

# **Radio Packet Networks**

## **Seminar**

## **Projekt Bericht**

Raffael Colonnello & Giovanni Utzeri

Universität Basel  
Radio Packet Networks (72913-01)  
Herbstsemester 2024

## **Idee:**

Unsere Idee war es, ein Programm zu schreiben, das es ermöglicht, mit Hilfe von Tönen Daten zu übertragen. Hierbei sollten die Standards für Modems genau eingehalten werden, sodass es möglich sein sollte, mit möglichst wenig Aufwand mit herkömmlichen Einwahlmodems über das Telefonnetz zu kommunizieren. In diesem Bericht bezeichnen wir uns, die die Kommunikation durch einen Anruf starten, als Sender und das Gerät, das den Anruf entgegennimmt, als Empfänger.

## **Arbeitsteilung:**

Giovanni suchte nach einer Bibliothek, die es ermöglichte, Sound aufzunehmen und abzuspielen und stellte Messungen an, um Probleme an der Bibliothek zu finden. Ferner sorgte er mittels Threads dafür, dass es uns möglich ist, live sowohl Signale zu empfangen als auch synchron zu dekodieren.

Raffael widmete sich den Protokollen G.168, V.8bis, V.21, sowie dem Einwahlprotokoll und implementierte die Cooley-Tucker-FFT, sowie die Frequenzen-Detektion.

Den Bericht verfassten beide.

## **Standards:**

### Wählen:

Zuerst widmeten wir uns dem Detektieren von Telefonnummern. Diese werden übertragen, indem von zwei Mengen à 4 Frequenzen jeweils eine ausgewählt wird. Es ergeben sich 16 Frequenzen, die Ziffern kodieren (sowie auch \* und #). Die 4 übrigen Zeichen finden anderweitigen Einsatz und sind für uns irrelevant. Die ersten Frequenzen sind: 697Hz, 770Hz, 852Hz, 941Hz. Ist  $x$  die zu sendende Ziffer, so wird die  $(x-1)/3$  te Frequenz gewählt. Null ist ein Spezialfall, für sie wird 941Hz verwendet.

Die zweiten Frequenzen sind: 1209Hz, 1336Hz, 1477Hz, 1633Hz. Wird erneut  $x$  gesendet, so entspricht die  $(x-1)\%3$  te Frequenz der gewünschten. Für Null wird stattdessen 1336Hz verwendet.

### Echounterdrückungsunterdrückung G.168:

In unseren Nachforschungen haben wir erfahren, die Echounterdrückung uns möglicherweise in die Quere kommen könnte und damit in der Regel deaktiviert wird.

Dies ist ein sehr markantes Element im Einwählvorgang, da dies durch einen 2100Hz Dauerton erzeugt wird, dessen Phase regelmäßig umgekehrt wird. Wir entschieden uns, dieses Signal möglichst früh zu senden, damit wir die Fehlerrate möglichst tief halten können. In den von uns gefundenen Tonbeispielen haben wir bemerkt, dass dies ausschliesslich nach der Ausfahrt des Modus' (s. unten) geschieht, jedoch erlauben die Standards die deutlich frühere Unterdrückung, sodass wir dies direkt nach dem Wählen machen.

#### Übertragungsmodus-Auswahlprotokoll V.8bis:

Die Auswahl des Übertragungsprotokolls war für uns bei Weitem der Komplexeste Teil.

Dies aufgrund der breiten Spanne an Optionen, die dieses Protokoll bietet. So ist es unter anderem möglich, aktuelle Übertragungen zu unterbrechen und den verwendeten Standard zu wechseln oder nachzufragen, wer welche Standards anbietet.

Um eine Nachricht zu senden, ist es erst nötig, ein Spezielles Signal zu senden, das aus 2 Tönen besteht. Danach ist eine bekannte Sequenz (0x7e) 2-5 fach zu Wiederholen. Danach kann die eigentliche Nachricht folgen.

