



Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Signale, Systeme und Sensoren

DIGITALISIERUNG

Kiattipoom Pensuwan, Thanh Son Dang

Konstanz, 15. Januar 2019

Zusammenfassung

Thema:	DIGITALISIERUNG	
Autoren:	Kiattipoom Pensuwan	ki851pen@htwg-konstanz.de
	Thanh Son Dang	th851dan@htwg-konstanz.de
Betreuer:	Prof. Dr. Matthias O. Franz	mfranz@htwg-konstanz.de
	Jürgen Keppler	juergen.keppler@htwg-konstanz.de
	Simon Christofzik	si241chr@htwg-konstanz.de

In diesem Versuch wird ein einfacher Spracherkenner aufgebaut, welcher nur vier einfache Befehle "Hoch", "Tief", "jvgykuvjkkghk Fenster durchführen und daraus den Mittelwert abbilden. Mit diesen Referenzspektren und der Mustererkennung durch Korrelation(Korrelationskoeffizienten nach Bravais-Pearson) können das gesprochen Wort erkannt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis	III
Tabellenverzeichnis	IV
1 Fourieranalyse lang andauernder Signale	1
1.1 Fragestellung, Messprinzip, Aufbau, Messmittel	1
1.2 Messwerte	2
1.3 Auswertung	2
1.4 Interpretation	5
2 Spracherkennung	6
2.1 Fragestellung, Messprinzip, Aufbau, Messmittel	6
2.2 Messwerte	7
2.3 Auswertung	8
2.4 Interpretation	9
Anhang	10
A.1 Quellcode	10
A.1.1 Quellcode Signal Aufnehmen, Triggerfunktion und Fouriertransformation	10
A.1.2 Quellcode Windowing und Spektrum Versuch 1	11
A.1.3 Quellcode Windowing und Referenzspektrum Versuch 2	13
A.1.4 Quellcode Korrelation und Spracherkenner Versuch 2	14

Abbildungsverzeichnis

1.1	Das komplette aufgenommene Signal von den Worten "was cooles"	2
1.2	Das abgeschnittene Signal von den Worten "was cooles"	2
1.3	Das Amplitudenspektrum von dem abgeschnittenen Signal	3
1.4	Beispiel von einem Fenster(Blau:ohne Fensterfunktion, Orange:mit Fensterfunktion)	3
1.5	Amplitudenspektrum durch Mittelung aller lokalen Fouriertransformierten .	4
1.6	Amplitudespektrum aller lokalen Fouriertransformierten	4
2.1	Referenzspektrum von dem Befehl "Hoch"	7
2.2	Referenzspektrum von dem Befehl "Tief"	7
2.3	Referenzspektrum von dem Befehl "Rechts"	8
2.4	Referenzspektrum von dem Befehl "Links"	8

Tabellenverzeichnis

2.1	Detektionsrate	9
-----	--------------------------	---

1

Programmierung der AD/DA-Wandlerkarte

Den Code davon bitte den Anhang A.1.1 sehen.

