Fachbereich für Informationstechnik

Projektarbeit

Name: Ramon Michel Leber

Thema: Evaluierung des Mini-Computer Banana Pi
als Grundlage für ein Spracherkennungssystem

Dozent: Prof. Dr. Alznauer

Abgabetermin: 09.02.2015

Kurzfassung

Die vorliegende Projektarbeit befasst sich mit den Möglichkeiten die ein Einplatinencomputer heute eröffnet. Deshalb werden gängige und etablierte Mini-Computer vorgestellt und deren Einsatztauglichkeit anhand einer technischen Aufgabe geprüft.

Die jeweilige Hardware der einzelnen Mini-Rechner wird verglichen und die dafür verwendbaren Betriebssysteme vorgestellt.

Verschiedene Lösungswege werden aufgezeigt, die den Mini-Computer zu einem Helfer für kleine und größere Aufgaben machen.

Schlagwörter: Projektarbeit, Mini-Rechner, Raspberry Pi, Banana Pi,

Abstract

The presented student project concerns the possibilities of a single board computer. Therefore some of the most common single board devices will be introduced and their possibilities will be applied to a technical task.

The hardware of each device will be compared and all usable operating system will be introduced.

Different solutions will be shown, that make a single board computer an usefull assistant for small and medium size tasks.

Keywords: Student project, Smart Device, Raspberry Pi, Banana Pi

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In der heutigen Zeit sind Mini-Computer und Einplatinencomputer immer weiter verbreitet. Ihren Siegeszug traten die kleinen Rechner besonders dadurch an, dass sie besonders kostengünstig zu erwerben, und einfach zu verwenden sind. Besonders das Raspberry Pi, welches von der Fachpresse auch ausgiebig gefeiert wird, ist weit verbreitet.

Das System verspricht eine einfache Programmierung und Inbetriebnahme für unzählige Funktionen, bei einem Anschaffungspreis von gerade mal circa 40€. Ob als Media-Server oder zur mobile home Anwendung, bis zum Linux-Lite-Rechner scheint dem Gerät kaum eine Grenze gesetzt zu sein.

Doch das britische System beherrscht schon lange nicht mehr alleine den Markt der Mini-Rechner. So gibt es seit dem Jahr 2014 auf dem deutschen Markt auch einen chinesischen Ableger des Raspberry Pi zu kaufen, das Banana Pi.

Das Banana Pi besticht mit einer noch großzügiger dimensionierten Hardware und mehreren Betriebssystem Optionen und bietet damit noch mehr Möglichkeiten als das Raspberry, bei einem vergleichbaren Anschaffungspreis. Dies ist jedoch noch lange nicht das Ende der Fahnenstange. Cubiboard, Ethernut und PandaBoard... die Liste der modernen Einplatinencomputer ist lange und nicht jedes der Systeme ist gleich gut für bestimmte Anwendungen zu verwenden.

Um die Nutzbarkeit dieser Systeme mit einer sinnvollen Anwendung zu überprüfen, werden in dieser Projektarbeit das Raspberry- und das Banana Pi als Spracherkennungssysteme verwendet.

2 Problemstellung

2.1 Ausgangssituation

Die Gesellschaft ist im Begriff im älter zu werden. Durch diesen demografischen Wandel ergeben sich auch immer neue Probleme mit denen sich unsere Gesellschaft auseinandersetzen muss. Eines davon ist, dass die Zahl der Gehörbeeinträchtigten und Gehörlosen immer weiter ansteigen wird. Dadurch steigt natürlich auch das Bedürfnis nach medizinischen Lösungen, oder aber auch nach Techniken und Geräten, welche für diese Probleme eine Hilfestellung bieten können.

Natürlich ist es möglich mit Hilfe von Hörgeräten diesem Problem Abhilfe zu schaffen. Doch durch schlechte Klangqualität und schlechtem Tragekomfort kommt es nicht selten zu Kopfschmerzen und Unwohlsein bei den Patienten.

[1] Bedingt durch die Bauform ergibt sich bei einigen Hörgeräten nur eine geringe Belüftungsmöglichkeit. Dieses so genannte Venting sorgt für einen Druckausgleich. Findet dieser nicht in ausreichendem Maße statt, kann sich die eigene Stimme oder Körperschall (zum Beispiel durch Kauen) unnatürlich und unangenehm anhören. Außerdem sorgt der Verschlusseffekt für ein häufigeres Auftreten von Rückkopplungseffekten. Letzteres wird zudem durch die Tatsache begünstigt, dass der Abstand zwischen Mikrofon und Hörer bauartbedingt besonders klein ist.

Des Weiteren sind einige Frequenzen schwer von Hörgeräten zu erfassen beziehungsweise schwer wiederzugeben. So werden hohe Töne oftmals vom Gerät überhört oder sehr schrill wiedergegeben. Tiefe Töne hingegen sind oftmals unnatürlich lauter als hohe Töne in gleicher Lautstärke. Dies erzeugt teilweise eine sehr verzerrte Wiedergabe der Umgebungsgeräusche.

Hörgeräte sind meist rein auf die Wiedergabe von Sprache konzipiert. Auf die Wiedergaben von Tönen wie Haustürklingel oder Telefon sind die Hörgeräte oftmals nicht eingestellt.

Viele Patienten resignieren nach einiger Zeit mit einem Hörgerät und verzichten völlig darauf. In dem Fall bleibt dem Patienten nur noch die Möglichkeit, sich im Lippenlesen zu üben um sich weiter an einer Unterhaltung beteiligen zu können.
Durch die Verbreitung neuer Minicomputer ergeben sich auch neue Möglichkeiten und neue Ansätze, wie man kleine technische Helfer in der Medizintechnik oder im Alltag einsetzen kann. Das Raspberry Pi hat sich als Experimentier-

computer bereits etabliert. Jedoch wird das System zum größten Teil nur für Entertainment Zwecke verwendet.

