Signalverstärkung durch Rauschen

von

Jonathan Schindler
Florian Fülbert

WEIRD SCIENCE CLUB
DARMSTADT

an der

Europaschule
Lichtenbergschule

Ludwigshöhstr. 105
64285 Darmstadt

Inhaltsverzeichnis

1. Einleitung

Bei der Aufnahme, Auswertung und Übertragung von Daten möchte man ein Rauschen stets so gut wie irgend möglich vermeiden, denn das Rauschen stört das Signal, verzerrt es und kann es bis zur Unkenntlichkeit „untergehen“ lassen. Es wird deshalb in der Regel nicht nur größte Sorgfalt angewandt, Rauschen zu unterdrücken, es wird auch ein hoher Grad an technischen Mitteln eingesetzt, um etwaiges Rauschen auszufiltern.

Bei einem Besuch des „Chaos-Labors“ der Arbeitsgruppe von Prof. Benner an der TU Darmstadt erfuhren wir von einem sehr jungen Forschungsgebiet der Physik – der stochastischen Resonanz. Dabei werden schwache Signale mit Rauschen einer bestimmten „Resonanzintensität“ überlagert und dadurch verstärkt. Für diesen hochinteressanten Effekt gibt es bislang noch keine Anwendung obwohl die Signalverstärkung von vielen Autoren als mögliche Technologie in Betracht gezogen wird.

Uns faszinierte die scheinbar widersprüchliche Natur einer Signalverstärkung durch Rauschen und wir wollten untersuchen, ob es möglich ist, ein beliebiges aperiodisches Signal mit Methoden der stochastischen Resonanz zu verstärken.

2. Zielsetzung

Wir wollen ein System entwickeln mit dem aperiodische Signale mit Hilfe stochastischer Resonanz verstärkt werden können.

3. Theorie

Stochastische Resonanz

Das Standardmodell der stochastischen Resonanz geht von einem bistabilen System aus, in dem sich ein Teilchen stark gedämpft bewegen kann. Hinzu kommt ein Rauschen $\zeta(t)$. Die Bewegungsgleichung des Teilchens (Masse m=1) lautet dann:

$$\ddot{x} = -\gamma\dot{x} - \frac{dV_{eff}(x,t)}{dx} + \zeta(t)$$

Hierbei ist γ die Dämpfung und V_{eff} ein effektives Potential (Abbildung 1), welches aus einem statischen Potential und einer Modulation besteht. Diese Modulation verändert die Potentialbarriere ΔV periodisch.

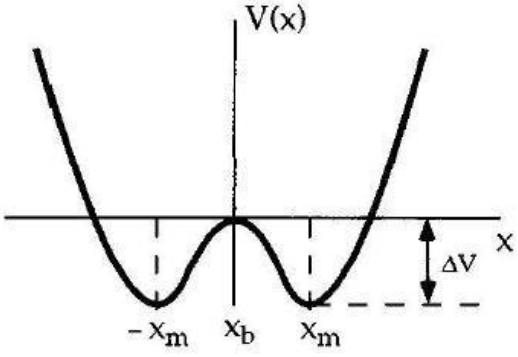


Abbildung 1: Doppelmuldenpotential

Regt man das System stochastisch an, kann es zu zufälligen Übergängen zwischen den beiden Mulden kommen (Abbildung 2).

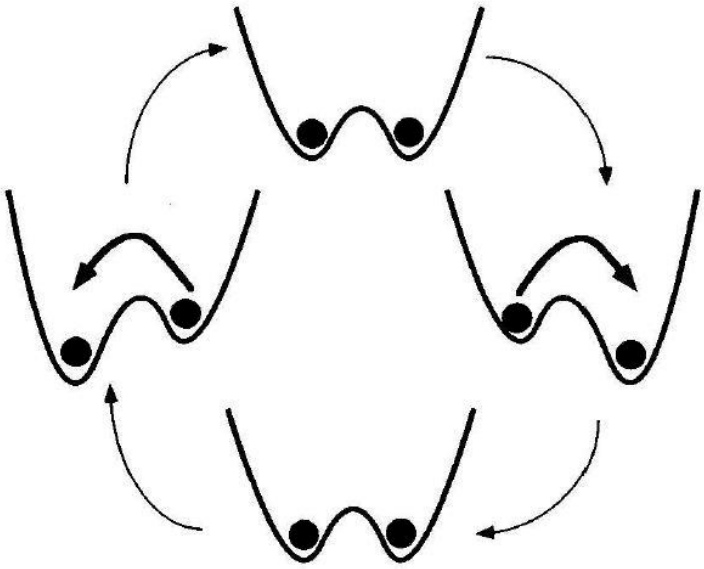


Abbildung 2: Grundprinzip der stochastischen Resonanz

Die Übergangsraten, die sog. „Kramers Raten“, sind aufgrund der Symmetrie des Systems bei einem statischen System natürlich in beide Richtungen gleich groß und können durch:

$$r_K = \frac{\omega_0 \omega_b}{2\pi\gamma} \exp(-\frac{\Delta V}{D})$$

berechnet werden, wobei ω_0 und ω_b die Krümmung an den jeweiligen Extrema (x_b und x_m) und D die Rauschintensität ist. Fügt man eine periodische Modulation hinzu, verändert sich nun die Potentialbarriere und dadurch natürlich auch die Übergangsraten periodisch.
Wie in Abbildung 2 zu sehen ist, wird während eines Zyklus die Potentialbarriere zweimal minimal, einmal für Übergänge von rechts nach links und einmal für die andere Richtung. Genau dies nutzt man aus. Da an diesen Stellen die Übergänge wesentlich wahrscheinlicher stattfinden, kommt es im zeitlichen Mittel vermehrt zu Sprüngen die synchron zum Signal (also der periodischen Anregung) sind und das obwohl das Signal allein nicht ausreichen würde einen Sprung zu erzeugen.
Um die optimale Synchronisation zu erreichen muss die mittlere Verweildauer zwischen zwei Sprüngen der Hälfte der Periodendauer entsprechen:

$$\frac{1}{r_K} = \frac{2\pi}{\omega}$$

Man sieht sofort, dass diese Bedingung nur von der Modulationsfrequenz ω und der Rauschintensität D abhängt und dass bei einer gegeben Größe die andere so gewählt

werden kann das eine optimale Synchronisation und damit stochastische Resonanz erreicht werden kann.

