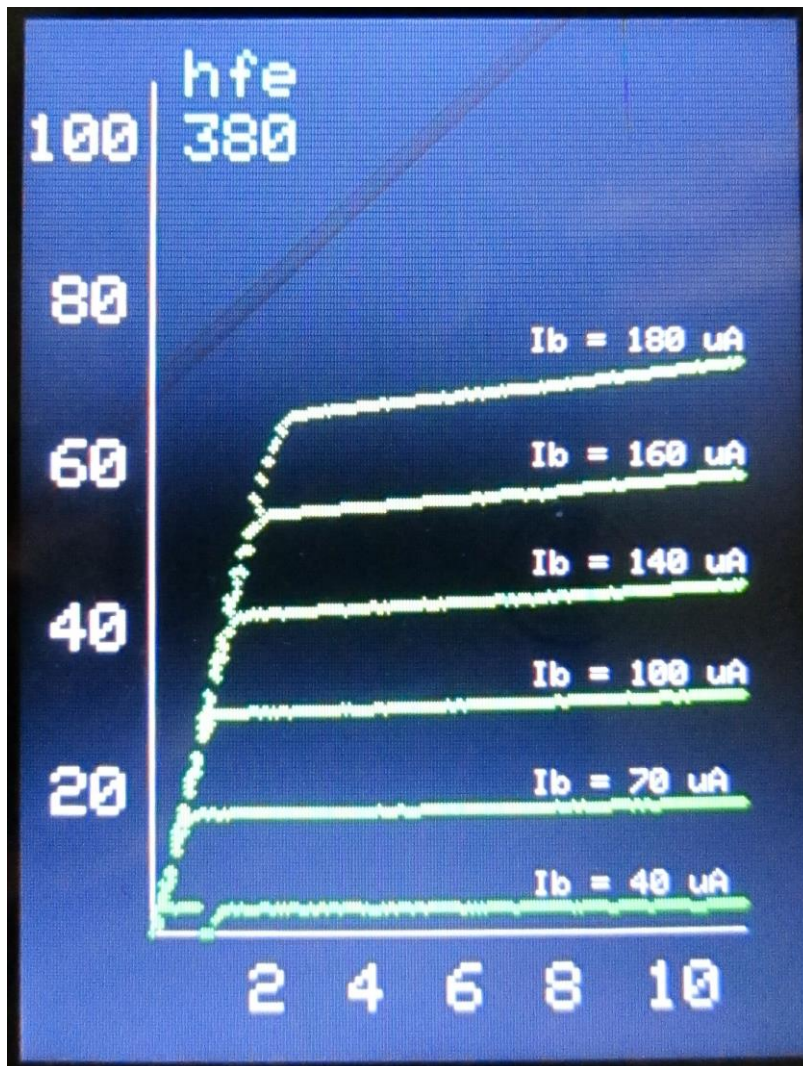


NPN-Analyzer

Ein Kennlinienschreiber für Kleinsignaltransistoren. Nur für NPN-Transistoren und Dioden. Ein Bericht über die Vorarbeiten zu diesem Projekt findet man hier:

<https://berndg42.de/funk/kennlinienschreiber-dok.pdf>



Das Foto zeigt nicht die wirklichen Farben. Der Hintergrund ist schwarz und die Kennlinien grün.

Bauvorschläge für so ein Prüfgerät gibt es einige im Netz. Ich wollte eine Ausführung, die ohne den Oszillografen auskommt. Diese lästigen Verbindungen mit Koaxialkabeln entfallen bei unserem Teil. Und man muss auch nicht die Kurven auf dem Schirm interpretieren. Manchmal ist es sogar nicht so einfach, den X/Y-Betrieb am Oszillografen einzustellen. Ich suchte lange danach, bis ich auf meinem Siglent SDS 2102 X Plus den Modus entdeckte. Das Manual schweigt sich dazu aus. Selbst eine Anfrage bei ChatGpt brachte nicht den Erfolg. Schließlich fand ich die Betriebsart unter „Erfassen“. Keine zielführende Übersetzung ins Deutsche!

Der Reiz des Bastelns ist hier auch in der durchschaubaren Anwendung eines Mikroprozessors zu finden. Und eigentlich wollte ich ganz ohne Bibliotheken auskommen, die man nach einigen Jahren oft nicht mehr findet. Doch der Einsatz eines Farbdisplays ist leider an eine solche von Adafruit gebunden. Am besten sichert man sich diese noch heute.

Ebenso werden wir keine speziellen ICs verwenden. Alle Bauteile sind Standardware und leicht zu beschaffen.

So ein Prüfgerät sollte nicht viel Platz im Lager beanspruchen. Ein Netzbetrieb ist meistens umständlich, weil allein das Netzkabel mit seinem Gewicht ein leichtes Gehäuse vom Tisch zieht. Auch darüber ärgerte ich mich schon oft und ich habe deshalb einen speziellen Batteriebetrieb vorgesehen. Über USB zu einem Powerpack wird die Schaltung versorgt. Leider verlangt der Einsatz eines Operationsverstärkers eine negative Betriebsspannung. Wir erzeugen sie mit einem selbst angefertigten DC/DC-Wandler. Der ist mit einer kleinen

ansteigen. Die Abfrage des Kollektorstromwertes geschieht synchron. Diese yval-Werte bestimmen die Position des Punktes, den wir in Richtung y zeichnen. Die x-Position ergibt sich aus dem augenblicklichen Zählerstand. Aus dem yval-Wert in der dritten Kennlinie ermitteln wir den Kollektorstrom. Mit dem zugeordneten Basisstrom wird die Stromverstärkung hfe errechnet. Sie wird dann oben links erscheinen. Leider sind die Kollektorströme für die Analyse an einem Leistungstransistor zu gering. Aber man erkennt die ordentliche Funktion, wenn Kennlinien gezeichnet werden. Das Erscheinen eines hfe-Wertes beim Test an einer Diode ist irreführend. Sollte man ignorieren. Ich habe nicht vorgesehen, diese falsche Date zu löschen. Denn dazu müsste man zusätzlich einen Schalter für diese Betriebsart einbauen und in der Software abfragen.

Fazit

Es handelt sich also um einen Kennlinienschreiber, der die Grundfunktion erfüllt. Das Konzept wird bewusst einfach gehalten, damit ein Nachbau ohne größeren Aufwand sicher funktioniert. Die Steuerung mit einem Mikroprozessor ist nicht zu kompliziert. Der Anwender kann den Ablauf in der Software ohne Verständnisprobleme nachvollziehen. Bei Bedarf wird das Gerät an eine Powerbox(5V) über ein USB-Kabel angeschlossen. Alle Betriebsspannungen werden intern mit DC/DC-Wandlern erzeugt.

Der Analysator kann nicht die Pins eines Halbleiters zuordnen. Man muss also zuvor E,B und C selbst ermitteln. Dafür gibt es aber preiswerte Prüfgeräte aus China. Z.B. ***LCR-T4 Mega328 Transistor Tester für ca. 15 EUR.***

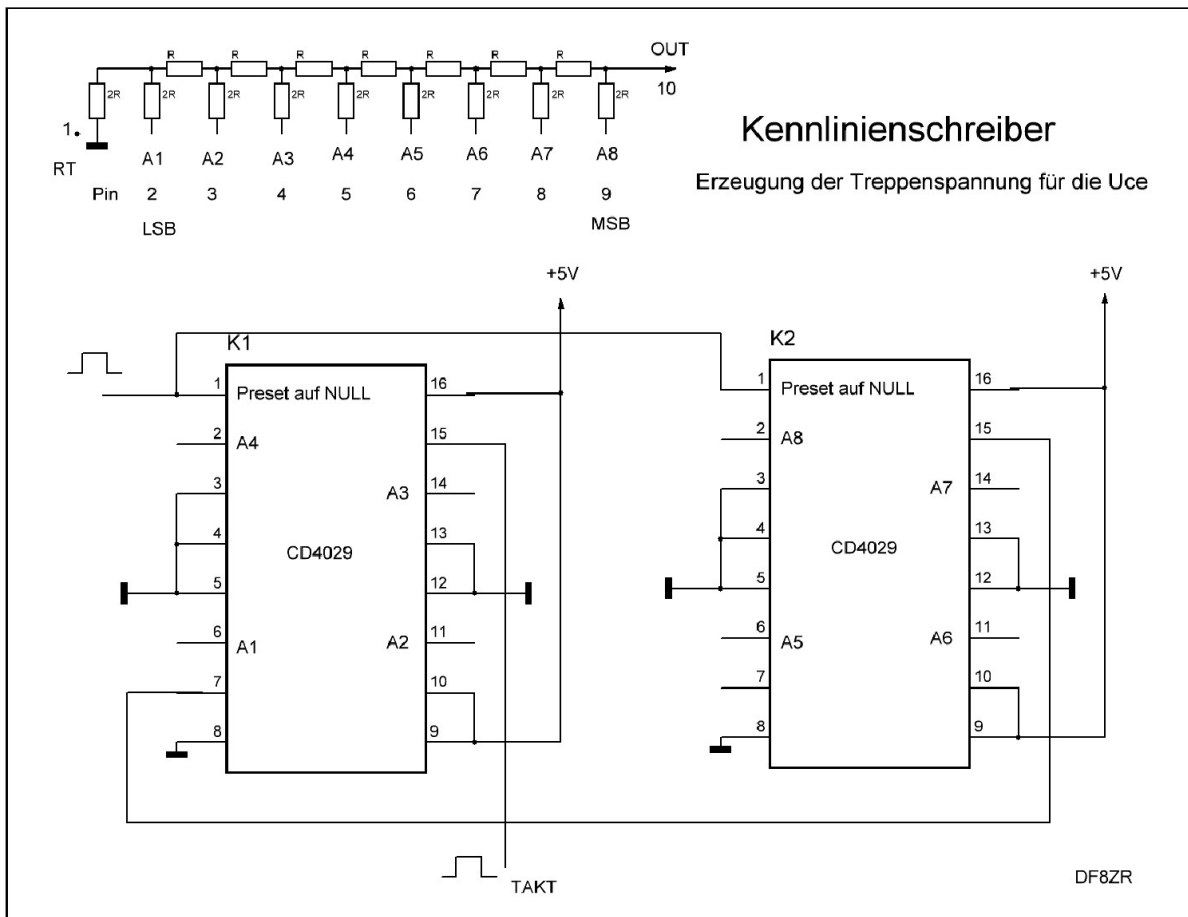
Erweiterung

Eine wesentliche Erweiterung wird vorgestellt. Es ist die Möglichkeit, Kennlinien von zwei Transistoren zu vergleichen. Nützlich, wenn man Dioden auf gleiche Fluss-Spannung oder Transistoren mit gleichen Eigenschaften aus einer größeren Menge sortieren muss. Auf die Umschaltung vom A auf den B-Modus wird später eingegangen. Diese Erweiterung ist kein Muss und man kann einen festen Modus leicht in der Software einstellen.