2.2 Zielsetzung

Um zu evaluieren wo genau die Grenzen von Raspberry Pi und Banana Pi liegen, soll mit den beiden Systemen versucht werden, ein Spracherkennungssystem zu gestalten, welches Hörgeschädigten und Gehörlosen eine Unterstützung bietet, ohne Hörgeräte an einer Unterhaltung teilzunehmen. Das System soll durch gute Spracherkennung möglichst exakt die Unterhaltung auf einem Display wiedergeben, damit alles von den Beteiligten abgelesen werden kann.

Die Fragen die sich dabei stellen sind unter anderem:

- Welche Spracherkennungssysteme gibt es für die Minicomputer
- Wie gut sind diese Spracherkennungssysteme (bspw. Dialekt)
- Welche Displays sind mit den Minicomputern kompatibel
- Welche Mikrofone eignen sich für den Einsatz bei dieser Anwendung überhaupt
- Was ist der Mehrwert der gesamten Anwendung gegenüber von Hörgeräten

Im besten Fall soll die Anwendung ein Hörgerät ersetzen können und dabei individuell erweitert werden können, auf spezifische Aufgaben wie zum Beispiel das Erkennen von Haustür- und Telefonklingel.

3 Single Board Computer

[2] Bei einem Single Board Computer sind alle elektronischen Bauteile die zum Betrieb benötigt werden auf einer Platine zusammengefasst. In der Industrie werden diese Systeme vorwiegend in der Mess-, Steuer- und Regelungstechnik eingesetzt, da sie wesentlich billiger sind als speicherprogrammierbare Steuerungen (SPS).
Einplatinencomputer wie der KIM-1, der Microprofessor I und der Apple I bildeten ab Mitte der 1970er Jahre eine der Vorstufen der späteren Heimcomputer. Sie wurden oft von Computerenthusiasten und Hackern gebaut und verwendet, später ging ihre Verbreitung bei Privatnutzern stark zurück. Konstruktiv gesehen können auch die meisten Heimcomputer der 1980er wie der Sinclair ZX81, der Commodore 64 oder der Atari ST als Einplatinencomputer angesehen werden, wurden jedoch in der Regel nicht so bezeichnet.
Durch die Entwicklung des Raspberry Pi haben seit 2012 die Single Board Computer wieder zunehmend Verbreitung im Privatbereich gefunden. Das Raspberry Pi wurde bis zum Juni 2014 etwa 3 Millionen mal verkauft.

3.1 Raspberry Pi

[3] Das von der britischen Raspberry Pi Foundation entwickelte Raspberry Pi hatte seit der Einführung 2012 einen großen Markterfolg zu verzeichnen. Die Motivation hinter der Entwicklung war die Tatsache, dass die Anzahl der Informatikstudenten an der Universität Cambridge immer weiter zurückgegangen war. Außerdem hatten die Studienanfänger jedes Jahr geringere Programmierkenntnisse.

Die Initiatoren des Raspberry Pi machten dafür die teuren und komplexen Heimrechner verantwortlich, zu denen die Jugendlichen oftmals keinen Zugriff hatten. Deshalb hatte sich die Stiftung der Raspberry Pi Foundation zum Ziel gesetzt, das Studium der Informatik und verwandter Themen zu fördern.

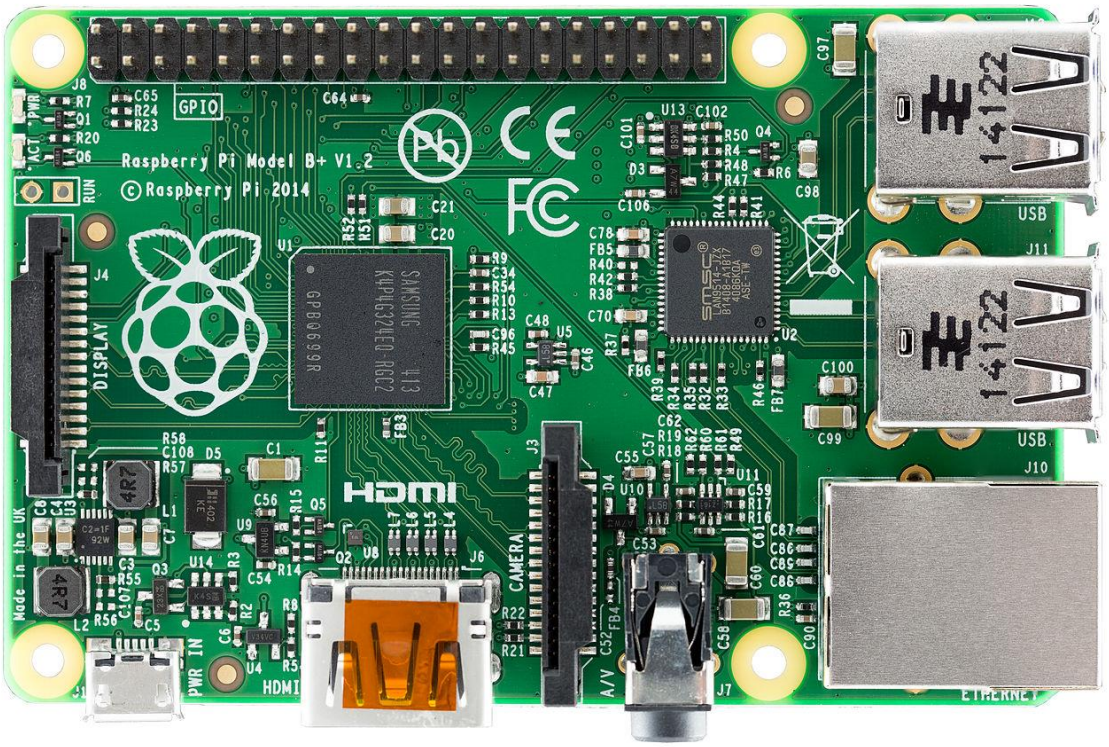


Abbildung 1: Raspberry Pi [3]

Das in dieser Projektarbeit verwendete Raspberry Pi Modell war folgendermaßen zusammengestellt:

- Preis circa 40€
- SoC: Broadcom BCM2835
- CPU: ARM1176JZF-S / 1 Kern / 700MHz
- Arbeitsspeicher: SDRAM 512MB
- 4 USB-2.0-Anschlüsse
- Videoausgabe: Composite Video, HDMI
- Tonausgabe: 3,5mm-Klinkenstecker
- Festspeicher: Kartenleser für SD
- Netzwerk: 10/100-MBit-Ethernet
- 26 GPIO-Pins

Als Betriebssysteme sind einige Open-Source-Betriebssysteme verfügbar. Das Linux-System Raspbian, welches auf Debian Wheezy basiert, ist das am häufigsten verwendete Betriebssystem für das Raspberry Pi. Um das Raspberry als Media-Center zu verwenden kann das Betriebssystem XMBC verwendet werden. Mit XMBC lässt sich das Raspberry auch als Fernbedienung verwenden. Neben den Linux-Betriebssystemen gibt es unter anderem auch eine Android-Beta-Version, welche allerdings nur sehr eingeschränkt funktionsfähig ist.

3.2 Banana Pi

[4] Das Banana Pi wurde von der chinesischen Bildungsinitiative Lemaker.org entwickelt und ist seit März 2014 zu erwerben.

Abbildung 2: Banana Pi [4]

Das Banana Pi hat folgende Spezifikationen:

- SoC: Allwinner A20
- CPU: ARM-Cortex-A7 / 2 Kerne / 1GHz
- Grafikprozessor: Mali-400MP2-GPU
- Arbeitsspeicher: 1 GB DDR3-SDRAM
- Festspeicher: MicroSD, SATA-Port
- Videoausgang: HDMI, Composite Video
- Netzwerk: 10/100/1000-Mbit/s-Ethernet
- USB: 2 USB-Hosts
- 40 GPIO-Pins
- CSI-Kamera-Anschluss
- DSI-Displayanschluss

Als Betriebssysteme sind neben den Linux-Systemen wie das auf Debian basierenden Bananian, Lubuntu for Banana Pi und ArchLinux for Banana Pi, auch das Media-Betriebssystem XBMC verfügbar. Der große Vorteil des Banana Pi

ist, dass auch eine Lauffähige Android Version zur Verfügung steht. Das Android auf dem Banana Pi 4.2.2 bietet alle Funktionen, die auch auf einem Android-Tablet verfügbar sind.

4 Spracherkennungssysteme auf Single Board Computer

Abbildung 3: Ext. Soundkarte [9]

Um überhaupt ein Spracherkennungssystem auf einem Single Board Computer in Betrieb nehmen zu können, müssen einige Grundeinstellungen vorgenommen werden. Zu aller erst müssen Raspberry- und Banana Pi um eine externe Soundkarte erweitert werden, da das Raspberry Pi nur mit einem Mikrofonausgang ausgestattet ist und das Banana Pi ein Mikrofon auf der Platine zwar installiert hat, dieses aber in eingebautem Zustand nicht mehr ansprechbar ist. Mit dem Banana Pi funktioniert die externe Soundkarte per Plug-and Play, wobei mit dem Raspberry Pi, unter Verwendung des Linux basierten Raspbian Betriebssystems eine Konfiguration vorgenommen werden muss.

Zunächst muss der Soundtreiber mit der Linux-Kommandozeile geladen werden. Dies geschieht mit folgendem Befehl:

```
sudo modprobe snd_bcm2835
```

Anschließend muss die alsa-base.conf Datei konfiguriert werden:

```
sudo nano /etc/modprobe.d/alsa-base.conf
```

Hier muss der audio index von -2 auf 1 geändert werden:

```
options snd-usb-audio index=1
```

Danach muss der Standardausgang festgelegt werden:

```
sudo nano /etc/asound.conf pcm.!default { 2 type plug 3 slave { 4 pcm “hw:1,0“ 5 } 6 } 7 ctl.!default { 8 type hw 9 card 1 10 }
```

Abschließend muss ein Neustart durchgeführt werden:

```
sudo reboot
```

Weitere Einstellungen wie Mikrofon- und Lautsprecherlautstärke können im alsamixer vorgenommen werden.

4.1 Raspberry Pi mit CMU Sphinx

Sphinx ist ein endlos Spracherkennungssystem und bietet mit dem Ableger PocketSphinx eine Anwendung für eingebettete Systeme die einen ARM-Prozessor besitzen. In dem Projekt “Jasper“ hat sich PocketSphinx bereits auf dem Raspberry bewährt. Jasper ist ein Kommandosteuerungsprogramm. Die Anwendung wartet ständig auf Sprachbefehle und kann so per Kommando das Licht an- und ausschalten oder einen bestimmten Musiktitel abspielen.

[5] PocketSphinx wird folgendermaßen mit der Linux-Kommandozeile auf das Raspberry Pi heruntergeladen, installiert und ausgeführt:

Zuerst muss ein neues Verzeichnis erstellt werden:

```
mkdir cmusphinx
```

um anschließend in das erstellte Verzeichnis zu wechseln:

```
cd cmusphinx
```

Danach werden PocketSphinx und Sphinxbase heruntergeladen und entpackt:

```
wget sphinxbase-0.8 und pocketsphinx-0.8
tar –xvzf pocketsphinx-0.8.tar.gz
tar –xvzf sphinxbase-0.8.tar.gz
```

Anschließend wird Sphinxbase installiert:

```
cd sphinxbase-0.8
./autogen.sh
./configure
make
sudo make install
cd –
```

Danach wird PocketSphinx installiert:

```
cd pocketsphinx-0.8
export LD_LIBRARY_PATH=/usr/local/lib
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
./configure
make
sudo make install
```

Letztendlich wird PocketSphinx ausgeführt:

```
pocketsphinx_continuous
```

Nach Ausführung des letzten Befehls wartet das System nun dauerhaft auf Spracheingaben. Ein großer Vorteil gegenüber anderer Spracherkennungssysteme ist bei PocketSphinx, dass die ganze Datenverarbeitung offline geschieht und man mit der Anwendung sehr mobil sein kann. Des Weiteren können die Sprachdaten nicht von Dritten mitgehört und weiterverwendet werden.