Rauschen:

Als Rauschen benutzt man in der Theorie üblicherweise ein gaußsches, weißes Rauschen. Das heißt, dass die Amplituden gaußverteilt sind und theoretisch alle Frequenzen mit gleicher Leistung abgedeckt werden. (daher auch der Begriff „weiß“ in Analogie zu dem weißen Licht). Praktisch ist dies natürlich nicht möglich, da das Rauschen dadurch eine unendliche Energie hätte. Die Intensität fällt daher bei sehr hohen Frequenzen ab, was aber in unserem Fall irrelevant ist, da wir nicht in diesem Bereich arbeiten.

Kreuzkorrelationsfunktion:

Normalerweise wird das sogenannte Signal-Rausch-Verhältnis benutzt, um die Qualität eines verrauschten Signals zu bestimmen. Dies funktioniert allerdings aufgrund eines zu breiten Frequenzspektrums nicht bei aperiodischen Signalen, daher müssen wir die Kreuzkorrelationsfunktion benutzen um das Ein- und Ausgangsignal zu vergleichen und so ein Maß für die Qualität unseres Systems zu bekommen.

Bei der Kreuzkorrelation handelt es sich um die Mittelung über das Produkt von zwei zeitlich zueinander verschobenen Signalen.

Sie ist als

$$\mathbf{C_{xy} = \lim_{T\to\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{+\frac{T}{2}} x(t) \cdot y(t+\tau) dt}$$

definiert. Zur besseren Vergleichbarkeit wird sie häufig auf die Standardabweichung der Signale normiert, sie nimmt dann Werte zwischen +1 und -1 an. Sie ist ein Maß dafür, wie sehr sich zwei Signale ähneln.

Hamming-Code:

Um fehlerhafte Bits zu identifizieren und korrigieren benutzen wir den so sog. (7,4)-Hamming-Code. Dieser hat eine Länge von 7 Bit, welche aus 4 Bits des eigentlichen Signals und 3 Kontrollbits besteht. Die Coderiungsvorschrift ist hierbei:

$$(x_0,...,x_3) \rightarrow (x_0,...,x_6)$$

mit

$$x_4 = x_1 + x_2 + x_3 \pmod 2$$

$$x_5 = x_0 + x_2 + x_3 \pmod 2$$

$$x_6 = x_0 + x_1 + x_3 \pmod 2$$

Dadurch werden 3 Prüfsummen in den Kontollbits gespeichert. Nach der Übertragung kann die Bitfolge y mit Hilfe einer Kontrollmatrix

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

decodiert werden. Hierfür berechnet man das Syndrom

$$s = Hy = H(x + e)$$

wobei e ein Fehlervektor ist.
Sollte s=0 sein, wurde die Nachricht korrekt übertragen, andernfalls entspricht das fehlerhafte Bit der Spalte i von H, die mit s übereinstimmt und kann somit korrigiert werden.

4. Experimente

4.1. Simulation

Nachdem wir uns in die theoretischen Grundlagen eingearbeitet hatten, beschlossen wir zunächst einige Simulationen durchzuführen. Hierfür programmierten wir ein Programm, mit dessen Hilfe es möglich war, verschiedene periodische oder aperiodische Signale zu erzeugen und dann mit einem Rauschen beliebiger Intensität zu überlagern. Die so simulierten Werte analysierten wir mithilfe des Programms xcor, welches die Kreutzkorrelationsfunktion zwischen zwei Signalen angibt. Dieses Programm gab uns einen Maximalwert der Übereinstimmung des Eingangssignals und des Ausgangssignals und die dazugehörige Rauschintensität an.
Mehrere Messreihen zeigten uns eine maximale Übereinstimmung von über 95%.

4.2 Planung des Versuchsaufbaus

Das aperiodische Signal, von einem externen Funktionsgenerator kommend, wird als erstes in einen Spannungsfolger geleitet. Dies war anfangs nicht eingeplant, doch da wir Probleme mit Rückkopplung des Rauschens hatten und dadurch unser Eingangsignal nicht zuverlässig messen konnten, bauten wir diesen Spannungsfolger zur besseren Messung ein.
Anschließend wird diesem Eingangssignal in einem Addierer ein normalverteiltes Rauschen überlagert, das ein zweiter externer Funktionsgenerator erzeugt.