90% aller Messungen erfolgen erfahrungsgemäß an NPN-Kleintransistoren. Wer mehr Funktionalität möchte, kann sich an den Hinweisen orientieren, die in dem o.g. Entwicklungs-Bericht zu finden sind.

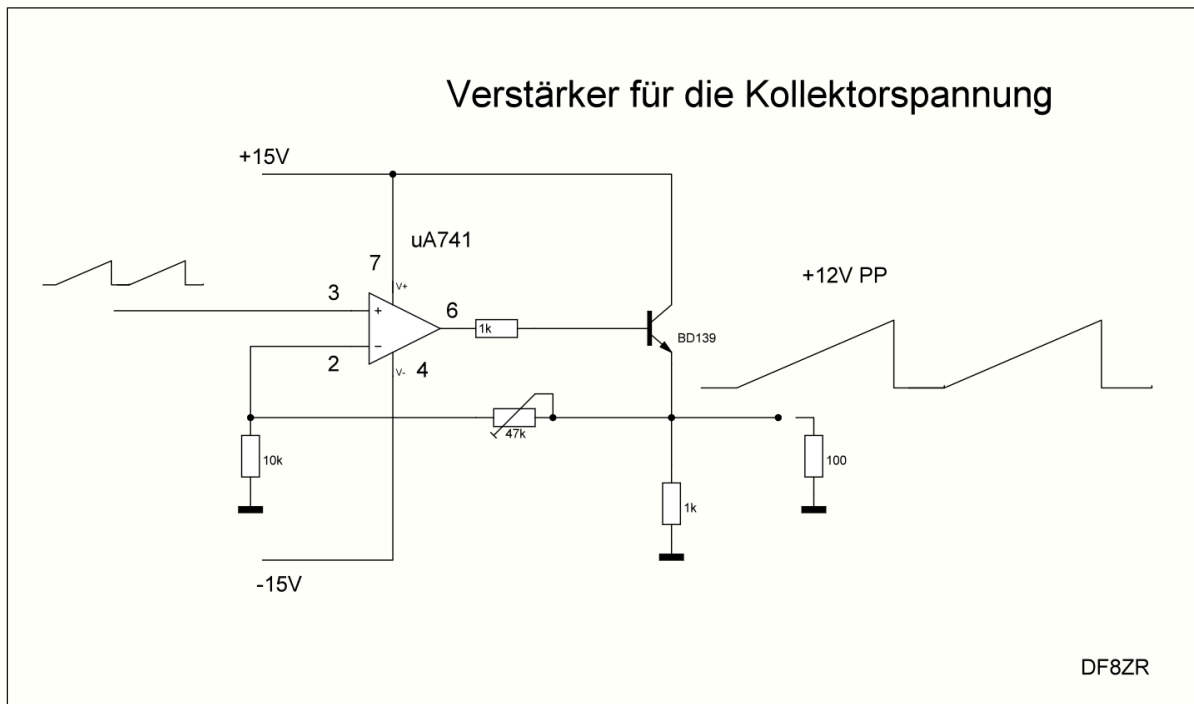
Erzeugung der Sägezahnspannung für Uce

Wir bauen einen 8-Bit-Zähler mit den CD4029. Gestartet und gestoppt wird er mit einem Impuls am Pin 1. Der setzt den Zähler auf den Anfangswert NULL. Der Takt kommt an 15 und wird im Arduino erzeugt. Die Zählerbausteine arbeiten mit +5V-Versorgung. Sie nehmen nicht viel Strom auf. Das R2R-Netzwerk kann man mit 22k(1%)-Widerständen basteln. $R = 11k$ erzeugt man mit der Parallelschaltung von zwei 22k.



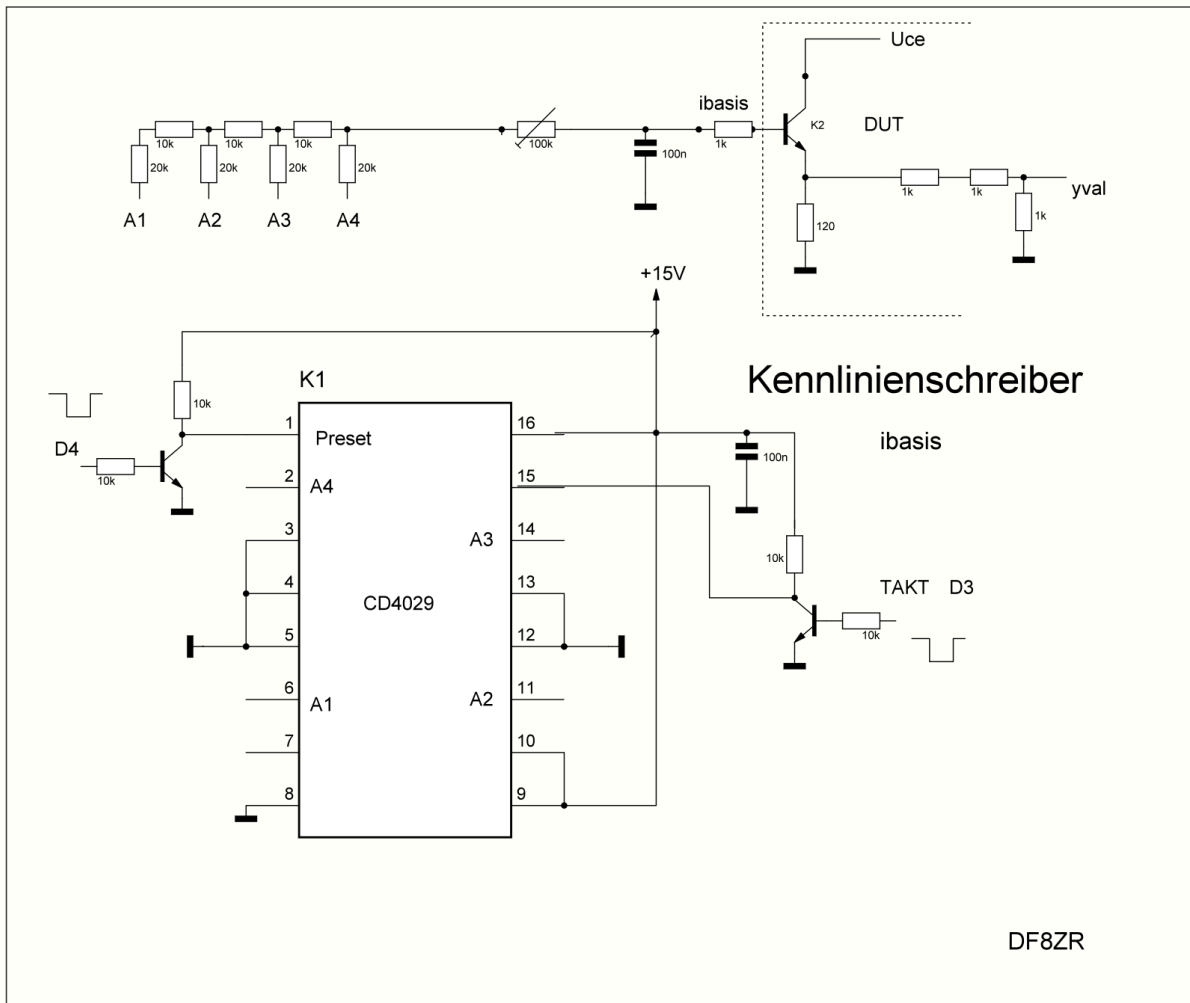
Manchmal werden R2R-Netzwerke bei Ebay angeboten. Da das Netzwerk hochohmig ist und nicht den hohen Kollektorstrom liefern kann, muss der Sägezahn verstärkt werden. Das machen wir mit einem OP und einem Leistungstransistor BD139. Der muss aber nicht besonders gekühlt werden. Immerhin ist der höchste Kollektorstrom 100 mA. Er wird durch den Messwiderstand am Emitter (120R) bestimmt. Die Kollektorspannung wird +12V nicht überschreiten. Die meisten OPs haben eine Obergrenze der Versorgung von +/- 15V. Ich habe bewusst den uA741 gewählt, weil man den bedenkenlos bis +/- 20V Versorgung verwenden kann. Und er ist bei +12V Spitzenspannung noch nicht in der Begrenzung. Um diese zu verhindern, wird er mit +/- 15V betrieben.

Uce-Verstärker



Die Erzeugung der Basis-Ströme

Sie wird ebenfalls mit einem selbst gebastelten DAC gemacht. Hier brauchen wir eigentlich nur 3 Bits, denn es werden max. 6 Stufenspannungen erzeugt. In der Schaltung sind aber 4 Bits vorgesehen. Bei Bedarf könnte man mit den restlichen 9 Stufen noch Gatespannungen für den Test am MOSFET erzeugen. Man kann aber auch die letzten beiden Widerstände im R2R-Netzwerk weglassen. Das wurde hier mit 10k(1%)-Widerständen gebaut. Der Zähler wird vom Arduino gesteuert. Und er wird deshalb mit +15V versorgt, weil dann die Spannungen am Ausgang des Netzwerkes hoch genug sind, um über den 100k-Trimмер die max. Basisströme zu liefern. Mit 5V-Versorgung wären die zu gering.