Ein Nachteil der Anwendung ist, dass die erwartete schnelle Sprachverarbeitung durch den lokalen Verarbeitungsprozess, leider aus bleibt. Auch sind die Ergebnisse der Spracheingaben nicht zufriedenstellend in Anbetracht der Zielstellung der Projektarbeit.

Im Folgenden einige Sprachbeispiele:

Spracheingabe:	Verarbeitungszeit:	Textausgabe:
“Hallo mein Name ist Ramon Leber“	22 Sekunden	“Hallo mein“
“Guten Tag“	16 Sekunden	“Guten Tag“
“Was gibt es heute zu Essen“	26 Sekunden	“Was“
“Die Würde des Menschen ist unantastbar“	40 Sekunden/Abbruch	- ergebnislos
“Drei mal drei ist neun“	28 Sekunden	“mal“

Die Textausgabe erfolgt auf der Konsole des Raspberry Pi und hat in allen Tests nie mehr als zwei Worte beinhaltet.

Fazit: Als Grundlage für eine Kommandosteuerungsanwendung, wo die Erfassung einzelner Worte ausreichend ist, kann PocketSphinx durchaus sinnvoll eingesetzt werden, falls die langen Verarbeitungszeiten keine Probleme darstellen oder als störend empfunden werden. Für den Einsatz in einem zuverlässigen Spracherkennungssystem, welches ganze, zusammenhängende Sätze in möglichst kurzer Zeit erfassen soll, ist PocketSphinx gänzlich ungeeignet.

4.2 Raspberry Pi mit Google Speech API v2

Ein weiterer Ansatz um eine qualitative Spracherkennung zu realisieren, bietet die Google Speech API. Google stellt den Zugriff auf die Programmierschnittstelle zur Verfügung. Jedoch muss ein Entwickler sich zuvor bei Google anmelden um einen Key zugestellt zu bekommen. Mit diesem Key muss sich jeder Nutzer der Google API später Zugang auf den Google-Server verschaffen um die Sprachdienste nutzen zu können. Alle Sprachdateien werden über Ethernet an Google gesendet und setzt einen Internetzugang des Nutzers voraus. Was Google des Weiteren mit den Audio Dateien macht ist unklar.

[6] Folgendermaßen wird die Google Speech API v2 auf dem Raspberry Pi installiert und verwendet. Zuerst muss mittels Linux-Kommandozeile der free-lossless-audio-codec installiert werden. Damit werden die Audio-Dateien die an Google gesendet werden sollen, komprimiert:

```
sudo apt-get install flac

sudo apt-get install python-pycurl
```

Ein Skript der folgenden Form soll die Anfrage an den Google-Server übernehmen:

```
#!/bin/bash

KEY=“DEN_GOOGLE_KEY_HIER_EINTRAGEN“

URL=https://www.google.com/speech-
api/v2/recognize?output=json&lang=de&key=$KEY

echo “Recording... Press Ctrl+C once to Stop and WAIT.“
```

```
Arecord –D plughw:0,0 –f cd –t wav –d 0 –q –r 16000 I flac - -s –f –best –sample-rate 16000 –o file.flac;

echo “Processing...“

wget –q –U “Mozilla/5.0“ –post-file file.flac –header “Content-Type: audio/x-flac; rate=16000“ –o – “$URL“>

echo –n “Google reply: “

cat stt.txt

rm file.flac > /dev/null 2>&1
```

Das Skript muss unter dem Namen “text2speech.sh“ gespeichert werden und kann mittels “$ chmod +x text2speech.sh“ ausführbar gemacht werden, um es letztendlich mit “./speech2text.sh“ auszuführen.

Als Parameter werden der URL innerhalb des Skripts neben dem Google Key auch die Sprache und das erwartete Rückgabeformat der Textdatei übergeben.

Im Folgenden wieder einige Sprachbeispiele:

Spracheingabe:	Verarbeitungszeit:	Textausgabe:
“Hallo mein Name ist Ramon Leber“	28 Sekunden	“Hallo mein Name ist“
“Guten Tag“	31 Sekunden	“Guten Tag“
“Was gibt es heute zu Essen“	32 Sekunden	“Was gibt es heute zu Essen“
“Die Würde des Menschen ist unantastbar“	27 Sekunden	“Die würdest du denken ist unfassbar“
“Drei mal drei ist neun“	19 Sekunden	“Mal 3 isst heute“

Fazit: Die Spracherkennung ist minimal besser, ganze Sätze lassen sich theoretisch erkennen. Die Verarbeitung dauert noch immer sehr lange und der Nutzer muss ständig an das Internet angeschlossen sein um die Google-Dienste

nutzen zu können. Das System ist für die Zielsetzung dieser Projektarbeit noch immer nicht ausreichend.

4.3 Raspberry Pi Display

Um ein vernünftiges Ergebnis einer Spracherkennung auch Anzeigen lassen zu können, benötigt man des Weiteren ein kompatibles Display für das Raspberry Pi. Die Firma Admatec bietet hierfür ein 3,5 Zoll Display mit dem Namen C-Berry an, welches über die GPIO Pins des Computers angeschlossen werden kann.

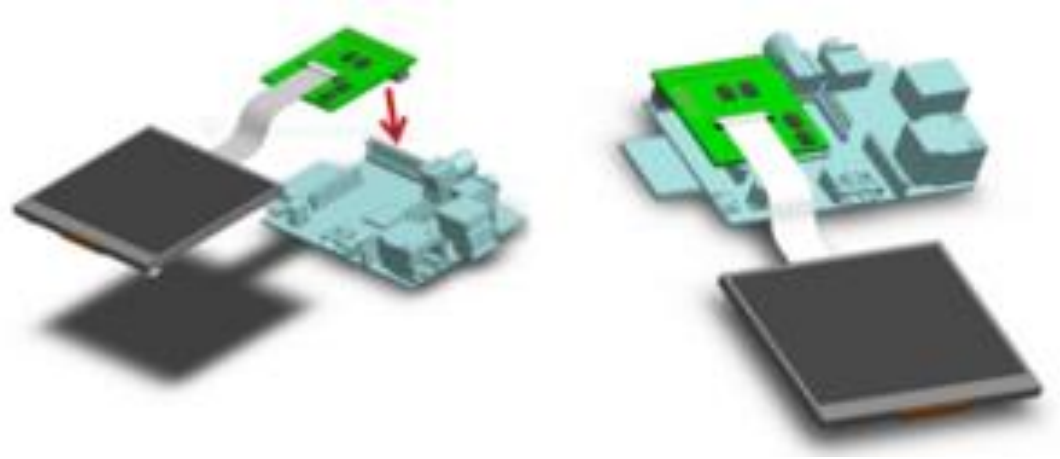


Abbildung 4: C-Berry Modul [7]

Die Ergebnisse der Spracherkennung aus der Anwendung mit PocketSphinx, sowie der Google Speech API, wurden bisher nur auf der Kommandozeile ausgegeben.