Das aperiodische, mit Rauschen überlagerte Signal kommt danach zu einem invertierenden Schmitt-Trigger, der bei überschreiten einer gewissen Schwelle, die kleinst mögliche Ausgangsspannung ausgibt. Bleibt das Signal unter dieser Schwelle erzeugt der Schmitt-Trigger hingegen die größt mögliche Spannung. Man erhält somit ein digitales Signal: Die obere Schwelle repräsentiert logisch eine 0, die untere Schwelle logisch eine 1. Der Schmitt-Trigger arbeitet somit invertierend. Anzumerken ist, das der Schmitt-Trigger eine eigene Spannungsversorgung von 5V benötigt (alle anderen Operationsverstärker werden mit 15V betrieben), da die Schaltzeit von der sogenannten Slew-Rate des Operationsverstärkers abhängt. Diese beschreibt die Geschwindigkeit, mit der eine Spannungsdifferenz am Ausgang erzeugt wird. Da der Schmitt Trigger ein in Sättigung betriebener OP ist, er also zwischen der positiven und der negativen Sättigungsspannung hin- und herschaltet, sollte diese Spannungsdifferenz minimiert werden, um die kleinstmögliche Umschaltzeit zu erhalten. Da aufgrund der endlichen Eingangswiderstände der Operationsverstärker ungewollte Störungen und Rückkopplungseffekte auftraten, bauten wir zur Stabilisierung nachträglich einen weiteren Operationsverstärker ein, der außerdem das Signal noch verstärkt.
Der letzte Operationsverstärker ist ein Treiber und passt die Impedanz an die Messkarte des PCs an.

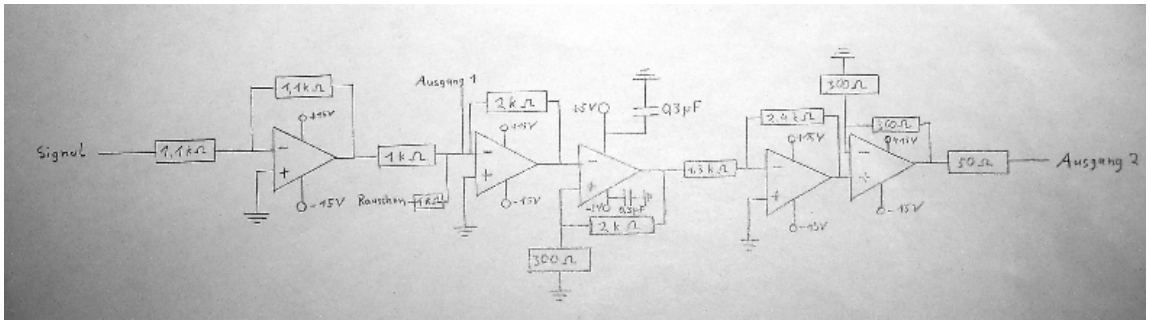


Abbildung 3: Schaltskizze

4.3 Praktische Durchführung

Die Planungen mussten nun auf einer Steckplatine verwirklicht werden. Im Chaos-Labor der TU-Darmstadt waren alle nötigen Komponenten vorhanden: Steckplatine, Operationsverstärker, Kabel, Spannungsquellen, Funktionsgeneratoren und BNC-Kabel zum Anschluss an den PC und die Geräte.
So ließ sich der Versuchsaufbau in der Realität verwirklichen. Wir konnten nun erste Zeitreihen aufnehmen. Diese zeigten zufrieden stellende Werte.

Nun, da sichergestellt war, dass der Versuchsapparat verlässlich funktionieren würde, löteten wir den Versuchsaufbau auf eine Platine. Dabei gingen wir äußerst vorsichtig vor, da wir keinerlei Erfahrungen mit dem Löten hatten.

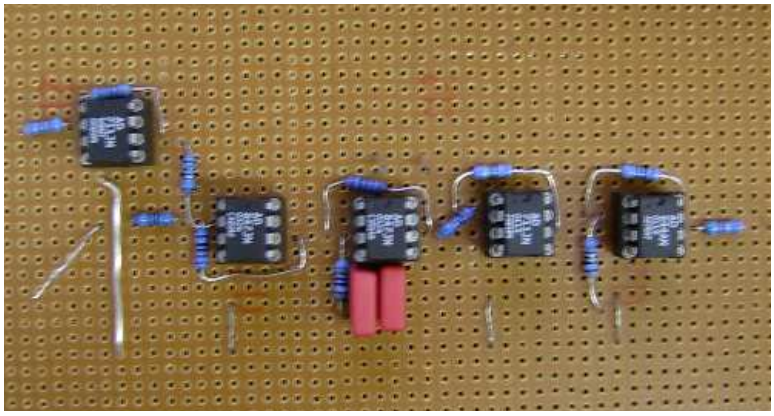


Abbildung 4: Unsre Platine: Spannungsfolger, Addierer, Schmitt-Trigger, Verstärker, Treiber

Als die Platine fertig waren, konnten intensivere Messungen beginnen. Hier nahmen wir mehrere Zeitreihen auf. Dabei war der Ablauf so:

Mit einem Funktionsgenerator wird ein aperiodisches Signal einer gewissen Amplitude und Frequenz erzeugt. Dieses wird über ein BNC Kabel in den Eingang unseres Aufbaus geleitet, aber auch in die Messkarte des PCs. Das Signal wird dann mit normalverteiltem Rauschen überlagert. Der Schmitt-Trigger wandelt dieses analoge Signal in ein digitales Signal um. Dieses Signal wird dann an die Messkarte des PCs geleitet. Mithilfe des Programms LabView werden die Werte des Eingangs- und des Ausgangssignals aufgezeichnet und in einer Datei ausgegeben.
Diese Datei wird dann mit Hilfe des Programms xcor ausgewertet, das die Kreuzkorrelationsfunktion ausführt, also die Übereinstimmung der Werte quantifiziert.

4.4 Digitales Signal

Das Ziel, ein Audiosignal durch den Versuchsaufbau zu schicken und auszuwerten, wollten wir mit einem digitalen Audiosignal erreichen.