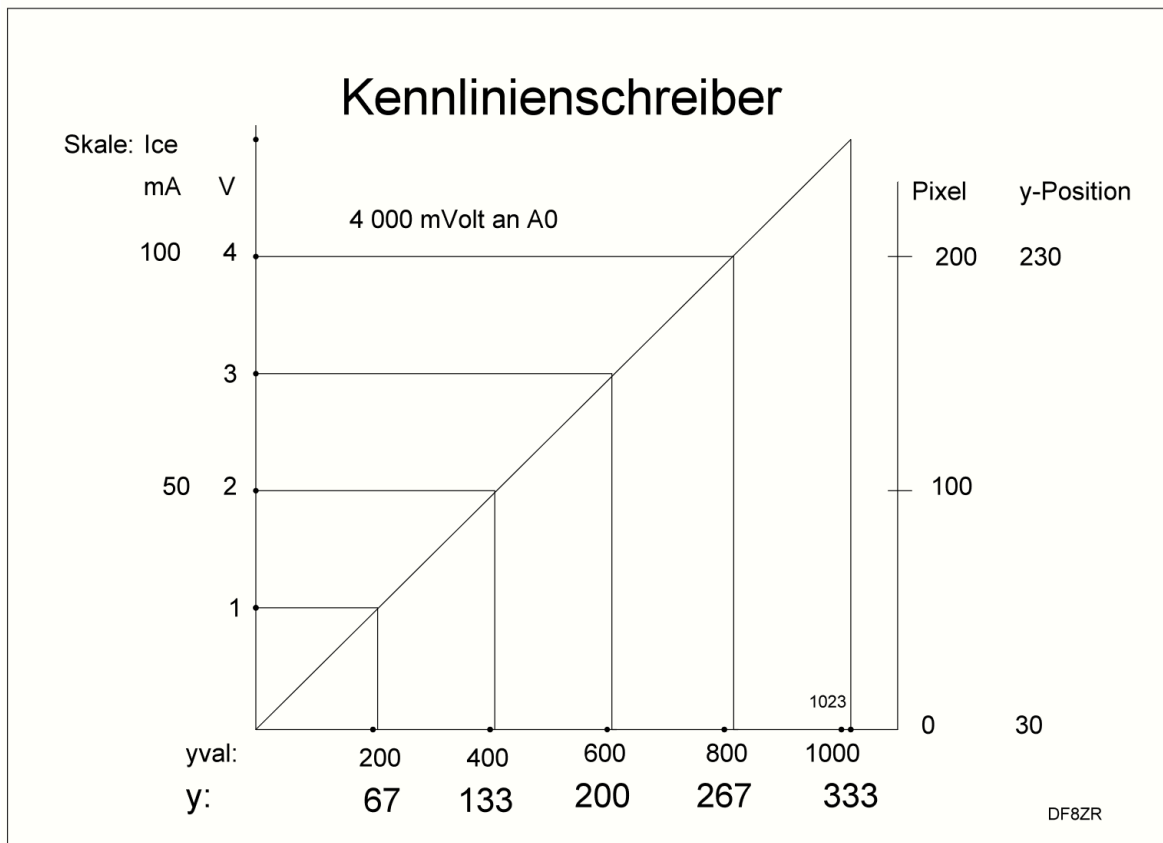


Die beiden Transistoren am CD4029 machen die Pegelanpassung zum Arduino. Ich habe so einen weiteren OP eingespart.

Abnahme der yval-Spannungen

Im Bild sieht man oben rechts den Messwiderstand (120 Ohm) am DUT. An ihm können max. 12V anstehen. Die würden den Analogeingang am Arduino event. zerstören. Daher wird eine Spannungsteilung gemacht. Mit der Reihenschaltung von drei 1k werden die +12V auf max. +4V begrenzt. Der Analogeingang A0 verträgt +5V. Und er löst dabei bis zu 1023 Werte auf. Die nach der Analog-Digital-Umsetzung im

Arduino vorliegenden Werte sind Integer. Ich habe willkürlich die Umsetzung bei +4V auf 100 mA gesetzt. Ein Kollektorstrom von 100 mA stellt sich also mit einem Umrechnungsfaktor von $1023/5$ dar. Das folgende Diagramm gibt eine Orientierung für die Berechnungen.

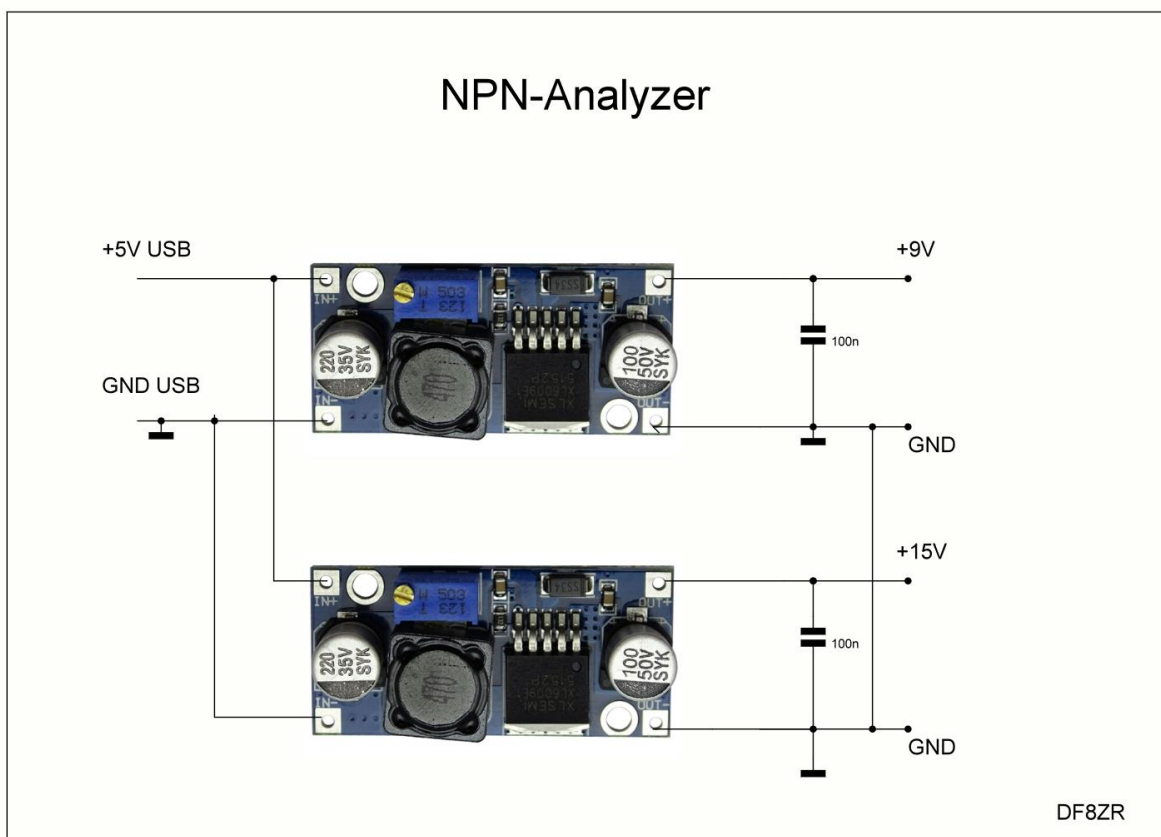


Ganz unten sehen wir die Werte von $yval(y)$. Für die Berechnung des hfe muss man aber mit Gleitkommazahlen(float) rechnen, denn sonst erhält man bei der Division von Integern NULL. Die y-Achse hat 200 Pixel, die x-Achse ebenfalls. Die y-Achse ist durch die Drehung bedingt in umgekehrter Zählweise zu programmieren. Das macht die Software an dieser Stelle etwas undurchsichtig. Der Fortschritt von y geht von oben nach unten. Er wird durch eine Differenz bestimmt.

Die Erzeugung der Versorgungsspannungen

Der Arduino wird mit einer externen Versorgung von +9V betrieben. Intern setzt er dann diese auf +5V um. Für die Uce brauchen wir eine Spannung von +15V. Dieser Quelle wird ein größerer Strom von $> 100\text{mA}$ entnommen. Und zuletzt wird noch eine schwächere Stromquelle mit -15V benötigt, um den OP zu betreiben. Der soll nämlich von NULL Volt aufwärts die Uce bestimmen.

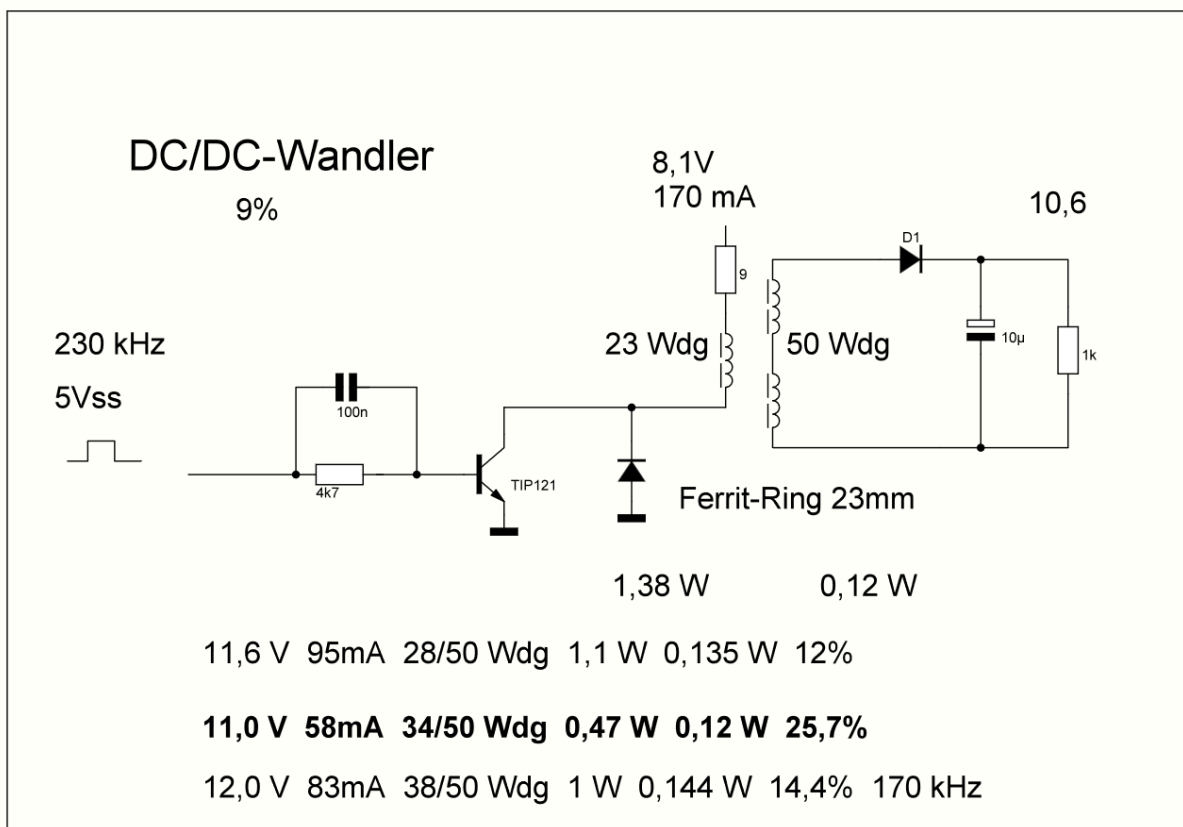
Es werden für +9V und +15V jeweils fertige Step-Up-Converter eingesetzt. Die sind preiswert aus chinesischer Herstellung zu beschaffen. Leider kann man sie nicht gleichzeitig für -15V verwenden, weil Eingang und Ausgang nicht galvanisch getrennt sind. Wir müssen hier eine Trennung der ausgangsseitigen Masse vornehmen. Das geht am besten mit einem selbst gebauten DC/DC-Wandler.