Da das C-Berry über die GPIO Pins angeschlossen ist, wird es nicht wie ein Bildschirm erkannt. Um grafische Ausgaben auf dem Display realisieren zu können muss ein eigenes Programm für die Grafik erstellt werden.

Außerdem müssen die Ergebnisse der Spracherkennung auf der Kommandozeile erst als Strings separiert werden, um anschließend auf dem Display ausgegeben zu werden.

Zuletzt stellt sich noch die Frage, wie das Raspberry Pi in Kombination mit dem C-Berry bedient werden kann. Das C-Berry ist ein reines Anzeigedisplay ohne Touchfunktion.

Alle diese Erkenntnisse führen zu der Schlussfolgerung, dass eine qualitative Spracherkennung mit Textausgabe für Gehörgeschädigte, mit dem Raspberry Pi nicht durchführbar ist.

4.4 Banana Pi mit Android und Google Offline Spracherkennung

4.4.1 Allgemein

Da eine Realisierung mit dem Raspberry Pi als sehr aufwändig und qualitativ wenig erfolgsversprechend ist, müssen zunächst einige Vorüberlegungen angestellt werden, um den richtigen Single-Board-Computer auszuwählen.

Die bislang angestellten Ansätze ergaben, dass Linux-Betriebssysteme nicht geeignet sind für eine qualitative Spracherkennung. Außerdem ist ein Ansatz, der den permanenten Anschluss an das Internet voraussetzt wenig sinnvoll. Wichtig ist auch, wie das Ergebnis der Spracherkennung angezeigt werden kann. Es muss also auch eine kostengünstige Lösung geben wie das System gesteuert werden kann und wie mit möglichst geringem Aufwand, die Ergebnisse ausgegeben werden können.

Da auf jedem Android-Gerät eine verhältnismäßig zuverlässige Spracherkennungsanwendung zur Verfügung steht kommt nur ein Computer in Frage, welcher eine uneingeschränkt funktionsfähige Android Version als Betriebssystem nutzen kann. Dies ist unter anderem mit dem Cubieboard oder dem Banana Pi möglich.

Die zweite Überlegung ist nun, welcher der Computer kompatibel mit einem touchfähigen Display ist, welches mit möglichst geringem Aufwand installiert werden kann.

Die Firma Pollin bietet ein 7 Zoll LCD Touchscreen an, welches mit einer Grafikkarte problemlos über HDMI/DVI/VGA/CVBS als Computerbildschirm verwendet werden kann.

DasTouchmodul des Displays wird einfach über USB an das Banana Pi angeschlossen. Weitere Erläuterungen zum Anschluss des Displays finden sich im Anhang 1.

Das Cubieboard ist eventuell ebenso mit dem 7 Zoll Touchscreen der Firma Pollin verwendbar, jedoch ist das Banana Pi kostengünstiger und wurde in diversen Entwicklerforen mehrfach in Verbindung mit dem Pollin Display vorgestellt und diskutiert.

4.4.2 Die Spracherkennungsanwendung - Analyse

Google ermöglicht den Nutzern von Android den Zugriff auf die Spracherkennungsdienste der Android-Geräte. Um eine eigene Anwendung zu erstellen, müssen zunächst die Anwendungsfälle erkannt und spezifiziert werden.

