Übung Embedded and Pervasive Systems	Institut für Pervasive Computing
2013S, Übungsprojekt	Univ.-Prof. Dr. Alois Ferscha
Elektr. Abgabe: Bis Di, 18. Juni 2013, **11.59 Uhr**	A. Riener, G. Hölzl

Zuname:				Tutor:	
Vorname:				Übungsgruppe:	
Matr. Nr.:		SKZ:		Punkte (max. 24)	

Signalverarbeitung: Morsecode (24+16+24+8=72 Punkte)

Der **großen Vorteile** bei der Übertragung von Informationen mittels Morsecode sind:

- Durch den einfachen Signalaufbau werden kaum Anforderungen an die Hardware zum Senden bzw. Empfangen gestellt.
- Der Code kann auch bei sehr ungünstigem Signal-Rausch-Verhältnis noch entschlüsselt werden.
- Die Übertragung geht äußert sparsam mit Bandbreite um.

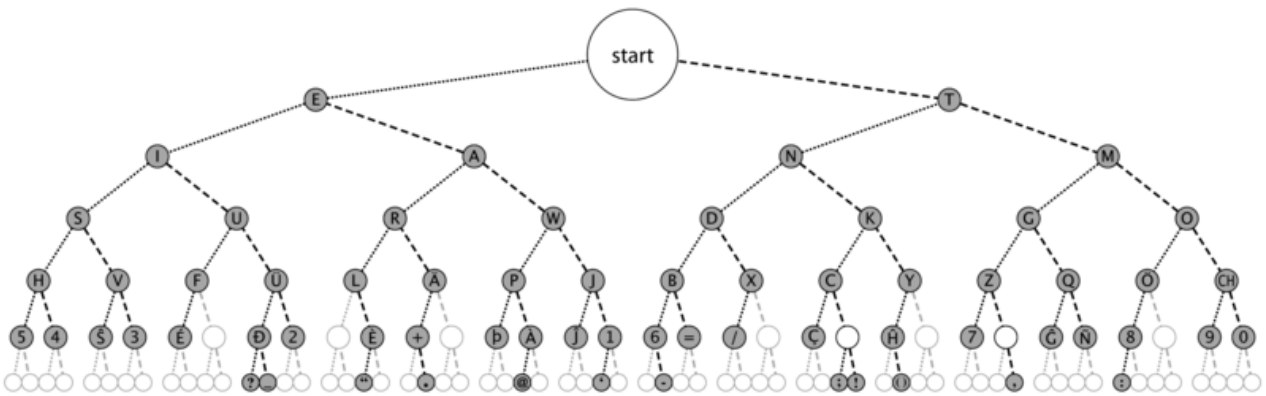
Aus all diesen Gründen hat sich der Morsecode in der Anfangszeit von Drahtloskommunikation sehr großer Beliebtheit erfreut (und wird auch heute noch – in abgewandelter Form – eingesetzt).

Das Projekt baut auf der Internationalen Definition des Morse-Codes auf. Eine detaillierte Übersicht und Spezifikation desselben kann auf http://de.wikipedia.org/wiki/Morsecode gefunden werden.

Zeitschema und Codierung:

- Der Morsecode verwendet nur die drei Symbole: Punkt („dot“, oft auch als „dit“ bezeichnet), Strich („dash“) und Pause („gap“). Ein „dash“ ist dreimal so lang wie ein „dot“.
- Die Pause zwischen zwei gesendeten Symbolen ist ein „dot“ lang.
- Zwischen Buchstaben in einem Wort wird eine Pause von einem „dash“ eingeschoben.
- Die Pause zwischen Wörtern beträgt sieben „dots“.
- Es gibt keine Unterscheidung zwischen Groß- und Kleinbuchstaben.

Der nachfolgende Baum gibt an, wie Buchstaben, Ziffern und Sonderzeichen aus dem Lateinischen Alphabet in das Morsealphabet übersetzt werden können. Jeder Ast in einem linken Teilbaum entspricht einem *dot*, ein Ast in einem rechten Teilbaum entspricht einem *dash*.