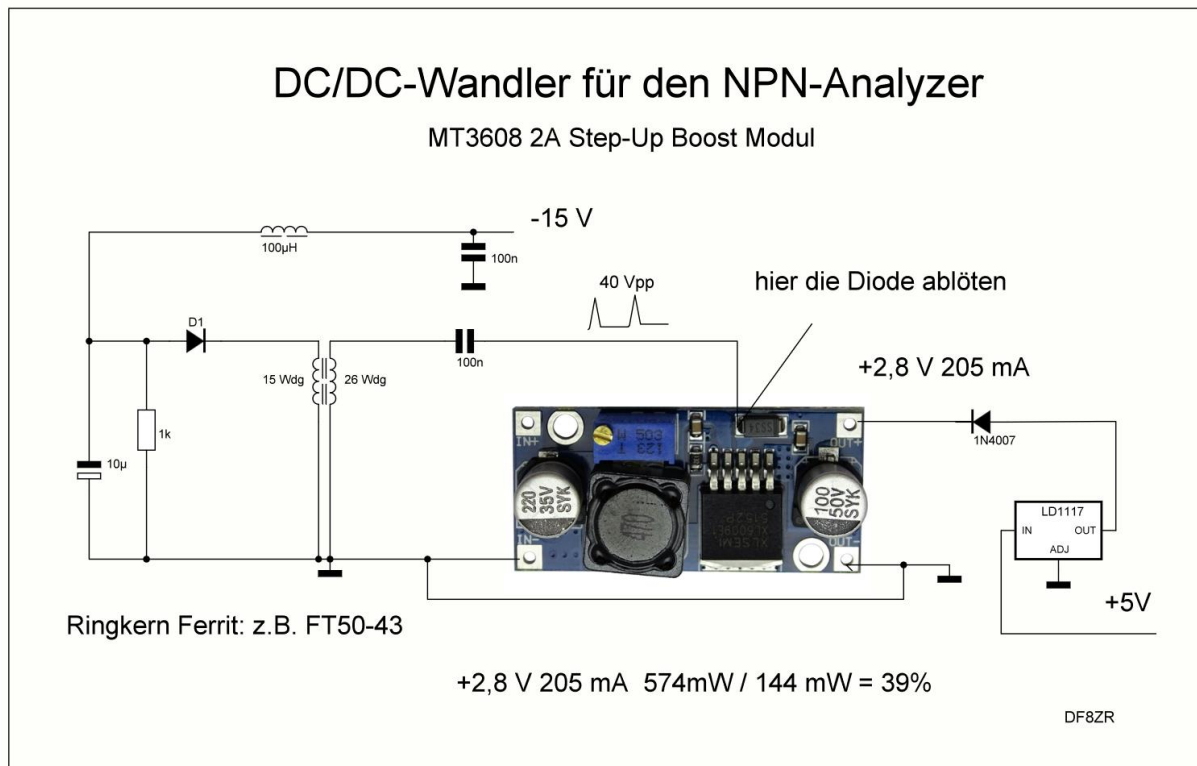
Die ersten beiden Stromversorgungen verwenden die chinesischen Platinen. Sie haben einen gemeinsamen GND. Sie müssen nicht gekühlt werden. Der Messvorgang ist einmalig und von kurzer Dauer. Nach dem Zeichnen eines Frames werden die Basisströme auf NULL gesetzt, sodass kein Kollektorstrom fließt. Im Messwiderstand wird dann auch keine Energie in Wärme umgesetzt.

Erzeugung -15V

Ich habe einige Versuche mit Ringkernen und Klappferriten gemacht. Die Ansteuerfrequenzen waren zwischen 170 kHz und 230 kHz. Leider sind die Wirkungsgrade sehr gering. Meine beste Schaltung hatte 25%. Weil alles vom Typ des Ringkernes abhängig ist, kann man keine Empfehlung für einen erfolgreichen Nachbau geben. Im folgenden Bild sieht man Resultate.



Danach setzte ich ein fertiges Modul ein.



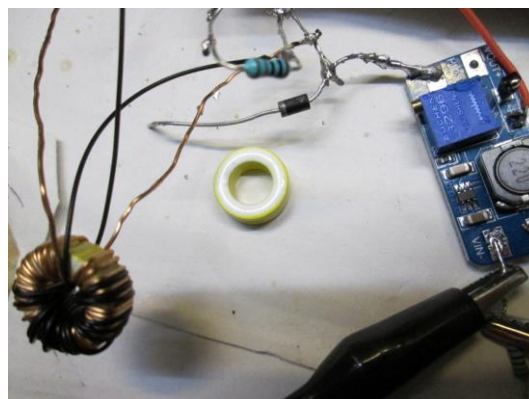
Es kostet in DL ca. 2,70 EUR. Leider haben diese Module keine galvanische Trennung zwischen dem Input und dem Output. Die besten Ergebnisse hatte ich mit einer Trennung durch einen Transformator. Das Modul gibt am Ausgang vor der Gleichrichtung Impulse mit 40Vss ab. Die Diode muss man ablöten. Eine Regelung mit dem Trimpoti nicht möglich. Durch die Transformation wird die Polarität der Impulsspannung umgekehrt. Ein spannungsfester Kondensator dient als Trennmittel zur Schaltung und dem Pluspol am Input. Lässt man den weg, entsteht ein Kurzschluss nach GND. Die Versorgungsspannung bricht zusammen. Jetzt aber haben wir eine vollständige Isolierung von der Eingangsspannung und gleichzeitig dürfen wir den GND der nachfolgenden Gleichrichterschaltung mit dem allgem. GND am Input verbinden. Die so erzielten -15 V muss man noch von Impulsresten durch eine Siebschaltung befreien. Die

Ausgangsspannung ist sehr feinfühlig mit 2,8 V von +5V einzustellen. Weil das zu Problemen führen kann, habe ich einen Spannungsstabilisator für 3,3V vorgesehen. Abzüglich der Fluss-Spannung einer 1N4007 liegen dann 2,8V am Wandler.

Leider braucht diese Art der Erzeugung einen relativ hohen Strom. Aber andere Lösungen zeigen dieselbe Problematik. Für den Betrieb des einzigen OPs ist sie aber wegen der Nullpunktlage der Uce erforderlich. Und man belastet beim Messen eines Transistors also die Powerbox. Die hat aber meistens große Reserven, sodass hier keine zu kurzen Betriebszeiten zu erwarten sind.

Ich fand preiswerte Ringkerne bei Ebay. Sie waren mit ca. 50 Wdg bewickelt. Mit dünnem Draht konnte ich sekundär leicht 15 Wdg aufbringen. Die Farbe ist gelb/weiß. Und diese Massenware wird für die EMV verwendet. Man sollte sich an die Windungszahlen halten, denn sonst sinkt der Wirkungsgrad. Der ist mit 39% akzeptabel. Vielleicht findet aber jemand eine bessere Lösung, z.B. mit NE555. Fertige Wandler kosten richtig Geld.

Versuchsaufbau:



Im Betrieb ist die Stromaufnahme von -15V nur 0,6 mA!

Ich habe also die Primärwicklung parallel zur Induktivität des Moduls geschaltet und versucht, bei etwa gleichem Wert eine Leistungsanpassung zu erreichen. Man könnte den Wirkungsgrad weiter steigern, wenn man die Induktivität des Originals abklemmt und statt dessen die Primärwicklung einfügt. Bei einem ersten Versuch ging das leider schief Die Lötunkte lösten sich. Es blieb dann eben bei der einfachen Lösung.

Man könnte selbstverständlich fertige Module für die Signalverarbeitung einsetzen. Es gibt sie z.B. bei Reichelt.

Eigentlich etwas viel Aufwand für einen Wandler, der kaum Strom liefern soll. Hier ist ein Vorschlag mit einem IC. Aber im Vergleich auch nicht weniger Bauteile. Und dann ist er vielleicht nicht einfach zu beschaffen. Meine Suche nach den kleinen Converttern, die aus +5V +15V machen, war bisher ohne Erfolg.

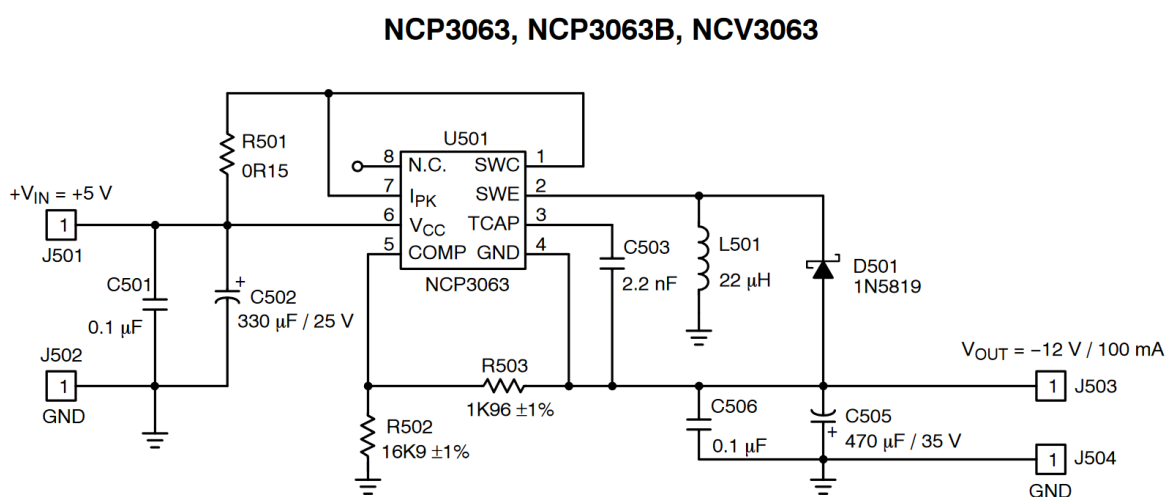


Figure 22. Typical Voltage Inverting Application Schematic

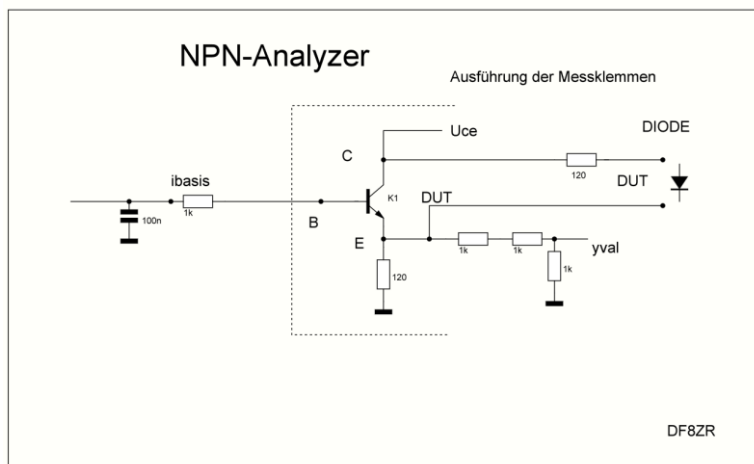
Die meisten Wandler-Module erfordern eine besondere Bestellung.

Schlussbemerkung zu den Netzteilen

+5 V kommen aus der Powerbox. Daraus machen wir +9V für den Arduino, +15V und -15V mit preiswerten Bauteilen. Die Energie wird über eine Buchse USB B eingeführt. Mit einem Standardkabel ist diese Verbindung flexibel und reißt mit dem geringen Eigengewicht den Analyzer nicht vom Tisch.

