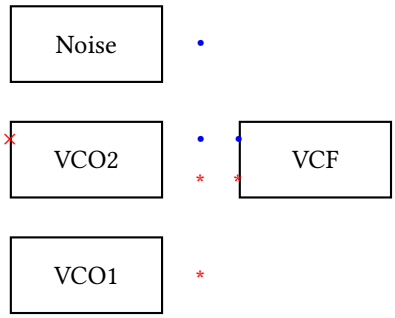

Abb. 5: Zu berechnende Punkte

Für Koordinatenberechnungen bietet die `calc` Library – die über `\usetikzlibrary{calc}` geladen wird – diverse Rechenbefehle, für uns ist jedoch nur eine Syntax relevant:

```
($‹coordinate›!‹number›!‹coordinate›$)
```

`‹coordinate›` steht dabei für eine Koordinate, die – vielleicht etwas vereinfachend erklärt – einfach nur ein Node ohne den (optionalen) Text ist. `‹number›` ist eine Zahl zwischen 0 und 1 und gibt in Dezimaldarstellung den Prozentsatz an, um den wir uns von Koordinate 1 zu Koordinate 2 bewegen. `0.25` steht also für ein Viertel des Weges, `0.5` für den Mittelpunkt zwischen den beiden Koordinaten, etc.

Damit können wir jetzt also die Mittelpunkte zwischen zwei Nodes/Koordinaten bestimmen. Abbildung 7, entnommen einem Artikel von TeXWelt (`https://texwelt.de/fragen/21486/tikz-wie-muss-ich-den-anchor-richtig-setzen`) zeigt noch einmal die relevanten Anker in einem Rechteck, mit denen wir wir jetzt arbeiten werden.

Den Eintrittspunkt des `offset`-Pfeils können wir direkt bestimmen: es ist die Koordinate, die genau mittig zwischen `north west` und `west` liegt. Listing 5 zeigt den Quellcode für die Berechnung, Abbildung 6 die entsprechende Ausgabe.

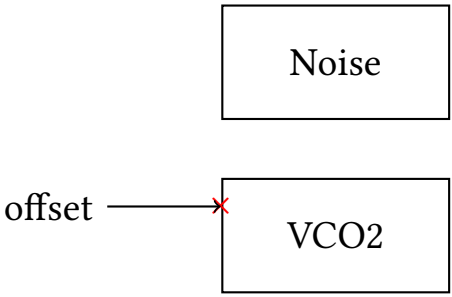

Abb. 6: Ausgabe von Listing 5

```
\node at (0,0) [box] (noise) {Noise};
\node [box, below = of noise] (vco2) {VCO2};
\coordinate (coordoffset) at ($(vco2.west)!0.5!(vco2.north west)$);
\node at (coordoffset){\textcolor{red}{$\times$}};

\node [left = of coordoffset](offset) {offset};
\draw [thick,->] (offset) -- (coordoffset);
```

Listing 5: Quellcode für Abbildung 6

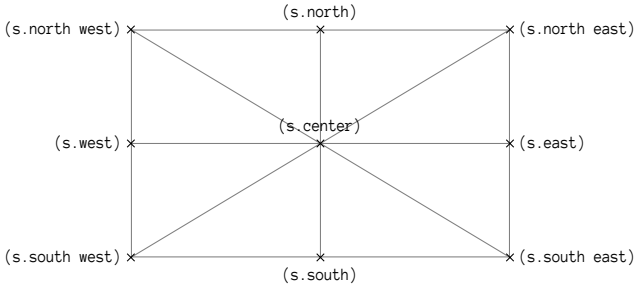

Abb. 7: Anker in einem Rechteck, Quelle: `https://texwelt.de/fragen/21486/tikz-wie-muss-ich-den-anchor-richtig-setzen`

Der Weg des Pfeils

Die Punkte für die Pfeilwege zu bestimmen ist ein wenig aufwändiger: Zuerst definieren wir einen Nodes `temp1` , der rechts von `Noise` liegt. Er wird im finalen

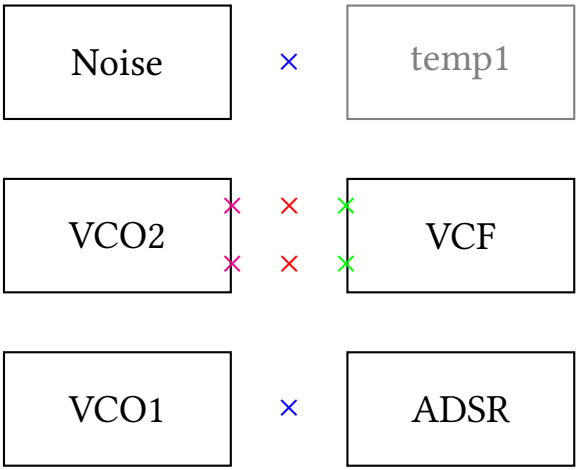

Abb. 8: Hilfs-Node `temp1` und die Punkte, durch die die Linien gehen werden

Diagramm nicht auftauchen, sondern dient nur zur Berechnung der korrekten Abstände für den oberen Pfeil. Für den Pfeil von `VCO1` zu `VCF` benötigen wir keinen solchen Hilfs-Node, hier nutzen wir einfach den linken der beiden `ADSR` Nodes. Abbildung 8 zeigt sie zusammen mit den Hilfskoordinaten für die Linien.