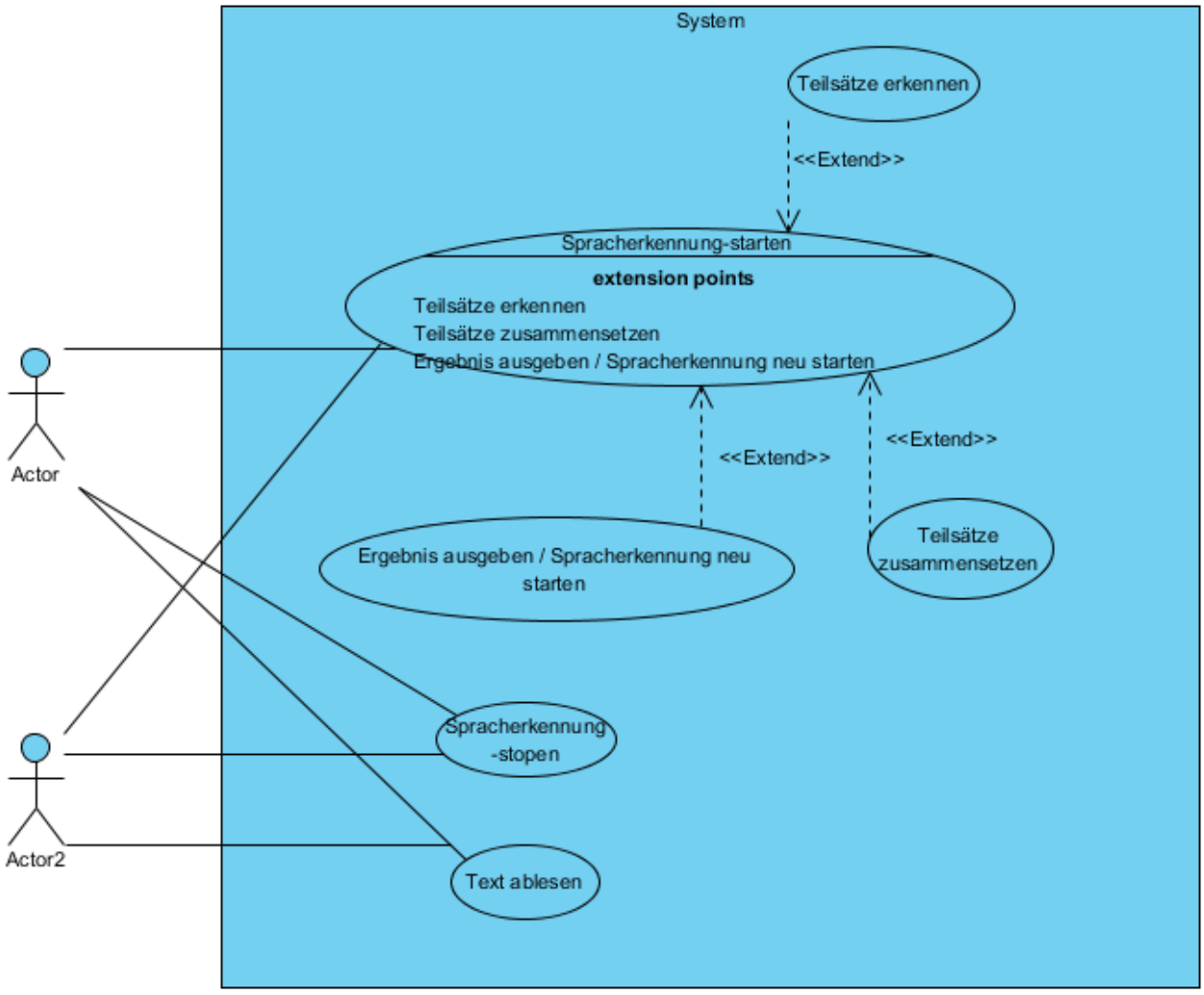


Abbildung 5: Use-Case-Diagramm Spracherkennung

Einer der Aktoren startet die Anwendung. Die Aktoren führen eine Unterhaltung, während die Anwendung ständig Teilsätze erkennt und diese auf dem Display ausgibt. Nachdem mehrere Teilsätze erkannt wurden, werden diese zu einem ganzen Satz zusammengefügt und ausgegeben. Die Anwendung stoppt für ei-

nen kurzen Moment und startet sich anschließend neu. Dies geschieht so lange, bis einer der Aktoren die Spracherkennung beendet.

Das Layout der Anwendung wird in XML definiert. Hier soll nur eine TextView erstellt werden, welche die Ausgabe der Spracherkennung anzeigt. Die TextView muss so gestaltet sein, dass sie bei fortlaufendem Text automatisch weiter scrollt. Dazu muss das ganze Layout in einer so genannten </ScrollView> stehen. Die TextView ist einfach durch Größe, Farbe und Textstyle definiert, und bekommt einen String per id zugewiesen, welcher in der TextView ausgegeben wird.

```
<TextView
    android:id="@+id/txtSpeechInputField"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:maxWidth = "950dp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="100dp"
    android:textColor="@color/black"
    android:textSize="40dp"
    android:textStyle="normal"
    android:gravity="bottom" />
```

Außerdem wird ein Button in Form eines Mikrofons erzeugt welcher die Anwendung startet und stoppt. Der Button bekommt per id ein handle übergeben und ist sonst durch Größe und Position, sowie dem Mikrofonimage welches als Button dient definiert.

```
<ImageButton
      android:id="@+id/btnSpeak"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginBottom="1dp"
      android:background="@null"
      android:src="@drawable/microphone_icon" />
```

Das AndroidManifest.xml dient zur Spezifikation einiger Systemfunktionen. Unter anderem müssen hier Rechte der Anwendung zugewiesen werden. Für die Anwendung der Spracherkennung muss das Recht erteilt werden, Audio-Dateien aufzunehmen. Dies geschieht durch folgende Deklaration:

```
<uses-permission
android:name="android.permission.RECORD_AUDIO" />
```

Damit sind die Grundeinstellungen vorgenommen und das Layout ist nach den gewünschten Anforderungen erstellt.

Um nun die Spracherkennungsdienste von Google nutzen zu können, muss die Klasse SpeechRecognizer implementiert werden. [8] Diese Klasse dient dazu, Zugriff auf den Spracherkennungsdienste zu erhalten. Unter anderem können durch den SpeechRecognizer Audio Dateien auf entfernte Server gesendet werden, falls keine offline Spracherkennung installiert und initialisiert wurde. Die Schnittstelle ist nicht darauf ausgelegt, eine dauerhafte Spracherkennung durchzuführen, da dies sehr die Batterieleistung und Bandbreite der Android-Geräte beanspruchen würde. Aus diesem Grund beendet sich die Spracherkennung nach einiger Zeit von alleine.