Eine Diode prüfen

Germaniumdioden vertragen max. 60 mA. Deshalb darf man sie nicht an die Klemmen für den Transistor anschließen. Ich habe zusätzlich zwei Buchsen für die Dioden vorgesehen. An der oberen Klemme wird der Strom nie > 50 mA. Allerdings muss man jetzt berücksichtigen, dass die U_{ce} bis +6V läuft. Die Werte auf der x-Achse sind beim Ablesen also zu verdoppeln. Dadurch wird die Auflösung etwa besser und man kann sehr genau die Fluss-Spannung bestimmen.



Inbetriebnahme

Einen Einschalter brauchen wir nicht. Mit dem Anstecken an die Powerbox läuft die Elektronik. Der Analyzer meldet sich mit einem Startbild. Jetzt löst ein Druck auf den Taster START die Vorgänge aus. Es wird das Achsenkreuz mit der Beschriftung gezeichnet. Dann erfolgt die Messung und die Darstellung der Kennlinien. Ich spreche in der Software von einem Frame. Ist das Bild vollständig, pausiert das Programm. Das Diagramm bleibt also für ein Foto erhalten. Mit einem erneuten Klick auf den Taster wiederholt sich alles. Die Bedienung ist also nicht schwieriger als bei einem Multimeter. So wollte ich das auch. Denn wenn man noch mehr hineinpackt, werden die Kosten den Preis eines Fertiggerätes von Peak / atlas DCA 75 Pro übersteigen. Das Teil kann zwar etwas mehr, kostet aber 145,- EUR und muss am PC oder Laptop betrieben werden. Dafür habe ich leider keinen Platz auf meinem Werk Tisch, hi.

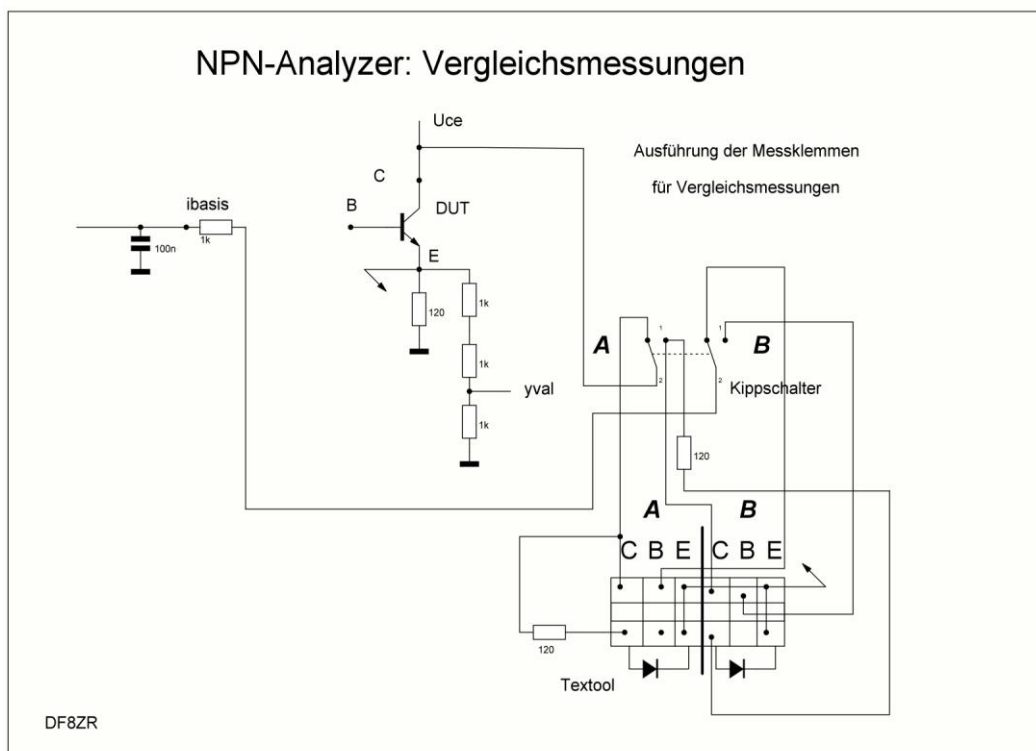
Aber die Erweiterung auf den Vergleich von Transistoren und Diode gleicher Baureihe wird beschreibe ich jetzt. Sie ist wahlweise zu realisieren und kann durch eine kleine Änderung in der Software abgeschaltet werden.

Vergleichs-Modus

Die C- und B-Anschlüsse werden mit einem zweipoligen Kippschalter den Buchsen umgeschaltet. Weiterhin verwende ich einen einfachen Textool-Sockel. Ein zweiter Taster entscheidet über den Ablauf der Messung. Mit dem Taster „Modus A“ wird zunächst der erste Transistor geprüft und

seine Kennlinien sind dauerhaft auf dem Display zu sehen. Aber gleichzeitig wird zuvor der Bildschirm gelöscht und die Achsen werden neu gezeichnet und beschriftet. Schaltet man danach auf den B-Modus um und drückt die Taste „Modus B“, dann werden nur die Kennlinien des anderen Transistors gezeichnet. Abweichungen sind so sehr gut zu erkennen. Vor dem Austausch des zweiten Transistors muss man die Taste „Clear“ drücken. Die B-Kennlinien werden dann durch Überschreiben mit der Hintergrundfarbe gelöscht. Das Display wird für die Aufnahme eines folgenden B-Transistors vorbereitet. So kann man wiederholt Vergleichstransistoren analysieren. Und man erspart sich das zwischenzeitliche Löschen des Schirms und die Neuzeichnung des kompletten Frames mit der A-Taste.

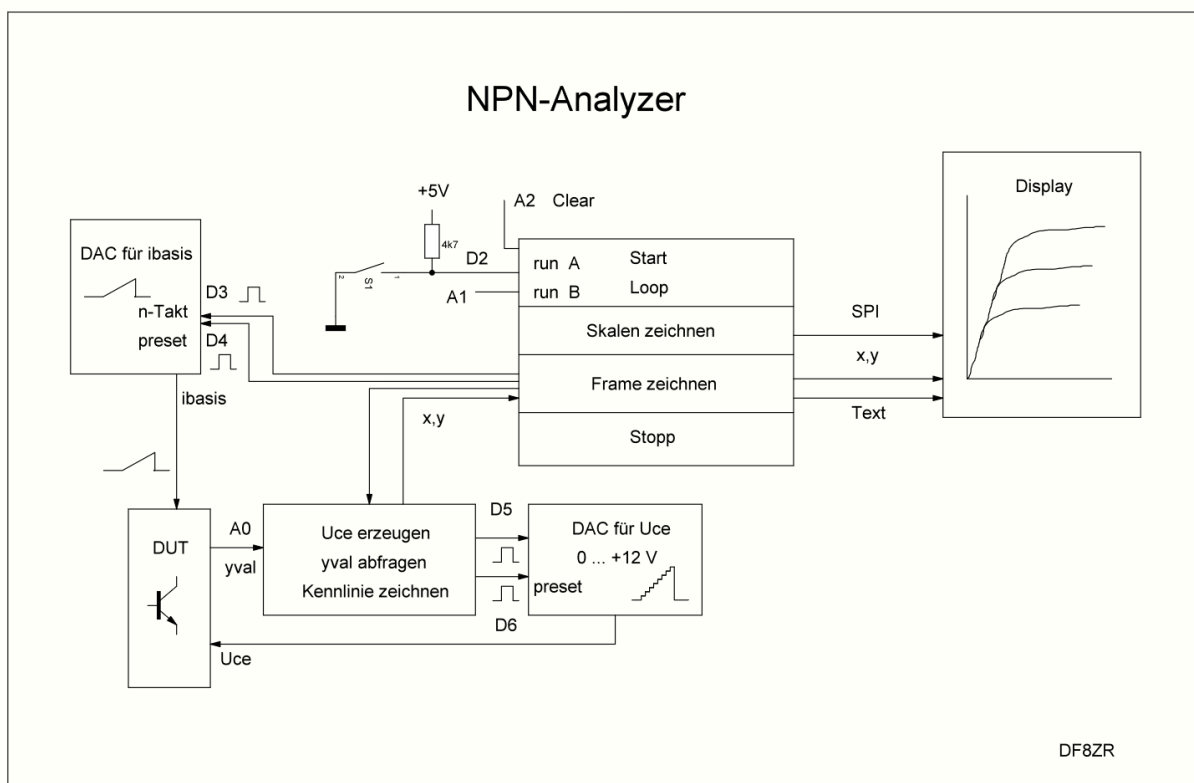
Schaltung für Vergleichsmessungen



Taster

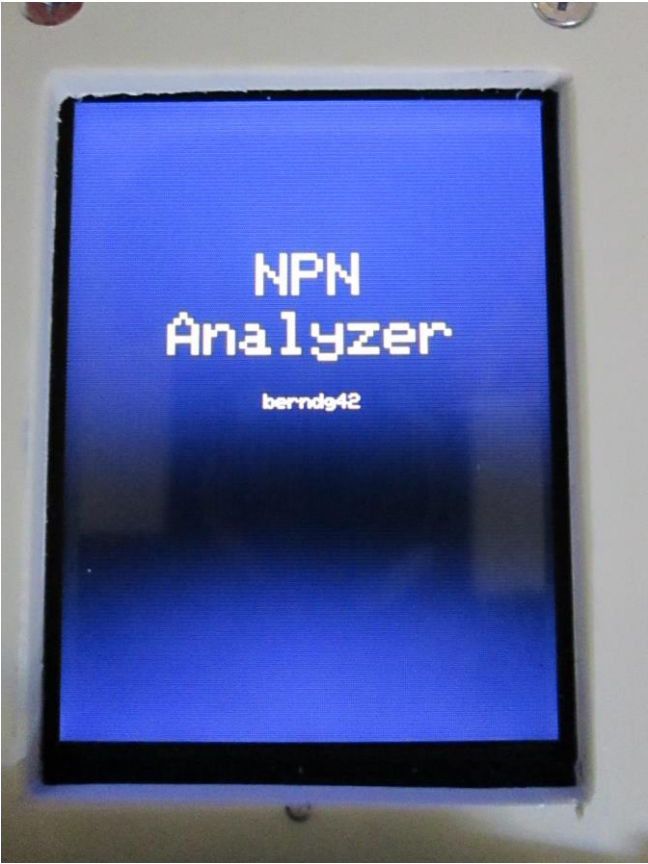
Macht man die Erweiterung, dann kommen noch zwei Taster hinzu: Modus B und Clear. Zur Abfrage habe ich in der Software die Analoginputs verwendet.