Die Mittelpunkte zwischen `Noise` und `temp1` bzw. zwischen `VCO1` und `adsr1` bilden dann die ersten Koordinaten für die Pfeilwege, im Quellcode werden sie `noisetemp1` und `vco1adsr1` genannt.

Als nächstes bestimmen wir für die Nodes `VCO2` und `VCF` jeweils die Hilfskoordinaten, die jeweils zwischen Mitte und oberer Ecke beziehungsweise Mitte und unterer Ecke liegen, insgesamt also vier Stück. Diese Koordinaten haben auch nur den Zweck, Hilfskoordinaten für die finale Berechnung zu sein, die Bestimmung der Mittelpunkte, im Diagramm mit roten »mal«-Symbolen markiert. Damit haben wir jetzt alle Punkte und Hilfspunkte, um das finale Diagramm in Abbildung 9 zu zeichnen, Listing 6 enthält dazu den vollständigen Code.

```
\begin{tikzpicture}[box/.style={rectangle,thick,draw=black,
minimum width=20mm, minimum height=10mm,
align=center,node distance=0.5cm}]

\node at (0,0) [box] (noise) {Noise};
\node [box, below = of noise] (vco2) {VCO2};
\node [box, below = of vco2] (vco1) {VCO1};
\node [box, right = 1cm of vco2] (vcf) {VCF};
```

```
\draw [thick,->] (vco2) -- (vcf);

\coordinate (coordoffset) at ($(vco2.west)!0.5!(vco2.north west)$);

\node [left = of coordoffset](offset) {offset};
\draw [thick,->] (offset) -- (coordoffset);
\node [right = 1cm of noise] (temp1) {};

\coordinate (vcf1) at ($(vcf.west)!0.5!(vcf.north west)$);
\coordinate (vcf2) at ($(vcf.west)!0.5!(vcf.south west)$);
\coordinate (vco21) at ($(vco2.north east)!0.5!(vco2.east)$);
\coordinate (vco22) at ($(vco2.south east)!0.5!(vco2.east)$);

\node [box, right = of vcf] (vca) {VCA};
\node [box, below = of vcf] (adsr1) {ADSR};
\node [box, below = of vca] (adsr2) {ADSR};
\node [right = 0.5cm of vca] (output) {output};
\node [box, below left = 0.5cm of vco1] (keyboard) {Keyboard};

\node[left = 1.5cm of vco1](cv){\parbox{\widthof{control}}{control \\ voltage
 ↪}};

\coordinate (noisetemp1) at ($(noise.east)!0.5!(temp1.west)$);
\coordinate (vco1adsr1) at ($(vco1.east)!0.5!(adsr1.west)$);

\coordinate (vco2vcf1) at ($(vco21)!0.5!(vcf1)$);
\coordinate (vco2vcf2) at ($(vco22)!0.5!(vcf2)$);

\draw[->,thick] (vco1.east) -- (vco1adsr1) --  (vco2vcf2) -- (vcf2);
\draw[->,thick] (noise.east) -- (noisetemp1) --  (vco2vcf1) -- (vcf1);

\draw[->,thick] (adsr1) -- (vcf);
\draw[->,thick] (adsr2) -- (vca);
\draw[->,thick] (vcf) -- (vca);
\draw[->,thick] (vca) -- (output);

\draw [thick,->] (keyboard) -| (adsr1);
\draw [thick,->] (keyboard) -| (adsr2);
\draw [thick,->] (keyboard) |- (vco1);
\draw [thick,->] (keyboard) |- (vco2.base west);
\end{tikzpicture}
```

Listing 6: Quellcode für Abbildung 9

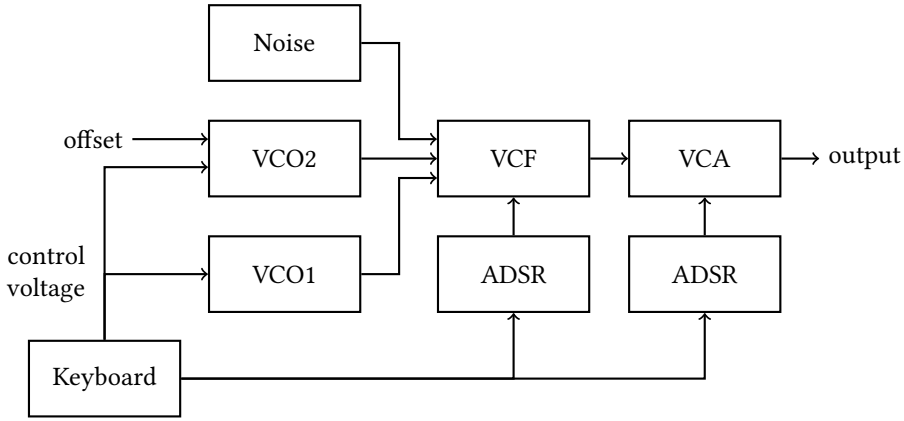

Abb. 9: Das »fertige« Diagramm

Fazit

War das Ergebnis diesen Aufwand wert? Ja, ich denke schon, wenngleich es mit Affinity Designer oder Inkscape um einiges schneller gegangen wäre! Das Schöne an TikZ-Code ist aber – wie auch an LaTeX allgemein, dass wenn man den Bogen erst einmal »raushat«, es beim nächsten Mal viel schneller geht. Das Diagramm ist auch – wie ich finde – ein schönes Beispiel, um einige der Möglichkeiten kennenzulernen, die TikZ uns bietet.