Dazu mussten wir unsere Apparatur mit einem digitalen Signal testen.
Mit LabView erstellten wir ein Programm, das einen Text über den ASCII-Code in eine Bitfolge umwandelt. Diese Bitfolge wird dann in einen GPIB –Quelltext integriert und an einen Funktionsgenerator übergeben. Dieser gibt dann ein Rechtecksignal aus, das durch die Apparatur geschickt werden kann.
Wie oben wurden Messreihen aufgenommen und ausgewertet. Das Ergebnis war eine Übereinstimmung von über 75%. Dieses Ergebnis kann noch verbessert werden, indem wir die Werte eines jeden Bits mitteln und so kleinere Sprünge beseitigen.

Ein großes Problem ist die Synchronisation, d.h. zu erkennen, wann ein Signal bzw. ein Bit anfängt. Daher ließen wir den Signalgenerator zusätzlich ein Triggersignal senden, welches unsere Signale synchronisierte. Außerdem schickten wir eine Startsequenz von 8 Einsen um den Anfang eines Signals zu markieren und berechneten zusätzlich die mittlere Verweildauer bei einem Bit.
Allerdings ist die Redundanz des ASCII-Codes gleich Null, d.h. ein einziges fehlerhaftes Bit verändert das gesamte Zeichen.

Dadurch konnten wir nur fehlerhafte Nachrichten übertragen. Allein durch die hohe Redundanz der Schriftsprache war es möglich, den Wortlaut des Textes zu rekonstruieren. Um dies zu verbessern gibt es mehrere Möglichkeiten. Die einfachste ist das mehrfache Senden der Nachricht mit anschließendem Mitteln, außerdem gibt es Fehlerkorrekturverfahren, wie z.B. den Hamming-Code. Für diesen entschieden wir uns auch, da er relativ einfach zu implementieren sein sollte. Leider stellte sich heraus, dass Korrekturverfahren sehr empfindlich auf schlechte Synchronisation reagieren, weswegen es uns bisher nicht gelang, eine Nachricht mit Fehlerkorrektur zu übertragen. Eine mögliche Lösung ist die Verwendung eines externen Triggersignales oder die Implementierung verbesserter Synchronisationssignale.

Unser nächstes Ziel wird es dann sein, eine Audiodatei in eine Bitfolge zu konvertieren und diese dann durch den Versuchsaufbau zu schicken, was auf Grund der höheren Redundanz sicherer gehen sollte als eine ASCII-Textnachricht.

5. Ergebnisse

Computersimulation eines aperiodischen Signals:

Amplitude: 0,5
Frequenz: 3
ST-Schwelle: 1
Standartabweichung des Rauschens: 1,5-8

Die Resonanzkurve (Abbildung 5) zeigt eine optimale Rauschintensität bei einer Standartabweichung von 4,6 mit einem Wert von 0,95 an

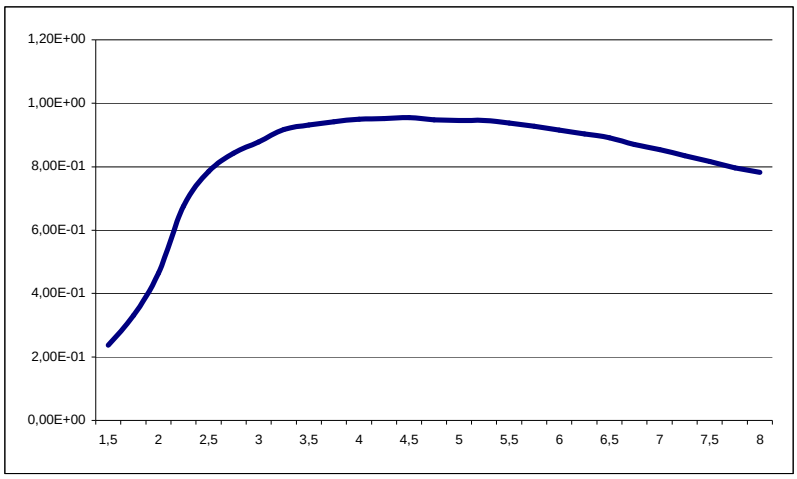


Abbildung 5: Resonanzkurve (X-Achse: Rauschintensität; Y-Achse: Kreutzkorrelationsfunktion

Experimentelle Messung eines aperiodischen Signals:

F=100kHz
Amplitude: 200mV
Rauschen: 100 - 1000 mV_{RMS}

Die optimale Rauschintensität lag bei 300 mV_{RMS} bei einer Kreuzkorrelation von 0,76