#### Übertragungsmodus V.21:

Mit erhöhter Nachfrage von Bildern im Netz wurden schnell höhere Übertragungsraten notwendig. Waren zuerst Raten von 100b/s möglich, so stieg dies mit dem V.21 Standard auf 300b/s und weiter bis zu einem Wert von maximal 56kb/s an. Diese wurden in der Praxis jedoch nicht konstant erreicht, so war es aus eigener Erfahrung auch noch vor knapp 20 Jahren möglich, dass die Kommunikationsgeschwindigkeit auf ca. 300b/s fiel und so das Senden eines emails mehrere Stunden in anspruch nehmen konnte.

Mit steigenden Bitraten stieg auch die Komplexität der Übertragungsverfahren, daher entschieden wir uns für den langsamsten Standard, der typischerweise von allen Modems (ab der 2. Generation) unterstützt wird. Hierbei handelt es sich um den V.21 Standard, bei dem Daten mittels vierer Frequenzen übertragen wird, wobei je zwei Frequenzen dem Sender bzw. dem Empfänger zur Verfügung stehen:

Der Sender verwendet die Frequenzen 980Hz und 1180Hz für Null resp. Eins.  
Der Empfänger verwendet 1650Hz und 1850Hz für den gleichen Zweck.  
So müssen sowohl Sender als auch Empfänger lediglich auf zwei Frequenzen achten.

Wir haben dies so implementiert, dass unser Programm jeweils über die relative Schallenergie eine Wahrscheinlichkeit berechnet (wir nahmen an, dass die Schallenergie, nicht die Lautstärke gleichverteilt ist), was uns mehr Sicherheit geben kann, welche Informationen übertragen wurden: Wird kein Signal gefunden, so wird -32768 zurückgegeben, andernfalls kann die Wahrscheinlichkeit über den Rückgabewert  $x$  wie folgt berechnet werden:

$$P(\text{Eins}) = (x/65536) + 0.5.$$

Dieser Standart ermöglicht auch optional das Einfügen von Zusätzlichen Signalen, falls eine lange (5) Sequenz von Einsen auftritt (beispielsweise beim Übertragen der Zahl 0xff).

Falls eine übertragene Nachricht nicht korrekt empfangen bzw. dekodiert wird, so wird ein negative-acknowledge anstelle eines acknowledge gesendet. Da es unklar ist, welche der vier negative-acknowledge gewählt wird, haben wir uns dafür entschieden einen typ-1-negative-acknowledge zu senden. Hierbei wird mitgeteilt, dass die Nachricht nicht dekodiert werden kann und die Kommunikation daher unterbrochen wird, daher wird in diesem Fall die passend genannte Crash-Funktion aufgerufen.

Im Falle eines NACK(1) kann das andere Gerät natürlich nicht weitermachen, daher ist es hinreichend, zu überprüfen, ob das Empfangene dem Erwarteten entspricht; im Falle eines NACK(1) ist dies nicht der Fall, weswegen auch ein NACK(1) retourniert und das Programm beendet wird.

#### FFT:

Da wir nur einzelne Samples, jedoch nicht direkt Frequenzen messen können, war es für uns unerlässlich, eine Funktion zu haben, die dies übersetzt. Aus diesem Grund entschieden wir uns, die Fast Fourier Transformation zu programmieren. Diese nimmt einen Vektor an Samples entgegen und retourniert einen Frequenzen-Vektor.

Des Weiteren haben wir einen Detektor verfasst, der entscheiden kann, welche Frequenzen signifikant vertreten sind; Diese werden zurückgegeben, sodass wir DTMF sowie FSK decodieren können. Anfangs litt dessen Genauigkeit, dies konnte jedoch durch Prof. Tschudins Hinweis, benachbarte Frequenzen ( $\pm 2\text{Hz}$ ) auch mitzuzählen, massiv verbessert werden.