Beispiele:

- „I“ = .. (*dot dot*)
- „2“ = ..--- (*dot dot dash dash dash*)
- „M“ = -- (*dash dash*)

(Initial)Festlegung für das Projekt:

- Als Signalträger wird eine Audiosignal der Frequenz 2000Hz verwendet.
- Die Basis-Übertragungsrate soll 20 WPM (Words per Minute) betragen, das entspricht lt. Spezifikation einer „*dot“*-Länge von 60ms.

Aufgabenstellung für das Projekt:

In diesem Projekt ist ein vollständiger „Signalverarbeitungs-Kreis“ zu implementieren, wobei Ihnen die Wahl der Entwicklungsumgebung bzw. Programmsprache(n) frei steht.

Teil A) Morse-Codierung (24 Punkte)

- Einlesen eines Textes (Buchstaben oder Ziffern, Sonderzeichen sollten nicht enthalten sein und brauchen im Weiteren nicht berücksichtigt werden) aus einer Datei. Der Dateiname ist als Parameter (z.B. Kommandozeilen-Option) anzugeben. (4 Punkte).
- Codierung des Textes in den „Internationalen Morsecode“ mit folgendem Aufbau (10 Punkte):
  - Ein *„dot“* ist durch einen „.“ (Punkt) repräsentiert.
  - Ein *„dash“* ist durch einen „-“ (Strich) repräsentiert.
  - Ein *„gap“* (Zeichentrennung) ist durch ein einfaches „ “ (Leerzeichen) repräsentiert.
  - Ein Wortabstand ist durch „ “ (3 Leerzeichen) repräsentiert.
  - Morsecode (Bsp.): „- .... . .-. . .. .- - ....“ („There is a Th“...)
- Ausgabe des Morsecodes in eine Datei deren Name ebenfalls als Parameter anzugeben ist. Dazu gleichzeitige Ausgabe des Codes über die Lautsprecher als Tonfolge (entsprechend obiger Spezifikation). Zur Ausgabe eines 2kHz Tons können Sie auf dem Beispiel in den Übungsfolien aufbauen. Sehen Sie die Möglichkeit vor, mit unterschiedlichen Übertragungsraten zu experimentieren. Starten Sie mit einem 2kHz-Signal. Sie können später, zu Optimierungszwecken, mit beliebigen anderen Übertragungsraten (Signalfrequenzen) experimentieren. (Anmerkung: das Audiosignal wird als Eingabe für die nachfolgenden Prozessschritte verwendet).

Teil B) Morse-Decodierung (16 Punkte)

- Öffnen und Einlesen einer Datei (bis EOF) die einen Morsecode enthält, wie er in Teilaufgabe **A** erzeugt wurde (4 Punkte).
- Dekodierung des Dateiinhaltes („.“, „-“, „ “, „ “) in lateinische Buchstaben bzw. Ziffern (Sonderzeichen brauchen wiederum nicht berücksichtigt werden) (6 Punkte).
- Ausgabe des Klartextes in eine Datei und am Bildschirm. Ein mittels Teilaufgabe **A** codierter Text sollte vollständig wiederhergestellt werden können. (6 Punkte).

Teil C) Öffnen/Aufnehmen, Filtern, Ausgeben/Speichern (24)

- Implementieren Sie eine Funktion (hier als Vorschlag für Octave/Matlab; entsprechende Änderung bei nicht Umsetzbarkeit mit den gewählten Tools möglich)

```
processmorse (duration, dotLength, outfile, showplot, infile)
```