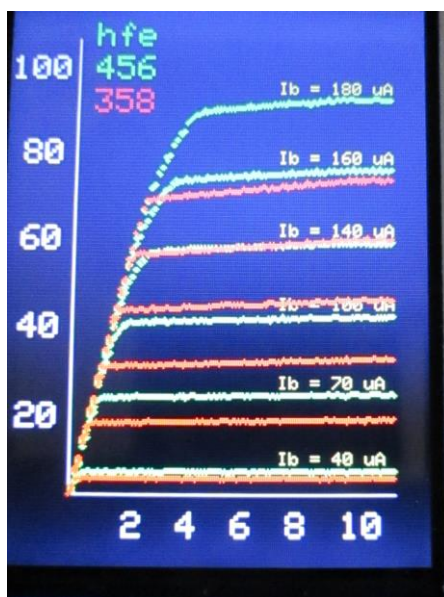
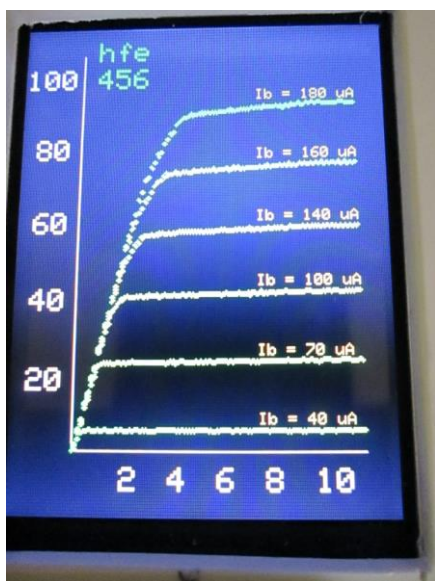
Hier der Überblick über den Systemaufbau



Viel Spielraum für die direkte Nutzung der Ports ist da nicht mehr. Dennoch könnte man noch einige Zusatzfunktionen realisieren. Man müsste dann eben die Analogports intelligent nutzen.

Erste Fotos





Wir sehen links das Kennlinienfeld des rechten Transistors. Daneben das des linken Transistors im Textool. Zunächst habe ich auf den linken grünen Taster gedrückt, dann den rechten.

Wenn man einen dritten Transistor rechts hineinsteckt und den rechten Taster nochmal drückt, werden dessen Kennlinien geschrieben und sein hfe ebenfalls neu dargestellt. So kann man beliebig viele Exemplare rechts kontaktieren und nach hfe auswählen.

Der rote Taster löscht die Kennlinien des rechten Transistors. Dazu muss der aber kontaktiert bleiben. Man kann auf diese Weise das Display „sauber“ halten.

Auf die Beschriftung mit roten Basisströmen habe ich verzichtet. Das würde alles sehr unübersichtlich werden. Nur die grünen Basisströme bleiben erhalten. Man sollte die Reihenfolge der Bedienung so wie beschrieben einhalten. Denn schreibt man zuerst die roten Frames, dann wird der Startbildschirm nicht gelöscht. Kleine Unschönheiten, würden aber umfangreiche Programmierung erfordern, wollte man sie beseitigen.

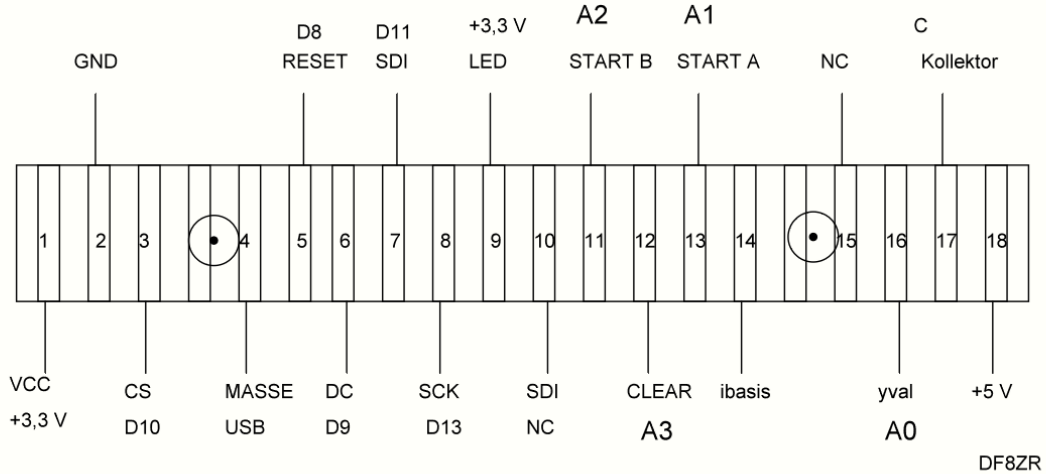
Verdrahtung

Hier habe ich zwei „Lötleisten“ für die Kabelverbindung der beiden Schalen des Gehäuses gebaut. Man sieht die Verbindungen zum Display mit den zugehörigen Signalnamen.

Vielleicht erleichtert das den Nachbau, denn die Namen sind nicht einheitlich.

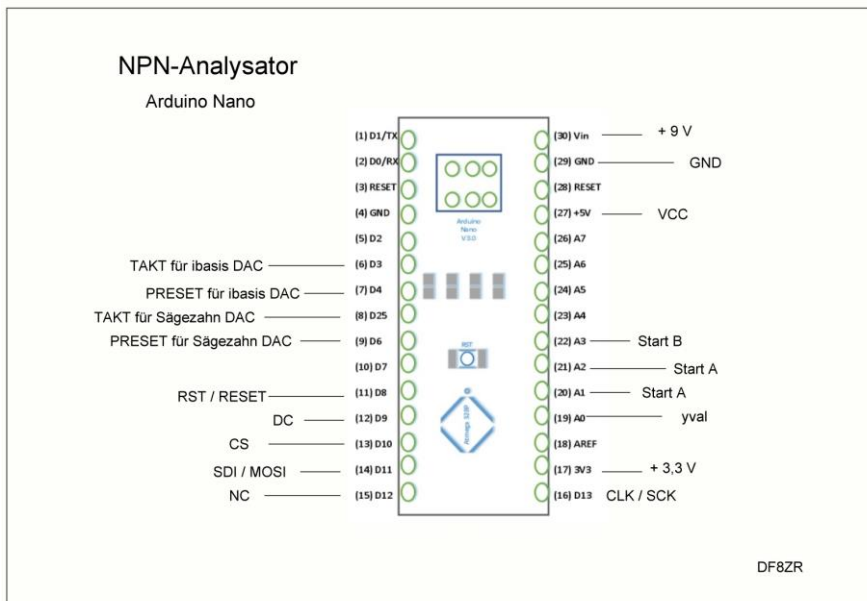
NPN-Analysator

Verbindungsklemmen



Und für die wichtigsten 6 Anschlüsse am Display müssen Spannungsteiler 2k2/3k3 nach Masse gebastelt werden. Das Display darf nur mit 3,3V angesteuert werden. Auch der Vcc erhält 3,3 V!

Die Belegung am Nano



Software

```
//nfn-analysator
//ein einfacher Kennlinienschreiber für NPN-Transistoren
//berndg42.de ; 4.Juni 2024

#include <SPI.h>
#include "Adafruit_ILI9341.h" //https://github.com/adafruit/Adafruit_ILI9341
#include <Fonts/FreeMonoBoldOblique12pt7b.h>
#include <Fonts/FreeSerif9pt7b.h>

#define TFT_DC 9
#define TFT_CS 10 //10
#define TFT_RST 8
#define TFT_MISO 12 //wird nicht gebraucht, bleibt frei
#define TFT_MOSI 11
#define TFT_CLK 13

Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_MOSI, TFT_CLK,
TFT_RST, TFT_MISO);

int i;
int amodus = 0;
int bmodus = 0;
int clearmodus = 0;

int ibeta;
int x1,x2,ibasis, nval;
float y3, i3, faktor,ibasisf, hfe, hfealt;

int y,x;
int n = 1; //Startbedingung für die Anzahl der Kennlinien in einem Frame

long myTimer1 = 0;
long myTimer2 = 0;
long myTimeout = 1000;
long zeit = 0;

long ymin, ymax,val,ywert, yval;
int icmax, ucmax, istep,ustep,ibstep;
//tft.setFont(&FreeMonoBoldOblique12pt7b);

void setup() {

  pinMode(6, INPUT_PULLUP);

  pinMode(2, INPUT_PULLUP);//Taster Modus A

  pinMode(3, OUTPUT); //Takt für Ibasis DAC
```

```

pinMode(4, OUTPUT);// Preset für Ibasis DAC
pinMode(5, OUTPUT);// Takt für den Sägezahn DAC
pinMode(6, OUTPUT);// Preset für Sägezahn DAC
pinMode(7, OUTPUT);//
pinMode(8, OUTPUT);// 11 Display
pinMode(9, OUTPUT);// DC Display

pinMode(A0, INPUT);//für y-wert
pinMode(A1, INPUT);//für Taster Modus A
pinMode(A2, INPUT);//für Taster Modus B
pinMode(A3, INPUT);//für Taster Clear

digitalWrite(6, LOW);

tft.begin();//Start des Displays

tft.fillScreen(ILI9341_BLACK);

tft.setCursor(95,80);
tft.setTextColor(ILI9341_YELLOW); tft.setTextSize(3);
tft.print("NPN");
tft.setCursor(51,110);
tft.print("Analyzer");
tft.setTextColor(ILI9341_WHITE); tft.setTextSize(3);
tft.setTextSize(1);
tft.setCursor(99,151);
tft.print("berndg42");

Serial.begin(115200);

}

//*****los geht's

void skalen(void){

int x1=40; int y1 = 20; int x2 = 40;int y2 = 280;
tft.drawLine(x1, y1, x2, y2, ILI9341_WHITE);
delay(1);
x1=40; y1 = 280; x2 = 220; y2 = 280;
tft.drawLine(x1, y1, x2, y2, ILI9341_WHITE);
x = 10; y =300;
tft.setCursor(x,y);
tft.setTextColor(ILI9341_WHITE); tft.setTextSize(2);

tft.setTextSize(2);

```

```

// X-Achse beschriften; 180 Pixel entsprechen +12V, --> 30 = 2V
tft.setCursor(70,290);
tft.println("2");
tft.setCursor(100,290);
tft.println("4");
tft.setCursor(130,290);
tft.println("6");
tft.setCursor(160,290);
tft.println("8");
tft.setCursor(190,290);
tft.println("10");

//Y-Achse beschriften; 50 pixel = 20mA
tft.setCursor(10,230);
tft.println("20");
tft.setCursor(10,180);
tft.println("40");
tft.setCursor(10,130);
tft.println("60");
tft.setCursor(10,80);
tft.println("80");
tft.setCursor(1,30);
tft.println("100");

}

void basisI(){

digitalWrite(3, HIGH);
digitalWrite(3,LOW);//einen Impuls herausgeben
delayMicroseconds(100);
digitalWrite(3, HIGH);

}

//-----eine Kennlinie zeichnen

void trace()
{

//an 120 Ohm wird gemessen; 100mA bei 12VUce erreicht ein
//Kleinleistungstransistor(BC548) seinen zulässigen Betriebsstrom;