# **Prototyp und Erste Schritte:**

## Bibliothek Portaudio:

Eine unserer ersten Aufgaben bestand darin, eine kompatible Audiobibliothek zu identifizieren, die sowohl unter Linux Ubuntu als auch unter Windows funktioniert. Eine erste Recherche führte zu keinen zufriedenstellenden Ergebnissen. Schliesslich stiessen wir auf die Bibliothek PortAudio, die wir daraufhin auf einem Linux-Laptop kompilierten

## Prototypen:

Nachdem wir erfolgreich eine passende Bibliothek gefunden hatten, machten wir uns mit ihr vertraut. Hierfür durchsuchten wir die Dokumentation, um die Bibliothek mittels zweier Beispiele zu testen: die Aufnahme mit Mikro, sowie die Wiedergabe über Lautsprecher. Hierfür schrieben wir zwei Prototypen.

## Wiedergabe:

Unser erster Prototyp ist primär ein simpler Tongenerator, der einen Input in Form einer Telefonnummer (z.B. 0611234567) entgegennimmt, die Ziffern mittels des DTMF Verfahrens codiert und diese mittels der von Portaudio zur Verfügung gestellten Ausgabefunktion abspielt. Hierfür schrieben wir die audioCallback2 Funktion, die dies mittels beepSequenceOfNumbers ermöglicht. Dies prüft, ob es sich bei der Eingabe um eine Zahl handelt, in welchem Fall sie mittls beeps ausgegeben wird. Die Tonwiedergaberate ist im Code definiert.

## Aufnahme:

Zusammen mit den Funktionen, die die Frequenzen entdecken und deren Maxima der Frequenzamplituden mittels detect und fft zu finden, konnten wir einen Prototypen verfassen, der einen Ton aufnimmt, diesen wiedergibt (was zur Fehlerfindung von unschätzbarem Wert ist, mehr dazu unter Experimente), und mittels der Detection-Blocks die Frequenzen anhand der Repräsentation in der Aufnahme auflistet, um die prominentesten Frequenzen zu retournieren.

# **Experimente:**

Die Modalitäten beider Messungen waren identisch. Wir nutzten die Android App Phyphox, um die Sinustöne zu generieren, um das Aufnahme-Programm zu testen. Zu den Modalitäten gehören eine manuelle Kalibrierung des Input Gains des Laptopmikros und der Ausgabelautstärke des Handys. Dabei wurde das Programm zum Zuhören gestartet und auditiv überprüft, ob man den Input des Handys, mit möglichst wenig Nebengeräuschen hören kann.

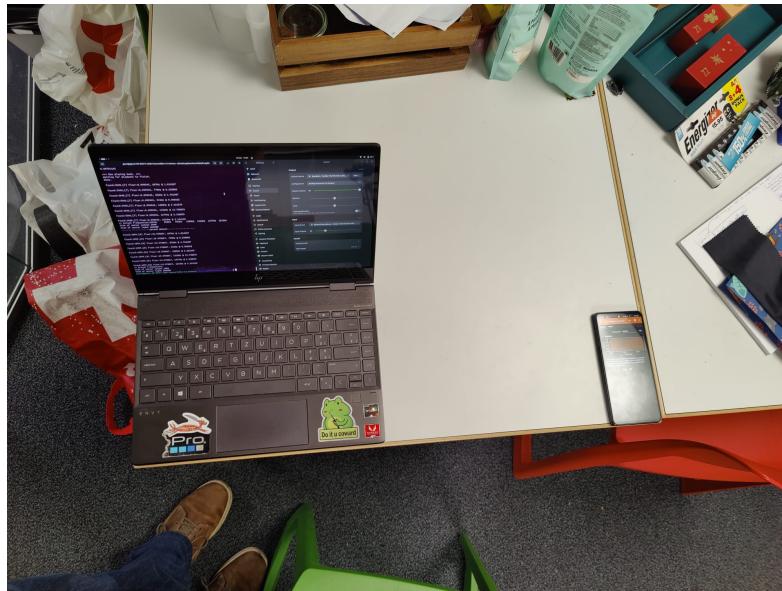


Abb. 1:Experiment Aufbau, Laptop mit unserem Prototypen(Links) und Android Smartphone mit Phyphox Tongenerator(Rchts).