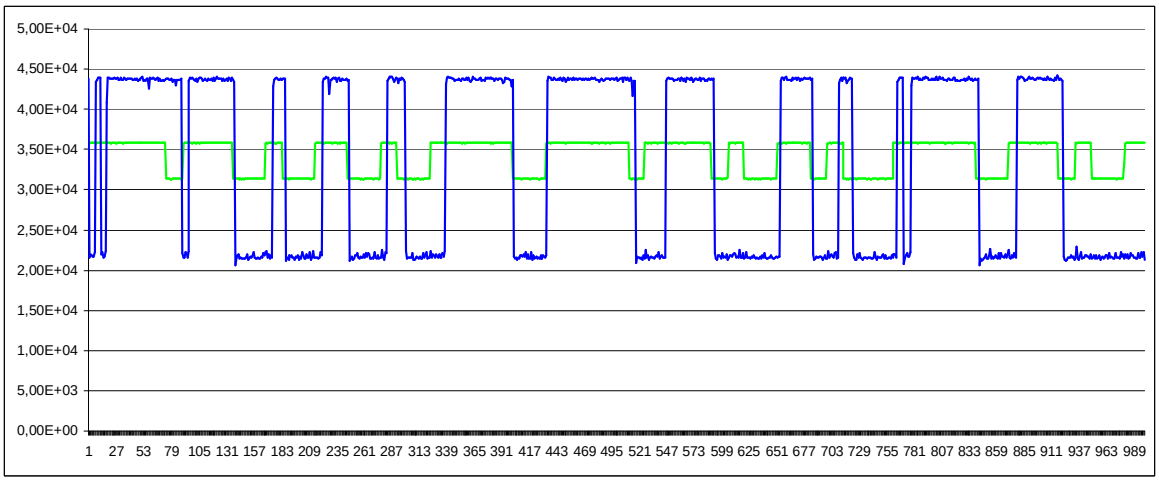


Abbildung 6: Grün: Eingangssignal; Blau: Ausgangssignal

6. Diskussion

Das von uns verwendete System zeigt schon jetzt sehr gute Ergebnisse, welche durch bessere Bauteile noch weiter verbessert werden könnte. Die Anwendungsgebiete dieser Technik sind sehr vielfältig: überall wo Signale übertragen werden, können diese mit Hilfe stochastischer Resonanz verstärkt werden. Dies ist besonders bei schwachen Signalen, wie z.B. bei Satelliten hilfreich. Natürlich wirkt stochastische Resonanz auch wie ein Rauschfilter und kann daher auch in diesem Gebiet eingesetzt werden. Hierbei müsste man natürlich den Schwellwert des Schmitt-Triggers an das Rauschen anpassen und nicht umgekehrt.
Ein weiterer großer Vorteil ist die Energieeinsparung. Die Akkulaufzeit von Handys könnte beispielsweise erhöht werden, da schwächere Sendesignale möglich wären, die logischerweise weniger Energie verbrauchen.
Auch in der Kryptographie könnte stochastische Resonanz benutzt werden. Zwar könnte jeder die verrauschten Signale mit etwas Aufwand relativ einfach entschlüsseln, jedoch ist schwierig das Signal überhaupt als solches zu erkennen (und nicht als sinnlosen Rauschen). In Kombination mit einem guten Verschlüsselungsverfahren könnten so Daten extrem sicher übermittelt werden, da das Signal zusätzlich zu der Verschlüsselung auch noch in einem starkem Rauschen verborgen ist.

Bis jetzt kann unser System nur digitale Signale verarbeiten. Allerdings könnte man einen AD-Wandler in die Platine integrieren und somit auch analoge Signale verstärken. Die Verstärkerschaltung mit AD-DA-Wandler benötigt wahrscheinlich eine Zeitbasis, die wir mit Hilfe eines parallelen Kanals übertragen würden.

7. Danksagungen

An dieser Stelle möchten wir uns besonders bei Johannes Werner aus der Arbeitsgruppe von Prof. Benner bedanken, dass er sich für unsere Ideen begeistern konnte und uns bei unseren Forschungsvorhaben unterstützte. Beim Chaos-Labor der TU Darmstadt bedanken wir uns für die Bereitstellung der Geräte und Bauteile. Weiterer Dank gilt Milan Dlabal für die Unterstützung und Organisation dieses Projektes.

8. Literaturverzeichnis

Phys. Rev. A 39, 4854 - 4869 (1989), B. Mc Namara, K. Wiesenfeld "The Theory of Stochastic Resonance"

Rev. Mod. Phys. 70, 223 - 287 (1998), L. Gammaitoni, P. Hänggi, P. Jung, F. Marchesoni "Stochastic Resonance"