4.4.3 Die Spracherkennungsanwendung - Design

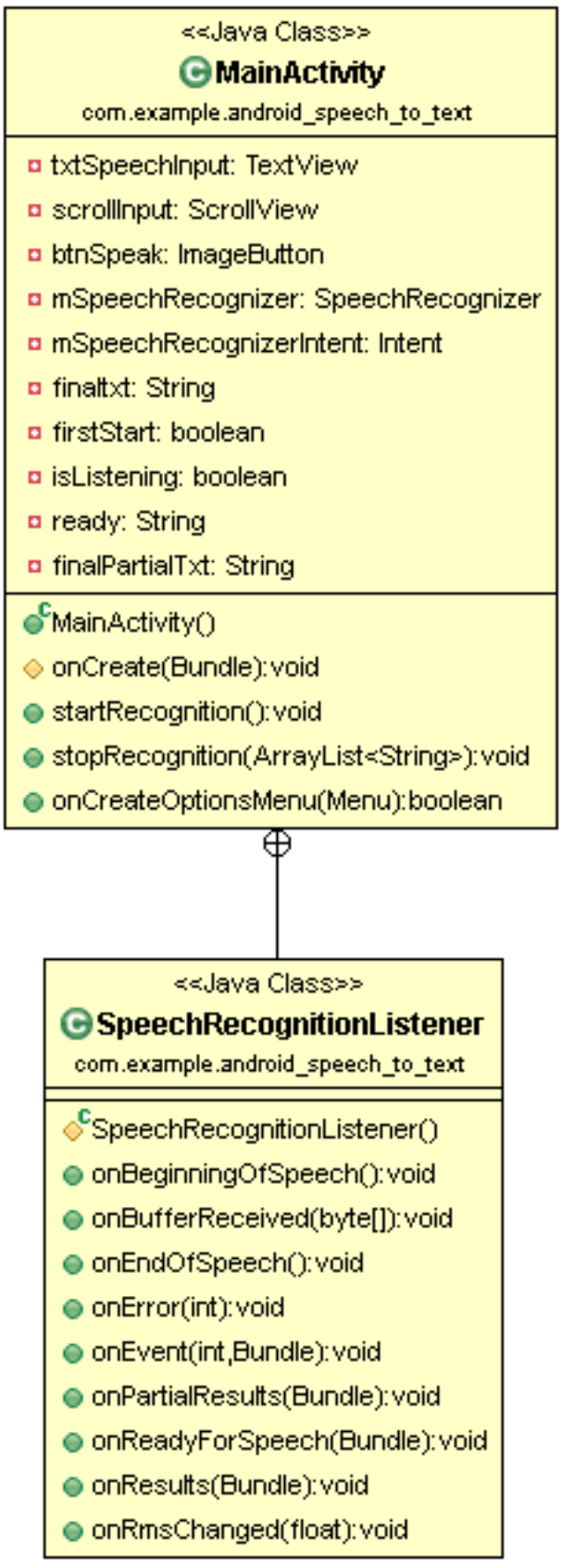


Abbildung 6: Klassendiagramm Spracherkennung

Nachdem in der MainActivity alle Variablen deklariert wurden, wird der SpeechRecognitionListener erzeugt.

OnCreateOptionsMenu(…) ist eine vom System generierte Methode und erzeugt die Action Bar der Anwendung, in dem neue Menüs hinzugefügt werden.

Der SpeechRecognitionListener besteht aus 9 vordefinierten Methoden. onBeginningOfSpeech() wird aufgerufen, wenn die Sprachanwendung gestartet wurde. OnEndOfSpeech() dagegen bei Ende der Anwendung. Im Fehlerfall wird onError(int) aufgerufen, mit einem Fehlercode als Parameter. onReadyForSpeech(…) wird aufgerufen, wenn die Anwendung nach Start auf eine Spracheingabe wartet. onPartialsResults(…) wird aufgerufen, nachdem ein Teilergebnis der Spracherkennung verfügbar ist. Dieses Teilergebnis wird im weiteren Verlauf vervollständigt und durch onResults(…) ersetzt.

4.4.4 Die Spracherkennungsanwendung – Implementierung

In diesem Abschnitt soll kurz auf die wichtigsten Methoden und Deklarationen der Sprachanwendung eingegangen werden. Nach einigen Deklarationen und Definitionen wird in der MainActivity die onCreate(…) Methode aufgerufen. Diese wird bei jedem Start der Anwendung ausgeführt und erstellt zunächst das Layout und weisst die entsprechenden Handler zu.

Anschließend werden SpeechRecognizer und SpeechReocgnizerIntent erzeugt, jedoch noch nicht gestartet. Mit der Methode putExtra(…) werden dem Intent einige Konfigurationen übergeben. So wird hier unter anderem mit EXTRA_PARTIAL_RESULTS bestimmt, dass Teilergebnisse durchaus erwünscht werden, um nicht zu lange auf das Ergebnis der kompletten Spracherkennung warten zu müssen.

```
public class MainActivity extends Activity {

  private TextView txtSpeechInput;

  ...
  private String finalPartialTxt = "";


  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    txtSpeechInput = (TextView) findViewById(R.id.txtSpeechInputField);
    scrollInput = (ScrollView) findViewById(R.id.scrollView1);
    txtSpeechInput.setMovementMethod(new ScrollingMovementMethod());

    scrollInput = (ScrollView) findViewById(R.id.scrollView1);
    btnSpeak = (ImageButton) findViewById(R.id.btnSpeak);
    mSpeechRecognizer = SpeechRecognizer.createSpeechRecognizer(this);
mSpeechRecognizerIntent =
new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

mSpeechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                    RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
mSpeechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_PARTIAL_RESULTS,
true);
mSpeechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE,
                    this.getPackageName());

    btnSpeak.setOnClickListener(new View.OnClickListener() {
```

```
        ...
    });

}
```

Startet den SpeechRecognizer:

```
public void startRecognition(){
        SpeechRecognitionListener listener = new SpeechRecognitionListener();
        mSpeechRecognizer.setRecognitionListener(listener);
        mSpeechRecognizer.startListening(mSpeechRecognizerIntent);
}
```

Stoppt den SpeechRecognizer und setzt alle Parameter auf den Ausgangszustand:

```
public void stopRecognition(ArrayList<String> text){
  finaltxt = finaltxt.concat(text.get(0));
  finaltxt = finaltxt.concat(" ");
  txtSpeechInput.setText(finaltxt);
  scrollInput.fullScroll(View.FOCUS_DOWN);
   txtSpeechInput.append("\n");
   scrollInput.post(new Runnable()
   {
       public void run(){
       scrollInput.fullScroll(View.FOCUS_DOWN);
       }
   });

  mSpeechRecognizer.stopListening();
  mSpeechRecognizer.destroy();
  ready = "";
  startRecognition();
}
```

Im SpeechRecognitionListener wurden nur 4 der 9 Standardmethoden implementiert:

```
protected class SpeechRecognitionListener implements RecognitionListener
{

   @Override
   public void onError(int error)
   {
       startRecognition();
   }

   @Override
   public void onPartialResults(Bundle partialResults)
```

```
    {
      ArrayList<String> text =
partialResults.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
      finalPartialTxt = finaltxt.concat(text.get(0));
       finalPartialTxt = finalPartialTxt.concat(" ");
      txtSpeechInput.setText(finalPartialTxt);
    }

    @Override
    public void onReadyForSpeech(Bundle params)
    {
       if(firstStart){
       txtSpeechInput.setText(ready);
       firstStart = false;
       }
    }

    @Override
    public void onResults(Bundle results)
    {
      ArrayList<String> text =
results.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
      stopRecognition(text);
    }
  }
}
```

Im Falle eines Fehlers wird die onError() Methode aufgerufen. Da dies auch den RecognitionListener beendet, wird dieser einfach neu gestartet.