#### Methodik:

Im Setup, siehe Abb. 1 aufgestellt und nach manuellen Kalibrierung, wurden alle Frequenzen einzeln abgespielt, für die Dauer, die im Programm definiert ist. Das Programm spielt das Aufgenommene erneut wieder, um gegebenenfalls nachzuhorchen, ob die Aufnahme erfolgreich war. Am Ende liefert das Terminal zwei Analysen zweier verschiedener Grenzwerte, welche bestimmen, welche Frequenzen am prominentesten sind, wobei wir manuell der Tabelle die Werte entnommen und diese in ein Excel file eingaben, um die jeweiligen Maxima zu bestimmen, und um mögliche Statisitscheauswertunge zu durchführen. Die Messungen wurden pro Frequenz fünffach durchgeführt und für einen einzelnen Fall im zweiten Experiment wurde auch ein Zweiklang abgespielt, um die Zweiklang-Detektion zu testen.

#### Messreihen:

Untenstehend sind 3 von 8 Messreihen angegeben, für welche die Erkennung funktioniert hat (Abb.2 und Abb. 3) und für ein Beispiel einer misslungenen Erkennung (Abb. 4). Die Excel-Tabelle haben wir in das Git-Repo gepackt, zusammen mit den Original-Abbildungen der Terminalaufnahmen (Abb. 5).

	Frequencies	697	770	852	941	1209	1336	1477	1633	MaxValues	Index
2	1	6260.587	5422.654	3959.594	3316.253	2818.975	2113.511	1694.748	1225.4	6260.587	697
3	2	20323.549	15453.501	13287.756	11964.668	7074.392	5718.687	5180.481	3683.775	20323.549	697
4	3	19841.861	16007.512	16898.424	12152.252	6790.633	5458.242	4958.267	3728.439	19841.861	697
5	4	19608.492	20237.778	12232.001	10077.172	6754.556	6030.508	4411.76	3880.007	20237.778	770
6	5	20744.956	16197.53	12969.267	12061.906	6727.604	5384.958	4292.768	3552.331	20744.956	697

Abb. 2: Tabelle mit der einzigen richtig erkannten Frequenzen, hier 697Hz.

	Frequencies	697	770	852	941	1209	1336	1477	1633	MaxValues	Index
2	1	37.024	57.573	55.781	72.399	21.342	17.552	25.959	12.809	72.399	941
3	2	30.73	99.336	32.478	73.768	10.7	40.919	67.889	11.32	99.336	770
4	3	28.069	49.392	61.942	120.533	36.943	23.433	32.178	10.465	120.533	941
5	4	61.634	90.338	66.733	116.267	66.222	52.139	45.108	8.247	116.267	941
6	5	30.203	20.909	42.506	103.286	44.969	20.785	53.839	15.829	103.286	941

Abb. 3: Tabelle mit der einzigen richtig erkannten Frequenzen, hier 941Hz.

	Frequencies	697	770	852	941	1209	1336	1477	1633	MaxValues	Index
2	1	19834.651	17275.447	12493.575	11014.935	6752.36	5603.604	4815.433	3608.433	19834.651	697
3	2	114.295	1360.304	2096.847	401.375	224.623	198.078	1294.251	226.422	2096.847	852
4	3	64.452	59.247	361.465	108.028	89.0033	72.34	121.038	12.083	361.465	852
5	4	29.3	314.587	99.645	52.233	52.215	63.253	90.033	14.932	314.587	770
6	5	21508.239	17508.032	13955.917	11070.356	6869.984	5122.931	4752.057	3675.71	21508.239	697

Abb. 4: Tabelle mit der Messreihe zu den 770Hz, wobei man hier sieht, dass die Frequenz nur einmalig erkannt wurde, ähnliches gilt auch für die Frequenzen: 852Hz, 1209Hz, 1336Hz, 1477Hz und 1633Hz.