9. Anhang

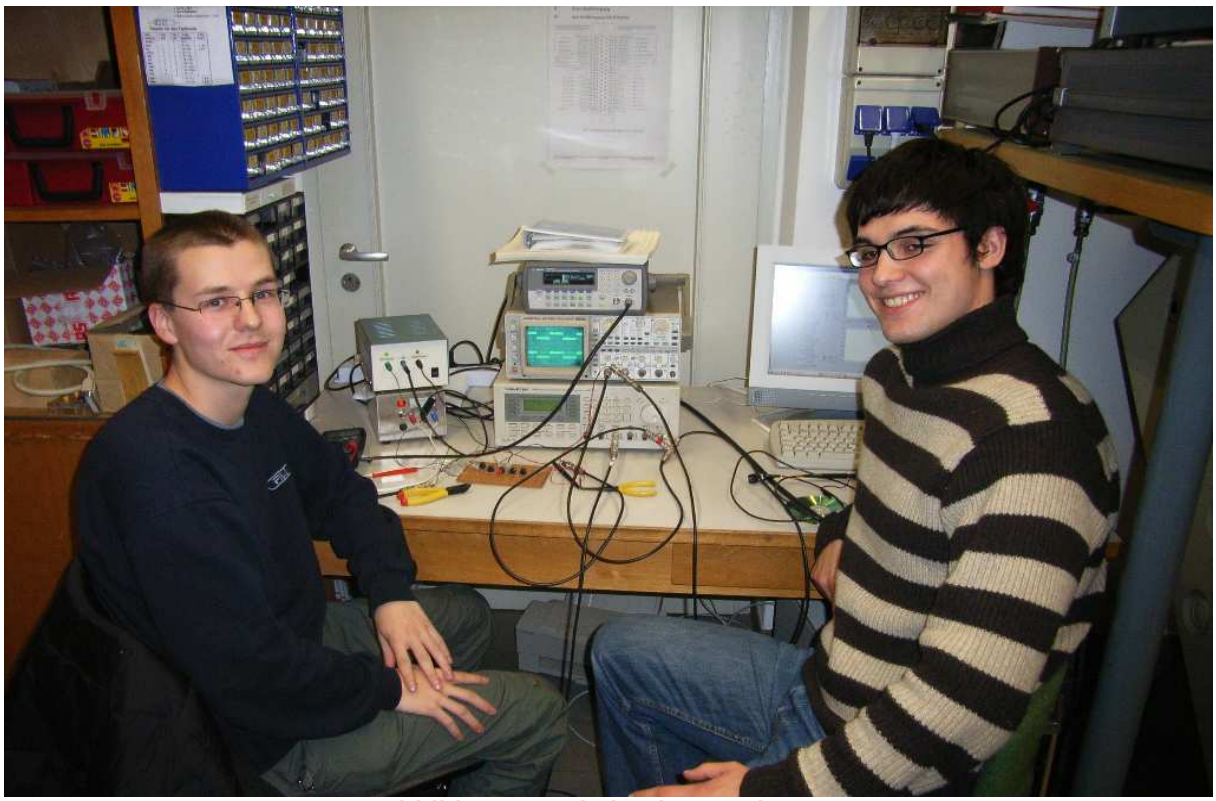


Abbildung 7: Arbeitsplatz an der TU

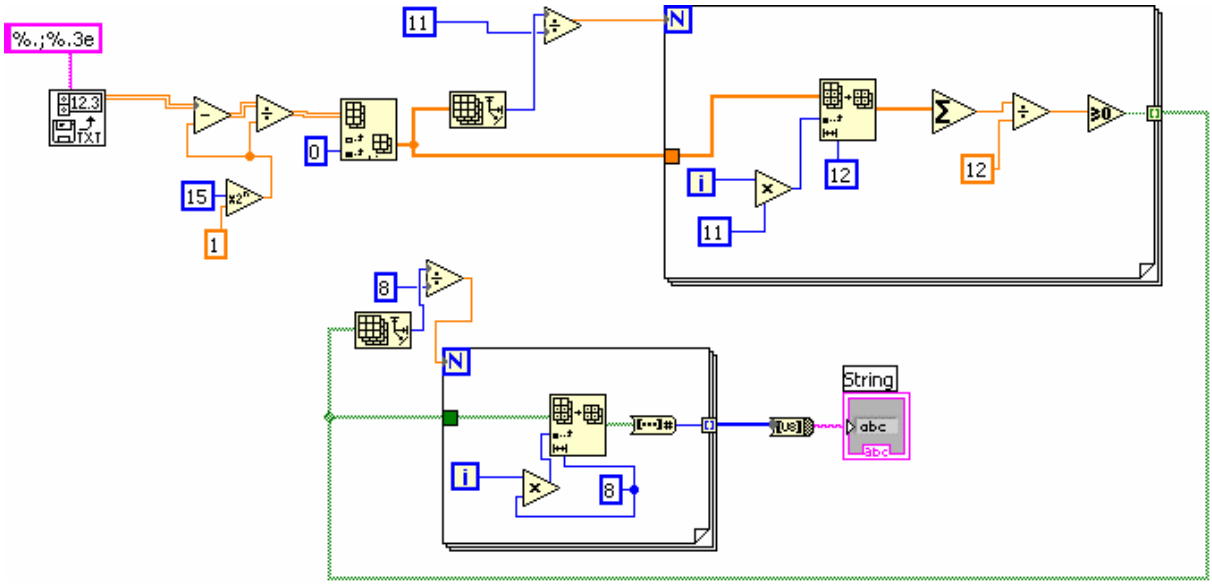
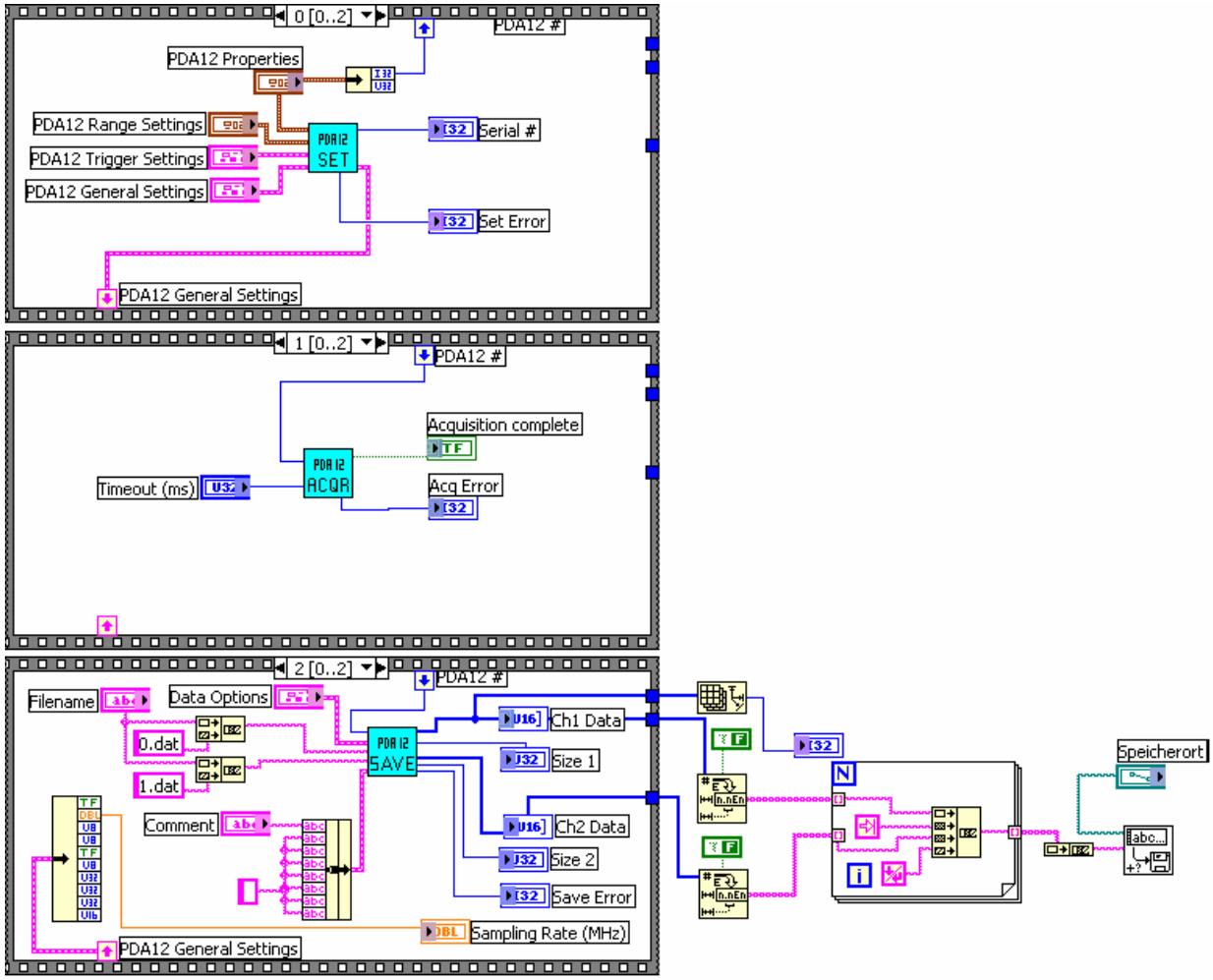