2

Genauigkeit der AD-Wandlung

2.1 Fragestellung, Messprinzip, Aufbau, Messmittel

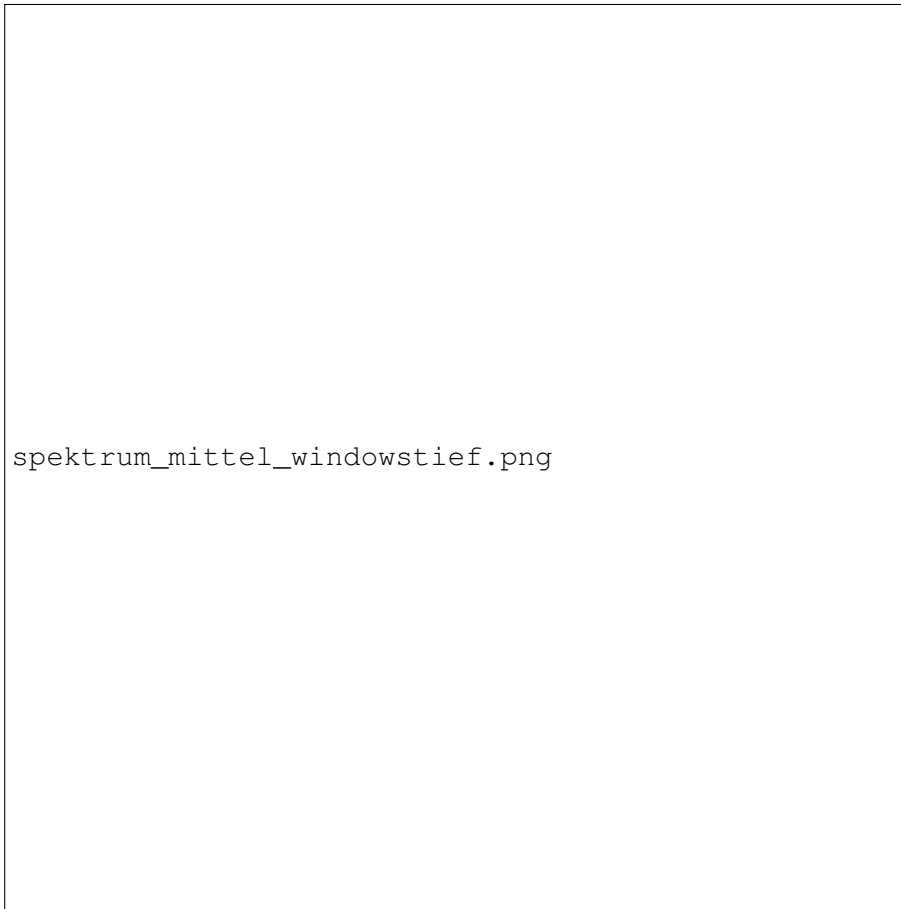
In diesem Versuch wird die Genauigkeit der AD-Wandlung von 3 Messgeräten (Multifunktionsbox ME-RedLab USB-1208LS, das hochgenaue Feinmessgerät Keithley TRMS 179 und das analoge Multimeter Philips PM 2503) ermittelt. Die Messgeräten werden mit Gleichspannungsquelle verbunden und die Spannung wird von 1 - 10 V in Schritten von 1 V variiert.

Zuerst wird der theoretische Quantisierungsfehler berechnet.

2.2 Messwerte



Abbildung 2.1: Referenzspektrum von dem Befehl "Hoch"



spektrum_mittel_windowstief.png

Abbildung 2.2: Referenzspektrum von dem Befehl "Tief"



spektrum_mittel_windowsrechts.png

Abbildung 2.3: Referenzspektrum von dem Befehl "Rechts"



Abbildung 2.4: Referenzspektrum von dem Befehl "Links"

2.3 Auswertung

Korrelationskoeffizient nach Bravais-Pearson:

$$r_{fg} = \frac{\sigma_{fg}}{\sigma_f \cdot \sigma_g}$$

indem σ_f und σ_g jeweils Standardabweichungen von den entsprechenden Signalen f und g sind, die nach der Formel berechnet:

$$\sigma_f^2 = \frac{1}{n-1} \sum_{k=1}^n (f_k - \mu_f)^2$$

Die Kovarianz σ_{fg} ist für diskrete Signale definiert als:

$$\sigma_{fg} = \frac{1}{n} \sum_{k=1}^n (f_k - \mu_f) \cdot (g_k - \mu_g)$$

μ_f ist der Mittelwert:

$$\mu_f = \frac{1}{n} \sum_{k=1}^n f_k$$

Befehl	Sprecher 1	Sprecher 2
Hoch	100%	100%
Tief	100%	80%
Links	100%	100%
Rechts	100%	80%

Tabelle 2.1: Detektionsrate

Den Code davon bitte den Anhang A.1.3 und A.1.4 sehen.