<img alt="Screenshot of a terminal window showing a list of detected frequencies and their corresponding floor values. The output is as follows: 
Found: 930. [8] Floor:0.999785, 697Hz @ 37.023908
Found: 930. [8] Floor:0.999783, 770Hz @ 57.57603
Found: 930. [8] Floor:0.999783, 852Hz @ 55.781057
Found: 930. [8] Floor:0.999783, 941Hz @ 72.399583
Found: 930. [8] Floor:0.999783, 1209Hz @ 21.342078
Found: 930. [8] Floor:0.999783, 1336Hz @ 17.552051
Found: 930. [8] Floor:0.999783, 1477Hz @ 25.959289
Found: 930. [8] Floor:0.999783, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz 1633Hz
Size of vector input 264688
-----second detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----third detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----fourth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----fifth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----sixth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----seventh detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----eighth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----ninth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----tenth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----eleventh detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----twelfth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----thirteenth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----fourteenth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----fifteenth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----sixteenth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----seventeenth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----eighteenth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----nineteenth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19.964665, 1209Hz @ 21.342078
Found: 2490. [16] Floor:19.964665, 1336Hz @ 17.552051
Found: 2490. [16] Floor:19.964665, 1477Hz @ 25.959289
Found: 2490. [16] Floor:19.964665, 1633Hz @ 12.809272
... Actual\_Frequencies:697Hz 770Hz 852Hz 941Hz 1209Hz 1336Hz 1477Hz
Size of vector input 264688
-----twentieth detect-----
Found: 2490. [16] Floor:19.964665, 697Hz @ 37.023908
Found: 2490. [16] Floor:19.964665, 770Hz @ 57.57603
Found: 2490. [16] Floor:19.964665, 852Hz @ 55.781057
Found: 2490. [16] Floor:19.964665, 941Hz @ 72.399583
Found: 2490. [16] Floor:19

Informatikdepartements befand, konnte das Hintergrundrauschen tief gehalten werden. So schlossen wir, dass das Problem an der Wahl der Samplerate lag, sodass Phänomene wie Aliasing mögliche Problemquellen waren. Nach einem Meeting im Plenum und nach einer Besprechung mit Prof. Tschudin ergab sich ein besserer Wert für die Samplerate, mit welcher die zweite Experiment Serie durchgeführt wurden.

#### Messreihe 2:

Mit der neuen, angepassten Samplerate wurden die Messungen wie zuvor im Abschnitt «Methodik» beschrieben wiederholt. Hierbei erkannte unser angepasster Prototyp, unerklärlicherweise nur die Frequenzen 697Hz und 941Hz in vier von fünf Fällen(siehe Abb. 2 und 3.). Die anderen wurden selten entdeckt. Dies brachte uns zu einem grossen Problem, bei welchem wir uns entscheiden mussten, wie weiter vorzugehen war: Das Problem lag entweder bei der Hardware (bzw. Portaudio) oder bei Software.

Da jedoch Wiedergabefehler bemerkt wurden (nachdem die Samples aufgenommen und wieder abgespielt wurden), wurden wir uns sicher, dass es am Zusammenspiel zwischen Hardware und der Portaudio Bibliothek liegen musste.

Wir entschieden uns, trotz den Problemen mit Portaudio, den Code aus den Prototypen zusammenzufügen, um unser finales Programm zu erstellen, welches den Übertragungsmodus V.21 und unsere Implementation für die Ein- und Ausgabe der Töne enthielt.

## **Finale Implementation:**

Zusammen haben wir das Programm kommentiert, was das Ganze verständlicher machte. Weiter haben wir den Ablauf des Verbindungsaufbaus zu zweit besprochen. Giovanni wurde der verwendete Standard und die von Raffael geschriebenen Funktionen erklärt, um den Code gemeinsam zu vervollständigen. Zu guter Letzt galt es, die Funktionen TX() und RX() zu schreiben, womit wir Daten senden(TX) und empfangen(RX) können. Diese mussten in separaten Threads laufen, um eine gleichzeitige Vollduplexkommunikation zu ermöglichen. Der Aufbau und die Verwendung der Threads wurde mittels ChatGPT zusammen erarbeitet. Wir liessen hierdurch unseren groben Code verbessern, sodass wir in C++ die Thread-Implementation fertigstellen konnten.