Abbildung 8: Quellcode des Programms „Bitkorrektur“ (vereinfachte Darstellung)



A

Abbildung 9: Quellcodes des Messprogramms

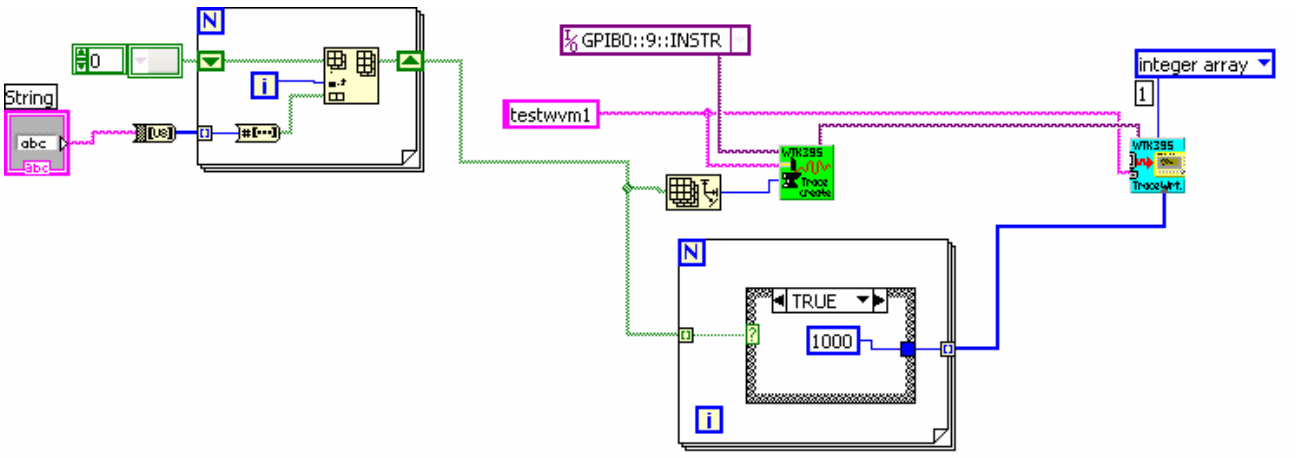


Abbildung 10: Quellcode des Programms „Textkonvertierer“ (vereinfachte Darstellung)

Quellcode des Programms „SR-Sim“ :

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, Spin;

type
  TForm1 = class(TForm)
    eAmp: TEdit;
    eFrq: TEdit;
    eSigma: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Button1: TButton;
    rgSignal: TRadioGroup;
    rgRauschen: TRadioGroup;
    eSchwelle: TEdit;
    Label4: TLabel;
    Button2: TButton;
    Label5: TLabel;
    eN: TSpinEdit;
    eDelta: TEdit;
    Label6: TLabel;
    Button3: TButton;
    eBT: TEdit;
    Label7: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  Form1: TForm1;

implementation

uses Unit2;

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
close;
end;

procedure TForm1.Button2Click(Sender: TObject);
var slSignal1, slSignal2, slRauschen, slGesamt: TStringList;
a, b, x, y, r, d, t: real;
sigma, amp, frq, st, omega: real;
n, i, z,  bt: integer;
begin
slSignal1:=TStringList.Create;  //Eingangssignal
slSignal2:=TStringList.Create;  //Ausgangssignal
```