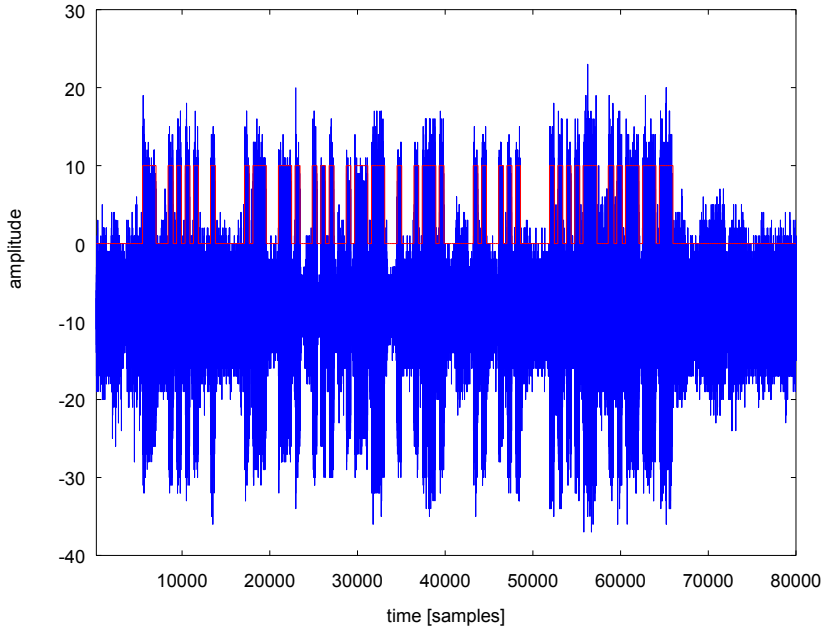
mit folgender Funktionalität:

(i) Die Funktion *„processmorse“* nimmt ein Audiosignal der Länge *„duration“* vom Mikrofon auf, *(3 Pkt.)*

(ii) filtert das Signal entsprechend der Übertragungsrate des Morsesignales (*„dotLength“; variable, um unterschiedliche Übertragungsraten zu decodieren*),

(iii) dekodiert das Signal in einen Morsecode, *(16 Pkt.[1])*
(iv) speichert diesen Code in der Datei *„outfile“*, und *(2 Pkt.)*
(v) stellt im Falle von *„showplot“* = „1“ (oder „TRUE“) das aufgenommene Audiosignal sowie das gefilterte Signal überlagert in einem Diagramm dar (siehe Bild unten). *(3 Pkt.)*

- Wird für *„outfile“* kein bzw. kein gültiger Dateiname angegeben, so wird das Ergebnis in keine Datei geschrieben.
- Wird für die Aufnahmedauer *„duration“* ein Wert „0“ angegeben, so ist das zu dekodierende Audiosignal aus der Datei *„infile“* zu lesen (andernfalls ist der Parameter *„infile“* zu ignorieren).
- Schreiben Sie einen detaillierten Hilfekommentar zur Verwendung der Funktion und dessen Parametern.
- Berücksichtigen Sie Fehlerbehandlung in den Funktionen (z.B. bei ungültiger oder fehlerhafter Angabe von Parametern, etc.).
- Der Übersichtlichkeit wegen sollte die Funktion *„processmorse“* selbst wieder Funktionen zum Lösen der Teilaufgaben aufrufen/verwenden, insbesonders für a) das Filtern des Signales und b) Dekodieren des gefilterten Signales in „.“ (*dot*), „-“ (*dash*) und „ “ (*gap,* Zeichentrennung, einfaches Leerzeichen) bzw. „ “ (*3-gap*, Worttrennung, 3 Leerzeichen).
- Die Plot-Ausgabe sollte mindestens das Ausgangssignal sowie das Endsignal nach der Filterung enthalten.



[1] Der Hauptaufwand der Übung steckt in den Funktionen „Signal filtern“ und „Dekodieren“. Je 4 Punkte werden für die korrekte Decodierung von „slow_clear.lin“, „slow_noise.lin“, „slow_insane.lin“ vergeben, je 1 Punkt für „fast_clear.lin“, „fast_noise.lin“, fast_insane.lin“.