2.4 Interpretation

Die Detektion besagt, dass beim Sprecher 1 100% das Wort korrekt erkannt wurde und beim Sprecher 2 ist abweichend bei der Erkennung der Befehle "Tief" und "Rechts". Das ist total verständlich, weil der Sprecher 1 ist die Person, von der die Referenzspektren hergestellt wurden. Bei dem anderen Sprecher wurde die Befehlen manchmal falsch erkannt. Die Erklärung dafür ist aus den anatomischen Gründe, dass jede Person eigene Stimme und eigene Sprechweise(Akzent) hat. Die Fehlerrate ist nicht so hoch, da die Datenbank der Referenzspektrum noch ziemlich klein ist (von nur 4 Befehlen).

Anhang

A.1 Quellcode

A.1.1 Quellcode Signal Aufnehmen, Triggerfunktion und Fouriertransformation

```
1 import pyaudio
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 FORMAT = pyaudio.paInt16
6 SAMPLEFREQ = 44100
7 FRAMESIZE = 1024
8 NOFFRAMES = 220
9 INPUT_BLOCK_TIME = 0.05
10 INPUT_FRAMES_PER_BLOCK = int(SAMPLEFREQ*INPUT_BLOCK_TIME)
11 p = pyaudio.PyAudio()
12 print('running')
13 stream = p.open(format=FORMAT,channels=1,rate=SAMPLEFREQ, input=True,frames_per_buffer=FRAMESIZE)
14 data = stream.read(NOFFRAMES*FRAMESIZE)
15 decoded = np.fromstring(data, 'Int16')
16 stream.stop_stream()
17 stream.close()
18 p.terminate()
19 print('done')
20
21 #####Signal speichern und darstellen#####
22 string = 'hoch1'
23 np.save(string,decoded)
24 sec = len(decoded) / SAMPLEFREQ
25 plt.xlabel('Zeit in s')
26 plt.ylabel('Amplitude')
```

```

27 Zeit = []
28 for i in range (len(decoded)):
29     Zeit.append(sec/len(decoded) * i)
30 plt.plot(Zeit,decoded)
31 plt.show()
32
33 ####Signal mit Triggerfunktion abschneiden####
34 trigger = 0.1 * np.max(decoded)
35 j = 0
36 for i in decoded:
37     j = j + 1
38     if np.abs(i) > trigger:
39         decoded = decoded[j:j+SAMPLEFREQ]
40         break
41 np.save(string+'abgeschnitten',decoded)
42 plt.xlabel('Zeit in s')
43 plt.ylabel('Amplitude')
44 Zeit = []
45 for i in range (len(decoded)):
46     Zeit.append(1/len(decoded) * i)
47 plt.plot(Zeit,decoded)
48 plt.show()
49
50 #####Fourriertransformation#####
51 spek = abs(np.fft.fft(decoded))
52 plt.plot(spek)
53 plt.savefig('Amplitudenspektrum.png')

```

A.1.2 Quellcode Windowing und Spektrum Versuch 1

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def fft(data,name):
5     plt.figure(figsize=(9,6))
6     Y = abs(np.fft.fft(data))
7     Y = Y[range(int(len(Y)/2))]
8     x = np.linspace(0,22050,22050,endpoint = True)
9     plt.xlabel('Frequenz($Hertz$)')
10    plt.ylabel('Amplitude($Unit$)')
11    plt.plot(x,abs(Y))

```

```

12 plt.savefig(str(name)+".png")
13 plt.show()
14
15 def winspek(data,name):
16     plt.figure(figsize=(9,6))
17     plt.xlabel('Frequenz($Hertz$)')
18     plt.ylabel('Amplitude($Unit$)')
19     st = np.std(data)
20     from scipy import signal
21     gfen = signal.gaussian(512, std = st * 4)
22     g = np.zeros(len(data))
23     fft = np.zeros(len(data))
24     x = np.linspace(0,22050,22050,endpoint = True)
25     for i in range(0,len(data),256):
26         if (i > len(data) - 512):
27             gfen = signal.gaussian(len(data)-i, std = st * 4)
28             g[i:] = data[i:] * gfen
29             fft += abs(np.fft.fft(g))
30             a = np.fft.fft(g)
31             a = a[range(int(len(a)/2))]
32             plt.plot(x,abs(a))
33             break
34             g[i:i+512] = data[i:i+512] * gfen
35             fft += abs(np.fft.fft(g))
36             a = np.fft.fft(g)
37             a = a[range(int(len(a)/2))]
38             plt.plot(x,abs(a))
39             g = np.zeros(len(data))
40     plt.savefig("spektrum_gesamter_windows"+str(name)+".png")
41     plt.show()
42     fft /= 171 #durch die Anzahl der Windows teilen
43     fft = fft[range(int(len(fft)/2))]
44     plt.figure(figsize=(9,6))
45     plt.xlabel('Frequenz($Hertz$)')
46     plt.ylabel('Amplitude($Unit$)')
47     plt.plot(x,fft)
48     plt.savefig("spektrum_mittel_windows"+str(name)+".png")
49     return fft
50
51 data = np.load('was_cooler_abgeschnitten.npy')
52 plt.figure(figsize=(9,6))
53 plt.xlabel('Zeit in s')

```