```
slRauschen:=TStringList.Create; //Rauschen
slGesamt:=TStringList.Create;   //Ein- und Ausgangssignal
sigma:=strtofloat(eSigma.text); //Standardabweichung
amp:=strtofloat(eAmp.text);     //Amplitude
frq:=strtofloat(eFrq.text);     //Frequenz
st:=strtofloat(eSchwelle.text); //ST-Schwelle
omega:=2*pi*frq;
n:=eN.value;                    //Anzahl der Durchläufe
d:=strtofloat(eDelta.text);     //Delta t
bt:=strtoint(eBT.text);
z:=1;
Randomize;
if d>=1 then
application.messagebox('Delta t sollte kleiner 1 sein','Warnung',0)
else
begin
if rgSignal.ItemIndex=0 then //periodisches Signal
 begin
   if rgRauschen.ItemIndex=1 then //periodisches Signal ohne Rauschen
     begin
       r:=0;
       for i:=1 to n do
         begin
           t:=i*d;
           x:=amp*sin(omega*t); // Sinusfunktion
           y:=r+X; //verrauschtes Signal
           if z<0 then
             begin
               if y>st then
                 begin
                   z:=1;
                 end;
              end
           else
             begin
               if y<-st then
                 begin
                   z:=-1;
                 end;
             end;
           slSignal1.Add(floattostr(x));
           slSignal2.Add(floattostr(z));
           slRauschen.Add(floattostr(r));
         end;
     end
   else //periodisches Signal mit Rauschen
     begin
       for i:=1 to n do
         begin
           a:=Random;
           b:=Random;
           t:=i*d;
           x:=amp*sin(omega*t); // Sinusfunktion
           r:=sigma{*(sqrt(d))}*((sqrt(-ln(a))*2.3)*(sqrt(2))*(cos(2*pi*b))); //gaußverteiltes Rauschen
           y:=r+X; //verrauschtes Signal
           if z<0 then
             begin
               if y>st then
                 begin
                   z:=1;
                 end;
               end
```

```
          else
            begin
              if y<-st then
                begin
                  z:=-1;
                end;
            end;
          slSignal1.Add(floattostr(x));
          slSignal2.Add(floattostr(z));
          slRauschen.Add(floattostr(r));
          slGesamt.Add(floattostr(x)+' '+floattostr(z));
        end;
    end;
  end
else //aperiodisches Signal
  begin
    if rgRauschen.ItemIndex=1 then //aperiodisches Signal ohne Rauschen
      begin
        for i:=1 to n do
          begin
            a:=Random;
            if a>0.5 then
              begin
                x:=1;
                slSignal1.Add(floattostr(x*Amp));
                slSignal2.Add(floattostr(x*Amp));
              end
            else
              begin
                x:=-1;
                slSignal1.Add(floattostr(x*Amp));
                slSignal2.Add(floattostr(x*Amp));
              end;
          end;
      end
    else //aperiodisches Signal mit Rauschen
      begin
        for i:=1 to n do
          begin
            a:=Random;
            b:=Random;
            t:=i*d;
            if (trunc(t/bt)-t/bt=0) then
            begin
            if a>0.5 then
              begin
                x:=1;
              end
            else
              begin
                x:=-1;
              end;
            end;
            a:=random;
            r:=sigma*sqrt(d)*((sqrt(-ln(a)))*(sqrt(2))*(cos(2*pi*b))); //gaußverteiltes Rauschen
            y:=r+x*Amp; //verrauschtes Signal
            if z<0 then
              begin
                if y>st then
                  begin
                    z:=1;
                  end;
```

```
          end
        else
          begin
            if y<-st then
              begin
                z:=-1;
              end;
          end;
        slSignal1.Add(floattostr(x*Amp));
        slSignal2.Add(floattostr(z));
        slRauschen.Add(floattostr(r));
        slGesamt.Add(floattostr(x*Amp)+' '+floattostr(z));
      end;
  end;



end;
end;
slSignal1.SaveToFile('c:\signal1.txt');
slSignal2.SaveToFile('c:\signal2.txt');
slRauschen.SaveToFile('c:\Rauschen.txt');
slGesamt.SaveToFile('c:\Gesamt.txt');

application.messagebox('Simulation erfolgreich','SR-Simulation',0);
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
form1.visible:=false;
form2.visible:=true;
end;

end.

  Form2: TForm2;

implementation

uses Unit1;

{$R *.DFM}


procedure TForm2.Button2Click(Sender: TObject);
begin
form2.visible:=false;
form1.visible:=true;
end;

procedure TForm2.Button3Click(Sender: TObject);
var
n: integer;
sl1, sl2, slr, sl3: TStringList;
begin

Chart1.Series[0].clear;
Chart1.Series[1].clear;
Chart1.Series[2].clear;
Chart1.Series[3].clear;
```

```
if cb1.checked=true then
  begin
    sl1:=TStringList.Create;
    sl1.LoadFromFile('c:\signal1.txt');
      begin
        for n:=0 to sl1.count-1 do
          begin
            Chart1.Series[0].AddXY(n,strtofloat(sl1[n]),'',clTeeColor);
          end;
      end;
  end;

if cb2.checked=true then
  begin
    slr:=TStringList.Create;
    slr.LoadFromFile('c:\rauschen.txt');
      begin
        for n:=0 to slr.count-1 do
          begin
            Chart1.Series[1].AddXY(n,strtofloat(slr[n]),'',clTeeColor);
          end;
      end;
  end;

if cb3.checked=true then
  begin
    sl1:=TStringList.Create;
    sl1.LoadFromFile('c:\signal1.txt');
    slr:=TStringList.Create;
    slr.LoadFromFile('c:\rauschen.txt');
      begin
        for n:=0 to sl1.count-1 do
          begin
            Chart1.Series[2].AddXY(n,(strtofloat(slr[n]))+(strtofloat(sl1[n])),'',clTeeColor);
          end;
      end;
  end;

if cb4.checked=true then
  begin
    sl2:=TStringList.Create;
    sl2.LoadFromFile('c:\signal2.txt');
      begin
        for n:=0 to sl2.count-1 do
          begin
            Chart1.Series[3].AddXY(n,strtofloat(sl2[n]),'',clTeeColor);
          end;
      end;
  end;

end;
end.
```