Zum **Testen** der korrekten Funktion Teilaufgabe **C** (sowie für das Protokoll) verwenden Sie bitte folgende im Angabeordner bereitgestellten Audiodateien:

- Langsamer Morsecode: 10 Sekunden, 8kHz Samplingrate, Übertragungsrate (*„dot"*-Länge) 60ms
  (i) „slow_clear.lin" (geringes Rauschen)
  (ii) „slow_noise.lin" (mittleres Rauschen, „Musik")
  (iii) „slow_insane.lin" (starkes Rauschen, „Musik")
- Schneller Morsecode: 15 Sekunden, 8kHz Samplingrate, Übertragungsrate (*„dot"*-Länge) 2ms
  (i) „fast_clear.lin" (geringes Rauschen)
  (ii) „fast_noise.lin" (mittleres Rauschen, „Musik")
  (iii) „fast_insane.lin" (starkes Rauschen, „Musik")

Teil D) Projektpräsentation (8)

Für die Projektpräsentation benötigen Sie ein „Real-time"-System dass den gesamten Ablauf der Morsecodierung / Decodierung autonom und in real-time durchführt. Entwickeln bzw. kombinieren Sie ihre Teilprogramme (Teilaufgaben **A**, **B**, **C**) so, dass sie den real-time Aspekt nach folgendem Schema erfüllen (Hinweis = Ablauf des Contests):

- jede Gruppe bekommt (die gleiche) Textdatei zum kodieren/übertragen/dekodieren.
- die Signalverarbeitung muss auf 2 Rechnern (gleiche Distanz zueinander für alle Gruppen) durchgeführt werden. **A** liest Datei, kodiert den Inhalt, überträgt das Signal via Lautsprecher; **B** empfängt das Audio-Signal, speichert es, dekodiert es mit Octave/Matlab, etc., und zeigt das Ergebnis an bzw. speichert dieses in eine Datei.
- Zeitmessung (beim korrekte Ergebnis) ergibt Platzierung beim „Contest", dh. finden sie einen optimalen Mittelweg zwischen „schnell" und „präzise".
- Die Projektpräsentation findet für alle Gruppen am 18. Juni 2013 zur Übungszeit (17.15 – ca. 19.00 Uhr) in HS 19 statt. **Eine Teilnahme ist verpflichtend!**

Geben Sie ein „technisches Protokoll" im LaTeX-/Word- oder PDF-Format inkl. Deckblatt (=ausgefülltes Übungsangabeblatt) ab. Dieses hat zu enthalten:

- Für die beiden Teilaufgaben A und B ist eine schrittweise Lösungsbeschreibung, der entwickelte Source-Code, Screenshots der Applikation (beim Testen von Teil A bzw. B), einige Testfälle mit eingegebenem Text und ausgegebenem Morsecode (Teil A) bzw. dekodierter Text der bereitgestellten Eingabesamples (Teil B) zu protokollieren.
- **Geben Sie explizit an, welche Signalübertragungsfrequenz und Übertragungsrate (dot-length) Sie gewählt haben, und warum.**
- Für Teilaufgabe C ist in das Protokoll eine schrittweise Lösungsbeschreibung sowie die Ausgabedateien für die 6 oben angegebenen Audiodateien bzw. für einige unter Verwendung des Programmes aus Teilaufgaben A und B generierten Morsesignalen aufzunehmen. Weiters sind Screenshots der Plots zu integrieren.
- **Beschreiben Sie, wie Sie den Filtern designed haben bzw. wie Sie zu den Parametern gekommen sind.**
- Sämtliche Quelldateien (Java, Matlab/Octave, etc.) sind abzugeben. Bei fehlenden Sourcen werden die entsprechenden Teile der Übung mit 0 Punkten bewertet.
- Die gesamte Übung ist in ein Archiv zu packen:

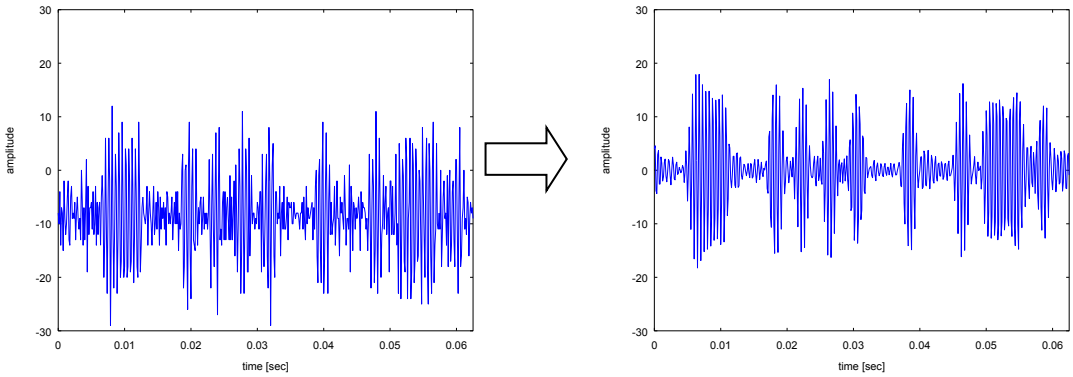
Jede abgegebene Datei muss eindeutig zuordenbar sein, beachten Sie deshalb die Namenskonventionen bei der Übungsabgabe. Der Dateiname (.arj, .zip) hat zu enthalten:

- **Übungsnummer, Matrikelnummer, SKZ, Nachname für beide Studenten**
- **keine Umlaute und Sonderzeichen im Dateinamen**
- **Bsp.: xxxx_Matr_SKZ_Nachname_Matr_SKZ_Nachname**
  PR_0555489_521_Riener_02333122_521_Hoelzl

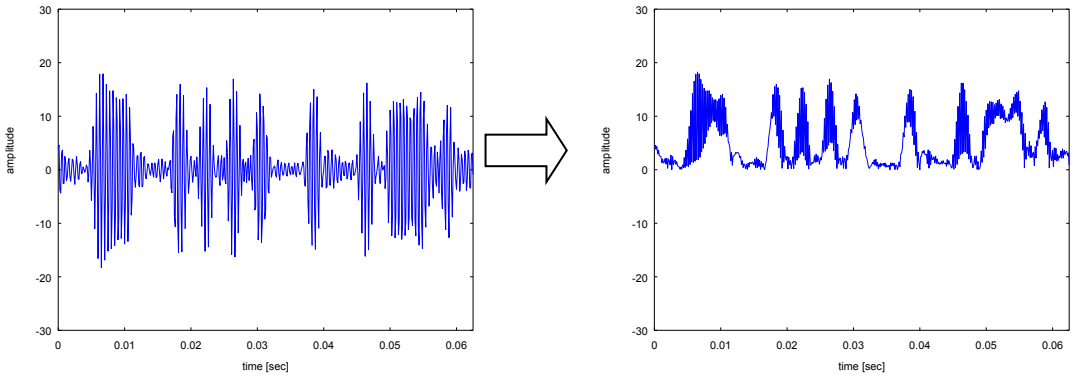
Hinweise zur Implementierung von Filtern:

Für das Projekt besteht das „Filtern“ aus 4 Schritten (Beispielbilder aus „fast_insane.lin“):