```

54 plt.ylabel('Amplitude')
55 Zeit = []
56 for i in range(len(data)):
57     Zeit.append(1/len(data) * i)
58 plt.plot(Zeit,data)
59 plt.savefig("v1_abgeschnitten.png")
60 plt.show()
61 fft(data,"Amplitudenspektrum")
62
63 winspek(data,"v1")

```

A.1.3 Quellcode Windowing und Referenzspektrum Versuch 2

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy import signal
4
5 def winspek(data):
6     st = np.std(data)
7     gfen = signal.gaussian(512, std = st * 4)
8     g = np.zeros(len(data))
9     fft = np.zeros(len(data))
10    for i in range(0,len(data),256):
11        if (i > len(data) - 512):
12            break
13        g[i:i+512] = data[i:i+512] * gfen
14        fft += abs(np.fft.fft(g))
15        a = np.fft.fft(g)
16        a = a[range(int(len(a)/2))]
17        g = np.zeros(len(data))
18
19    fft /= 171 #durch die Anzahl der Windows teilen
20    fft = fft[range(int(len(fft)/2))]
21    return fft
22
23 data = np.load('hoch1_abgeschnitten.npy')
24 mit = np.zeros(int(len(data)/2))
25 a = ['hoch', 'tief', 'rechts', 'links']
26 for j in a:
27     mit = 0
28     for i in range(1,6):

```



```

29     data = np.load(j + str(i) + 'abgeschnitten.npy')
30     np.save('spek' + j + str(i), winspek(data))
31     mit += winspek(data)
32     mit = mit/5
33     np.save('ref' + j, mit)
34     plt.figure(figsize=(9,6))
35     plt.xlabel('Frequenz($Hertz$)')
36     plt.ylabel('Amplitude($Unit$)')
37     plt.plot(mit)
38     plt.savefig("spektrum_mittel_windows" + j + ".png")
39     plt.show()

```

A.1.4 Quellcode Korrelation und Spracherkenner Versuch 2

```

1 import numpy as np
2
3 refhoch = np.load('experiment 2a/refhoch.npy')
4 reftief = np.load('experiment 2a/reftief.npy')
5 refrechts = np.load('experiment 2a/refrechts.npy')
6 reflinks = np.load('experiment 2a/reflinks.npy')
7 def spracherkenner(name, person):
8     for i in range(1,6):
9         test = np.load('Spek/spek'+str(name)+str(i)+'t'+str(person)+' .npy')
10        korrcoefh = np.corrcoef(refhoch, y = test)
11        korrcoeft = np.corrcoef(reftief, y = test)
12        korrcoefr = np.corrcoef(refrechts, y = test)
13        korrcoefl = np.corrcoef(reflinks, y = test)
14        a = np.max([np.mean(korrcoefh), np.mean(korrcoeft), np.mean(korrcoefr), np.mean(korrcoefl)])
15        if a == np.mean(korrcoefh):
16            print('hoch')
17        elif a == np.mean(korrcoeft):
18            print('tief')
19        elif a == np.mean(korrcoefr):
20            print('rechts')
21        elif a == np.mean(korrcoefl):
22            print('links')
23
24 spracherkenner('links', 'p')
25 print("-----")
26 spracherkenner('links', 's')

```