Wir schrieben, anhand des V.21 Standards folgende Funktionen, um den Ablauf des Verbindungsaufbaus und das Senden der Daten zu ermöglichen:

preamble(), detectPreamble(), encodeData(), initiateCallA(), receiveCallA(),  
initiateCallB(), detectMSe1(), detectMSe2(), detectESr1, detectESr2(), decodeA(),  
decodeB(), decodeR(), echoSuppressionSuppression().

#### Ablauf:

Die Kommunikation läuft wie folgt: Nachdem der Sender die Telefonnummer sowie den Echounterdrückungsunterdrückungston abspielt, initiiert der Empfänger die Kommunikation mittels V.8bis, indem er InitiateCallA startet. Daraufhin antwortet der Sender, bei korrektem Empfangen, mit seinen Optionen(receiveCallA). Sieht der Empfänger, dass beide den V.21 Standard unterstützen, so sendet dieser einen ACK mittels InitiateCallB. Kurz darauf wird die Übertragung vom Sender gestartet. Nach dem Senden der Daten terminiert der Sender. Der Empfänger wartet weiterhin auf einkommende Daten, da das Beenden nicht Teil der Standards ist.

## **Resultate:**

Die grössten Herausforderungen waren das Zusammenspiel der Mikroeingabe und Tonwiedergabe. Da der Audiokanal sehr rauschbehaftet ist, in starkem Kontrast zu einer typischen Modemkommunikation, sind Fehlerbehebungen für die Standards sekundärer Bedeutung. Auf Grund der hohen Fehlerrate kommt ein vollständiger Verbindungsaufbau des seltener zu Stande. Zusätzlich haperte es, wie oben erwähnt, bei der Portaudio Bibliothek, deren Tonaufnahme zu wünschen übrig liess.

Abschliessend konnten wir eine Fehlerrate von ca. 30% aufweisen, was wir bei 30 Baud erreichten.

## **Zusammenfassung:**

Mittels des Audiokanals können Daten übertragen werden, da wir uns jedoch auf die ITU-Standards beschränkten, wurden wir stark bezüglich der Wahl der Frequenzen eingeschränkt; Diese finden sich häufig in der Sprache, was neben der Bibliothekswahl einer der Gründe der stark fehlerbehafteten Kommunikation ist. Die hohe Fehlerrate führt zu dem Problem, das bei dem Verbindungsaufbau (bei dem keine Fehler passieren dürfen) oft Fehler auftreten, die einen erneuten Aufbau der Kommunikation mit sich ziehen.

# **Quellen:**

Hörbeispiele:

[archive.goughlui.com/legacy/soundofmodems/](http://archive.goughlui.com/legacy/soundofmodems/)  
[windytan.com/2012/11/the-sound-of-dialup-pictured.html](http://windytan.com/2012/11/the-sound-of-dialup-pictured.html)

ITU-G.168:

[itu.int/rec/T-REC-G.168-201504-I/en](http://itu.int/rec/T-REC-G.168-201504-I/en)

ITU-V.8bis:

[itu.int/rec/T-REC-V.8bis-200011-I/en](http://itu.int/rec/T-REC-V.8bis-200011-I/en)  
[itu.int/rec/T-REC-V.8bis-199608-S/en](http://itu.int/rec/T-REC-V.8bis-199608-S/en)  
[itu.int/rec/T-REC-V.8bis-199809-S/en](http://itu.int/rec/T-REC-V.8bis-199809-S/en)

ITU-V.21:

[itu.int/rec/T-REC-V.21-198811-I/en](http://itu.int/rec/T-REC-V.21-198811-I/en)

Portaudio:

<https://portaudio.com>

Phyphox:

[play.google.com/store/apps/details?id=de.rwth\\_aachen.phyphox](http://play.google.com/store/apps/details?id=de.rwth_aachen.phyphox)

Referenz zu den Vektoren:

[cppreference.com](http://cppreference.com)

ChatGPT:

[chatgpt.com](http://chatgpt.com)