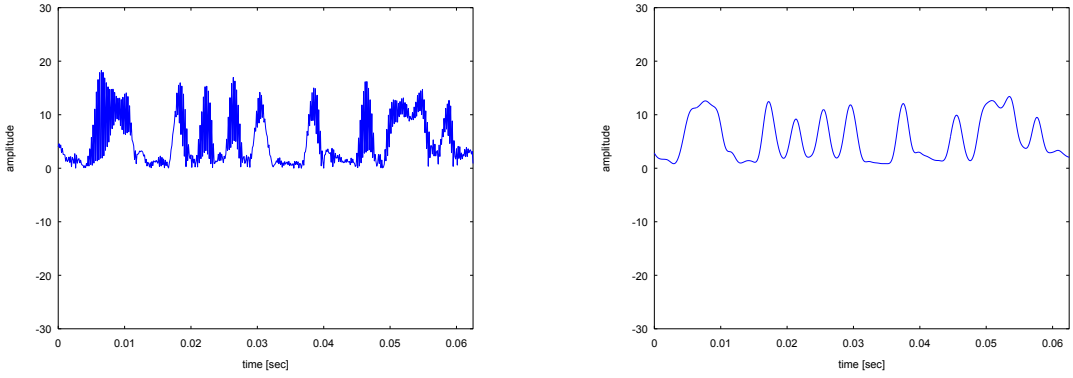
- Rauschen/Artefakte aus dem Signal entfernen:
  - Signalbasis nach 0 verschieben
  - Verwendung eines FIR Bandpass-Filters bei 2kHz



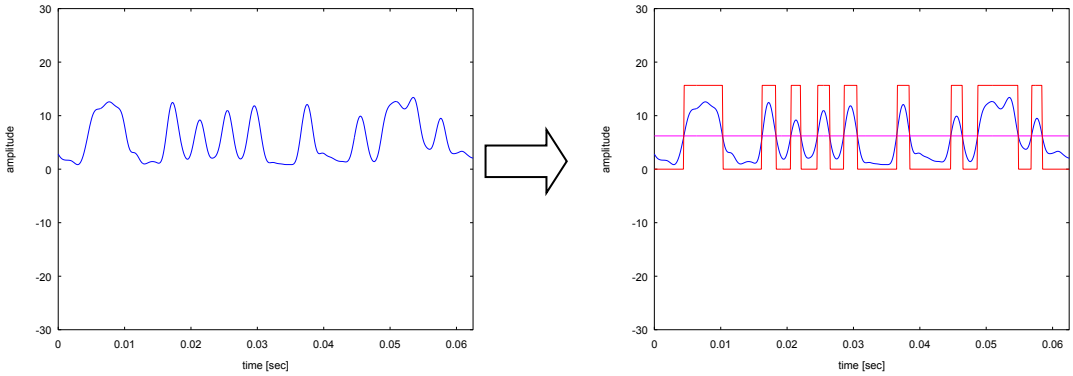
- Absolute Amplitude des Trägersignals isolieren:
  - Optimal wäre Multiplikation mit Trägersignal in korrekter Phase. Da wir nur mit einem isolierten Trägersignal arbeiten, reicht es, dieses in absolute Werte zu wandeln (abs(signal)).



- Demodulation:
  - Einsatz eines FIR Tiefpass-Filters bei der Übertragungsfrequenz
  (z.B. bei 60 ms *„dot“*-Länge: 1000 / 60 = 16.6 Hz)



- Diskretisierung:
  - An geeigneter Stelle in 0 / 1 auftrennen



Hinweis zur Implementierung der Decodierungsfunktion:
Aus dem gefiltertem Signal ist eine Folge von „Beeps“ und „Pausen“ mit ihrer jeweiligen Dauer zu ermitteln und entsprechend der Übertragungsrate als *„dot“, „dash“, „gap“ oder „3-gap“* zu interpretieren.

Beispiel (Samplingrate 8000 Hz, Übertragungsrate (*„dot“*-Länge) 60ms):

- Beep mit 1435 Samples, Dauer = 1435/8000 ~ 179.3 ms → entspricht *„dash“*
- Pause mit 1467 Samples, Dauer = 1467/8000 ~ 183.3 ms → entspricht *„gap“*
- Beep mit 467 Samples, Dauer = 467/8000 ~ 58.3 ms → entspricht *„dot“*
- …