```

```

//bei stärkeren Transistoren muss ein kleinerer Widerstand gewählt
werden(Umschalter)
//100mA > 240 pixel
//1mA > 2,4 pixel; y = (yval / 50) * 2.4; yval ist die Analogspannung an A0 in
mV

x = 40; //Beginn im Ursprung; der liegt bei 40,40

for ( i = 0; i <=180; i++) //Begrenzung in X-Richtung
{

//Kollektorspannung wird stufenweise erhöht; Treppenspannung bzw. wird hier
ein synchron laufender Sägezahn erzeugt

digitalWrite(5, HIGH);
delayMicroseconds(100);
digitalWrite(5,LOW);
delay(1);

yval = analogRead(A0);//Spannung Uce
//1023 bei +5V; max. stehen +4V am Spannungsteiler an ymax = 4000:5 = 800
// das entspricht einem Ice von 100mA; y wird mit 5mV aufgelöst
//es sind 280 Stufen auf der y-Achse; von 40 bis 320; 40 ist NULL = x-Achse

y = yval;

y =280 - y;//beginnt im Nullpunkt
if (amodus == 1)tft.fillCircle(x,y,1, ILI9341_GREEN);//dauert 2s

if (bmodus == 1) tft.fillCircle(x,y,1, ILI9341_RED);

if (clearmodus == 1)tft.fillCircle(x,y,3, ILI9341_BLACK);//ein Radius von 3
überschreibt besser

x++;

} // der for-Schleife, der Zähler cd4029 muss wieder auf NULL gesetzt
werden!!

digitalWrite(6, HIGH);//Preset auf 0000
delayMicroseconds(100);
digitalWrite(6, LOW);

delay(10);

//Ende trace()
}

//_____

```

```

void beta(){

Serial.print("bmodus1"); Serial.println(bmodus);

faktor = 100.0 / 268;

y3 = yval;
y3 = y3 - 4; // Korrektur
ibasisf = (float) ibasis;

i3 = (float) (faktor * y3);
hfe = i3 / ibasisf;
hfe = 1000 * hfe;

if(amodus == 1){
tft.setTextColor(ILI9341_GREEN); //hfe schreiben
tft.setTextSize(2);
tft.setCursor(50,10);
tft.print("hfe");
tft.setCursor(50,30);
tft.print((int)hfe);
}

if(bmodus == 1){

tft.setTextColor(ILI9341_BLACK);
tft.setTextSize(2);
tft.setCursor(50,50);
tft.print((int)hfealt);

tft.setTextColor(ILI9341_RED);
//tft.setTextSize(2);

tft.setCursor(50,50);
tft.print((int)hfe);
hfealt = hfe;
}

```

```
}
```

```
void frame(){  
  //es werden n Kennlinien in y-Richtung übereinander gezeichnet  
  digitalWrite(4, HIGH);  
  digitalWrite(4, LOW); //Preset des Basisstroms ; setzen auf NULL  
  delayMicroseconds(100);  
  digitalWrite(4, HIGH);  
  
  for(int i = 0; i <= (n-1); i++)  
  {  
  
    basisI(); //ibasis setzen; insgesamt n Impulse herausgeben; die Anzahl der  
    Impulse  
    //bestimmt den analogen Wert für die Ausgangsspannung des DAC und damit den  
    Basisstrom  
    //mit jedem Impuls steigt der Basisstrom  
    //vor dem Frame wird der DAC auf NULL gesetzt  
  
    trace(); //die Kennlinie zeichnen  
  
    if(amodus == 1)  
    {  
      tft.setTextColor(ILI9341_YELLOW);  
  
      //Beschriftung der Kennlinie mit ibasis  
  
      if(i == 5) ibasis = 180;  
      if(i == 4) ibasis = 160; //145  
      if(i == 3){ ibasis = 140; ibeta = i;}  
      if(i == 2){ ibasis = 100; ibeta = i;}  
      if(i == 1) ibasis = 70;  
      if (i == 0) ibasis = 40;  
  
      x = x - 65;  
      y = y - 10;  
      tft.setCursor(x,y);  
  
      tft.setTextSize(1);  
  
      tft.print("Ib = ");  
      tft.print(ibasis);  
      tft.print(" uA");  
  
      tft.setTextSize(2);  
    }  
  }  
} //Ende for
```

```
digitalWrite(4, LOW); //Reset des Basisstroms ; setzen auf NULL
delayMicroseconds(100); //ansonsten könnte der Transistor heiß werden
digitalWrite(4, HIGH);
//hier wurde das Frame komplett geschrieben
```

```
beta();
```

```
}
```

```
void runA()
```

```
{
```

```
//Serial.print("runA");
```

```
tft.fillScreen(ILI9341_BLACK); //Clear des Displays
```

```
delay(100);
```

```
skalen(); //nach dem Frame und dem Clear des Displays
```

```
y=30;
```

```
frame(); //Anzahl n Kurven zeichnen = ein Frame
```

```
delay(100);
```

```
}
```

```
void runB()
```

```
{
```

```
frame(); //Anzahl n Kurven zeichnen in Farbe = RED
```

```
delay(100);
```

```
}
```

```
void runClear()
```

```
{
```

```
frame();
```

```
delay(100);
```

```
}
```

```
//*****Loop
```

```
void loop()
```

```
{
```

```

n = 6; //Anzahl der Kurven; könnte man hier ändern

//-----amodus

if(analogRead(A1) <= 500) {
delay(100);
}

if(analogRead(A1) <= 500) {

amodus = 1;
bmodus = 0;
clearmodus = 0;
runA();
}

//-----bmodus

if(analogRead(A2) <= 500) {
delay(100);
}

if(analogRead(A2) <= 500) {

bmodus = 1;
amodus = 0;
clearmodus = 0;
runB();
}

//-----clearmodus

if(analogRead(A3) <= 500) {
delay(100);
}

if(analogRead(A3) <= 500) { //
clearmodus = 1;
amodus = 0;
bmodus = 0;
runClear();
}

}

```

Das Programm kann man noch ausbauen, um z.B. noch PNP zu testen. Falls unerwünschte Nebenerscheinungen nicht akzeptiert werden, steht es jedem frei, die Software anzupassen. Nur eine kommerzielle Nutzung möchte ich nicht.

Link zum YouTube-Film

<https://youtu.be/JBOOHVXqNQg>

Viel Spaß beim Basteln!

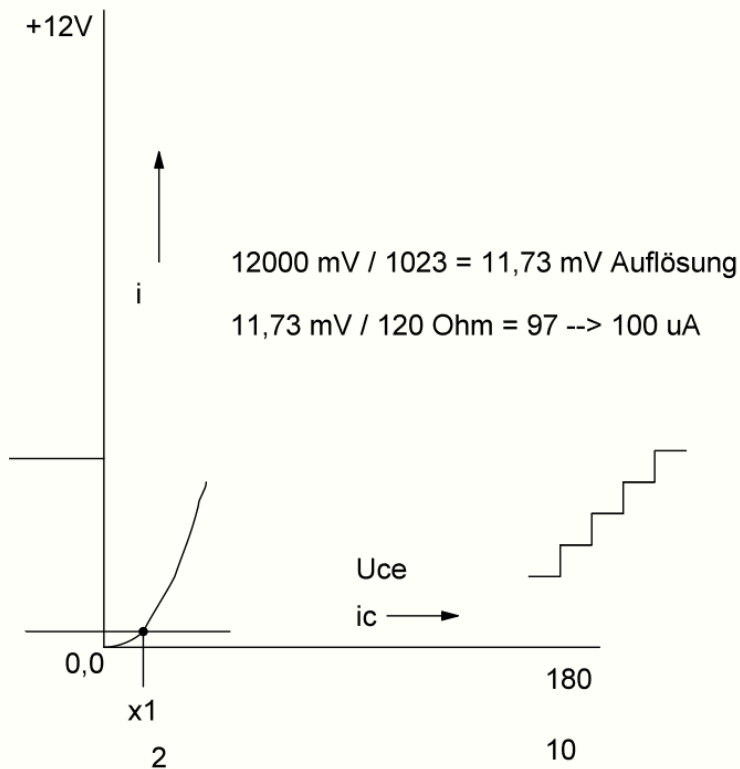
DF8ZR; 19. Juni 2024

NACHTRAG

Der rote Taster war zum Radieren gedacht. Diese Funktion ist aber nicht besonders nützlich. Daher dachte ich darüber nach, was man besser mit ihm machen könnte. Und so verbesserte ich die Anzeigen beim Test von Dioden. Der rote Taster toggelt jetzt zwischen den Betriebsmodi Diode und Transistor. Sie werden kurz in blauer Schrift angezeigt.

Im Modus „Diode“ wird das hfe nicht mehr erscheinen. Dagegen aber eine Anzeige der Diode-Fluss-Spannung „uf“. Für den Bereich bis 3V ist die Ermittlung hinreichend genau. Darüber entstehen leider zu große Abweichungen. Die Spannungsmessung wird mit 12mV aufgelöst. Entsprechend 100 uA. Sie steigt aber mit 47mV/Stufe. Diese Schwelle muss aber erst einmal erreicht sein, bevor man einen gültigen Wert einlesen kann. Die uf wird bestimmt, wenn zwei int-Werte von 1023 möglichen erreicht sind. Dazu wird die x-Position

ermittelt und damit die Spannung berechnet. Die folgende Skizze zeigt das Prinzip.



Schaltungsänderung

Es wird die Mess-Spannung nun bei $2/3 = 2k$ von Masse abgenommen! Das war notwendig. Um ein ausreichende Auflösung im Betrieb „Diode“ zu erreichen. Die Anzeige des hfe beim Transistor wurde durch eine Neukalibrierung angepasst.

Die **Software** hierzu hier:

```
//nnp-analysator
//ein einfacher Kennlinienschreiber für NPN-Transistoren
//berndg42.de ; 12.Juni 2024

#include <SPI.h>
#include "Adafruit_ILI9341.h" //https://github.com/adafruit/Adafruit_ILI9341
```

```

#include <Fonts/FreeMonoBoldOblique12pt7b.h>
#include <Fonts/FreeSerif9pt7b.h>

#define TFT_DC 9
#define TFT_CS 10 //10
#define TFT_RST 8
#define TFT_MISO 12 //wird nicht gebraucht, bleibt frei
#define TFT_MOSI 11
#define TFT_CLK 13

Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_MOSI, TFT_CLK,
TFT_RST, TFT_MISO);

int i;
int amodus = 0;
int bmodus = 0;
int diode = 0;
int transistor = 1;

float uf;

int ibeta,ic, ufl;
int x1,x2,ibasis;
float y3, i3, faktor,ibasisf, hfe, hfealt;

int y,x;
int n = 1; //Startbedingung für die Anzahl der Kennlinien in einem Frame

long myTimer1 = 0;
long myTimer2 = 0;
long myTimeout = 1000;
long zeit = 0;

long yval,ymin, ymax;

void setup() {

  pinMode(6, INPUT_PULLUP);

  pinMode(2, INPUT_PULLUP);//Taster Modus A

  pinMode(3, OUTPUT); //Takt für Ibasis DAC
  pinMode(4, OUTPUT);// Preset für Ibasis DAC
  pinMode(5, OUTPUT);// Takt für den Sägezahn DAC
  pinMode(6, OUTPUT);// Preset für Sägezahn DAC
  pinMode(7, OUTPUT);//
  pinMode(8, OUTPUT);// 11 Display
  pinMode(9, OUTPUT);// DC Display

```

```

    pinMode(A0, INPUT); //für y-wert
    pinMode(A1, INPUT); //für Taster Modus A
    pinMode(A2, INPUT); //für Taster Modus B
    pinMode(A3, INPUT); //für Taster Clear

digitalWrite(6, LOW);

tft.begin(); //Start des Displays

tft.fillScreen(ILI9341_BLACK);

tft.setCursor(95,80);
tft.setTextColor(ILI9341_YELLOW); tft.setTextSize(3);
tft.print("NPN");
tft.setCursor(51,110);
tft.print("Analyzer");
tft.setTextColor(ILI9341_WHITE); tft.setTextSize(3);
tft.setTextSize(1);
tft.setCursor(99,151);
tft.print("berndg42");

    Serial.begin(115200);

}

//*****los geht's

void skalen(void){

int x1=40; int y1 = 20; int x2 = 40;int y2 = 280;
tft.drawLine(x1, y1, x2, y2, ILI9341_WHITE);
delay(1);
x1=40; y1 = 280; x2 = 220; y2 = 280;
tft.drawLine(x1, y1, x2, y2, ILI9341_WHITE);
x = 10; y =300;
tft.setCursor(x,y);
tft.setTextColor(ILI9341_WHITE); tft.setTextSize(2);

    tft.setTextSize(2);

// X-Achse beschriften; 180 Pixel entsprechen +12V, --> 30 = 2V
    tft.setCursor(70,290);
    tft.println("2");
    tft.setCursor(100,290);
    tft.println("4");
    tft.setCursor(130,290);

```

```

tft.println("6");
tft.setCursor(160,290);
tft.println("8");
tft.setCursor(190,290);
tft.println("10");

//Y-Achse beschriften; 50 pixel = 20mA
tft.setCursor(10,230);
tft.println("20");
tft.setCursor(10,180);
tft.println("40");
tft.setCursor(10,130);
tft.println("60");
tft.setCursor(10,80);
tft.println("80");
tft.setCursor(1,30);
tft.println("100");

}

void basisI(){

digitalWrite(3, HIGH);
digitalWrite(3,LOW);//einen Impuls herausgeben
delayMicroseconds(100);
digitalWrite(3, HIGH);

}

//-----eine Kennlinie zeichnen

void trace()
{

//an 120 Ohm wird gemessen; 100mA bei 12VUce erreicht ein
//Kleinleistungstransistor(BC548) seinen zulässigen Betriebsstrom;
//bei stärkeren Transistoren muss ein kleinerer Widerstand gewählt
werden(Umschalter)
//100mA > 240 pixel
//1mA > 2,4 pixel; y = (yval / 50) * 2.4; yval ist die Analogspannung an A0 in
mV

x = 40; //Beginn im Ursprung; der liegt bei 40,40

```

```

for ( ic = 0; ic <=180; ic++) //180 Begrenzung in X-Richtung
//////////
{

//Kollektorspannung wird stufenweise erhöht; Treppenspannung bzw. wird hier
ein synchron laufender Sägezahn erzeugt

digitalWrite(5, HIGH);//Takt
delayMicroseconds(100);
digitalWrite(5,LOW);
delay(1);

yval = analogRead(A0);//Spannung Uce
//1023 bei +5V; max. stehen +4V am Spannungsteiler an ymax = 4000:5 = 800
// das entspricht einem Ice von 100mA; y wird mit 5mV aufgelöst
//es sind 280 Stufen auf der y-Achse; von 40 bis 320; 40 ist NULL = x-Achse
//delay(10);//////////

y = (yval / 2 ) ;

y =280 - y;//280 beginnt im Nullpunkt
if (amodus == 1)tft.fillCircle(x,y,1, ILI9341_GREEN);//dauert 2s

if (bmodus == 1) tft.fillCircle(x,y,1, ILI9341_RED);

//if (ceramodus == 1)tft.fillCircle(x,y,3, ILI9341_BLACK);//ein Radius von 3
überschreibt besser //////////

x++;

} // der for-Schleife, der Zähler cd4029 muss wieder auf NULL gesetzt
werden!!

digitalWrite(6, HIGH);//Preset auf 0000
delayMicroseconds(100);
digitalWrite(6, LOW);

delay(10);

} //Ende trace()

// _____

void beta(){

```

```

faktor = 100.0 / 268;

y3 = yval;

ibasisf = (float) ibasis;

i3 = (float) (faktor * y3);
hfe = i3 / ibasisf;
hfe = 540 * hfe; //Kalibrierung

if(transistor > 0){

if(amodus == 1){
tft.setTextColor(ILI9341_GREEN); //hfe schreiben
tft.setTextSize(2);
tft.setCursor(50,10);
tft.print("hfe");
tft.setCursor(50,30);
tft.print((int)hfe);
}

if(bmodus == 1){

tft.setTextColor(ILI9341_BLACK);
tft.setTextSize(2);
tft.setCursor(50,50);
tft.print((int)hfealt);

tft.setTextColor(ILI9341_RED);
//tft.setTextSize(2);

tft.setCursor(50,50);
tft.print((int)hfe);
hfealt = hfe;
}
}
}

void frame(){
//es werden n Kennlinien in y-Richtung uebereinander gezeichnet
digitalWrite(4, HIGH);
digitalWrite(4, LOW); //Preset des Basisstroms ; setzen auf NULL
delayMicroseconds(100);
digitalWrite(4, HIGH);

for(int i = 0; i <= (n-1); i++)
{

```

```

basisI(); //ibasis setzen; insgesamt n Impulse herausgeben; die Anzahl der
Impulse
//bestimmt den analogen Wert für die Ausgangsspannung des DAC und damit den
Basisstrom
//mit jedem Impuls steigt der Basisstrom
//vor dem Frame wird der DAC auf NULL gesetzt

trace(); //die Kennlinie zeichnen

if(amodus == 1)
{
tft.setTextColor(ILI9341_YELLOW);

//Beschriftung der Kennlinie mit ibasis

if(i == 5) ibasis = 180;
if(i == 4) ibasis = 160;//145
if(i == 3){ ibasis = 140; ibeta = i;}
if(i == 2){ ibasis = 100; ibeta = i;}
if(i == 1) ibasis = 70;
if (i == 0)ibasis = 40;

x = x - 65;
y = y - 10;
tft.setCursor(x,y);

tft.setTextSize(1);

tft.print("Ib = ");
tft.print(ibasis);
tft.print(" uA");

tft.setTextSize(2);
}
} //Ende for

digitalWrite(4, LOW);//Reset des Basisstroms ; setzen auf NULL
delayMicroseconds(100);//ansonsten könnte der Transistor heiß werden
digitalWrite(4, HIGH);
//hier wurde das Frame komplett geschrieben

if (diode == 0)
{
    transistor = 1;

```

```

    beta();
}

else ufe();

}

void ufe()
{
    x = 40; //Beginn im Ursprung; der liegt bei 40,40

    digitalWrite(6, HIGH); //Preset auf 0000
    delayMicroseconds(100);
    digitalWrite(6, LOW);

    for ( ic = 0; ic < 80; ic++) //ic == 0 führt zu einem nicht vom compiler
    bemerkten Fehler!!!
    {

        digitalWrite(5, HIGH); //einen Takt herausgeben und die Uce erhöhen
        delayMicroseconds(100);
        digitalWrite(5, LOW);
        delay(1);

        x++;

        uf = int(analogRead(A0));

        if(uf > 2) //280 * 2 wegen +6V treibende Spannung im Maximum
        {
            x1 = x;
            break;
        }

    }

    delay(100);

    digitalWrite(6, HIGH); //Preset auf 0000
    delayMicroseconds(100);
    digitalWrite(6, LOW);
}

```

```
uf = 4.71 * (x1 - 40);  
uf = uf * 20; //Nur für Fluss-Spannungen bis 3V ausreichend genau!
```

```
uf1 = round(uf);
```

```
delay(10);  
if(amodus == 1) anzeigeufe();  
  
}
```

```
void anzeigeufe()  
{
```

```
tft.setTextColor(ILI9341_BLACK);  
tft.setTextSize(2);  
tft.setCursor(45,50);  
tft.print(uf1);  
tft.setCursor(45,30);  
tft.print("uf");
```

```
tft.setTextColor(ILI9341_GREEN);
```

```
tft.setCursor(45,30);  
tft.print("uf");
```

```
tft.setCursor(45,50);  
tft.print(uf1);
```

```
}
```

```
void runA()  
{
```

```
//Serial.print("runA");  
tft.fillScreen(ILI9341_BLACK); //Clear des Displays  
delay(100);  
skalen(); //nach dem Frame und dem Clear des Displays  
y=30;  
if(diode == 1) n = 1;  
frame(); //Anzahl n Kurven zeichnen = ein Frame  
delay(100);
```

```
}
```

```
void runB()
```

```

{

if(diode == 1) n = 1;
frame();//Anzahl n Kurven zeichnen in Farbe = RED
delay(100);

}

void rundiode()
{
  tft.setTextSize(2);
  tft.setCursor(100,260);
tft.setTextColor(ILI9341_BLACK);
tft.print("Diode");
tft.setCursor(100,260);
tft.print("Transistor");

tft.setTextColor(ILI9341_BLUE);

  if(diode == 1)

  {
tft.setCursor(100,260);

  tft.print("Diode");

  }

  else {
    tft.setCursor(100,260);

    tft.print("Transistor");
    transistor = 1;
  }

delay(100);

}

//*****Loop

void loop()
{

n = 6;// 6 Anzahl der Kurven; kann man hier ändern
transistor = 1;//Modus Transistor

```

```

//-----amodus

if(analogRead(A1) <= 500) {
delay(100);
}

if(analogRead(A1) <= 500) {

amodus = 1;
bmodus = 0;
//diode = 0;////////////////////////////////////
runA();

}

//-----bmodus

if(analogRead(A2) <= 500) {
delay(100);
}

if(analogRead(A2) <= 500) {

bmodus = 1;
amodus = 0;

runB();
}

//-----clearmodus

if(analogRead(A3) <= 500) {
delay(100);
}

if(analogRead(A3) <= 500)
{

amodus = 0;
bmodus = 0;
if(diode ==1)diode =0;
else diode =1;

if(diode == 1)transistor = 0;
rundiode();

}

```

}

DF8ZR; 19. Juni 2024