Sind Teilergebnisse vorhanden, so werden diese mit der onPartialResults(…) Methode übergeben. In dem Fall wird der neue Teilstring einfach an den bereits bestehenden String der Spracherkennenung angefügt.

onResults(…) beinhaltet den String der ganzen Spracherkennung und ersetzt den String der aus den Teilergebnissen besteht.

Die onReadyForSpeech(…) Methode wird aufgerufen, sobald das System auf eine Spracheingabe wartet. Dieser Zustand wird mit einer kurzen Textmeldung quitiert, damit der Nutzer weiß, dass das System bereit ist.

Um die Anwendung im Offline-Modus zu verwenden müssen einige kurze Konfigurationen im Android Betriebssystem vorgenommen werden. Zunächst muss unter **Settings > Language & input** ein Hacken bei **Google voice typing** gesetzt werden um die Spracherkennungsdienste von Google überhaupt zu aktivieren.

Rechts von dem Menüpunkt können weitere Einstellungen vorgenommen werden. Nach Auswahl von **Offline speech recognition** können weitere Sprachdateien heruntergeladen und installiert werden, damit diese offline verfügbar sind. Nach der Installation neuer Sprachpakete kann bestimmt werden, welche Sprache als Standard genutzt wird oder es wird der Anwendung überlassen, zu erkennen, welche Sprache der Anwender spricht. Dies führt aber zu einer qualitativ schlechteren Spracherkennung.

5 Bedienungsanleitung

Die Hardware des Spracherkennungssystems, besteht nun aus folgenden Komponenten:

- Banana Pi Pro Platine
- 7 Zoll LCD Touchscreen
- Grafikkarte für Display
- Touchadapter angeschlossen an Display und USB des Banana Pi
- Externe Soundkarte angeschlossen an USB des Banana Pi
- Headset
- MicroSD-Karte als Festspeicher
- 2 Netzteile zur Stromversorgung
- 2 Taster als Reset und On/Off

Alle Komponenten wurden in einem PVC-Gehäse verbaut.

Einschalten:

Nach Anschluss der Netzteile an das Stromnetz fährt das System normalerweise von alleine hoch. Ist dies nicht der Fall, muss mit dem On/Off Schalter, welcher seitlich im Gehäuse montiert ist, das Gerät eingeschaltet werden. Falls das System noch immer nicht hochfährt, muss einige Sekunden der Reset Schalter gedrückt werden.

Betriebszustand:

Ist das Gerät hochgefahren, so befindet es sich im normalen Betriebsmodus wie ein Android Tablet. Um die Spracherkennung zu starten, muss zunächst

“Applications” ausgewählt und danach die Anwendung “Android Speech-To-Text” gestartet werden.

Die Spracherkennung wird nun durch berühren des Mikrofons auf dem Display gestartet und gestoppt.

Herunterfahren:

Das System sollte niemals in laufendem Zustand vom Stromnetz entfernt werden. Sollte dies doch geschehen, so muss bei erneutem Start, mehrere Sekunden, die Reset-Taste gedrückt werden.

Das System kann heruntergefahren werden, in dem der On/Off Taster so lange gedrückt wird, bis auf dem Display ein Pop-Up Menü mit dem Text “Ausschalten” erscheint.

Anwendung:

Nach Start der Spracherkennung sollte die Spracherkennungsbox, zwischen Sprecher und Hörer platziert werden, so, dass beide einen Blick in einem Winkel von circa 45° auf das Display haben.

Der Sprecher muss das beigefügte Headset korrekt aufgesetzt haben. Ein Headset hat den großen Vorteil, dass Umgebungsgeräusche so weit wie möglich vom Mikrofon ferngehalten werden. Ein Tischmikrofon ist zwar komfortabler in der Handhabung, jedoch wird dadurch die Spracherkennung erheblich negativ beeinflusst.

Ein Test der Spracherkennungsbox ergab folgendes Ergebnis:

Spracheingabe:	Verarbeitungszeit:	Textausgabe:
“Hallo mein Name ist Ramon Leber“	circa 2 Sekunden	“hallo mein Name ist rambo leber“
“Guten Tag“	t < 1 Sekunde	“guten tag“
“Was gibt es heute zu Essen“	circa 2,5 Sekunden	“was gibt es heute zu essen“

“Die Würde des Menschen ist unantastbar“	circa 2,5 Sekunden	“die würde des menschen ist unantastbar“
“Drei mal drei ist neun“	t < 1 Sekunde	“drei mal drei 9“

6 Zusammenfassung

Durch den demografischen Wandel wird es in unserer Gesellschaft immer mehr Menschen geben, die unter altersbedingten Handicaps zu Leiden haben. Schwerhörige Menschen leiden teilweise unter dem schlechten Tragekomfort und der ungenügenden Klangqualität von Hörgeräten.

Auf der anderen Seite sind Einplatinencomputer mittlerweile sehr weit verbreitet und günstig zu erwerben. Systeme wie das Raspberry Pi und Banana Pi sind leistungsstarke Systeme die gerne als kleine Helfer im Alltag, oder zur Lösung bestimmter Aufgaben individuell eingesetzt werden. Die Realisierung eines Spracherkennungssystems mit einem Einplatinencomputer soll deren Einsatztauglichkeit anhand einer technischen Aufgabe beweisen.

Es gibt zwei Ansätze wie man eine Spracherkennung mit dem Raspberry Pi durchführen kann. Die eine ist der Ansatz mit dem Spracherkennungssystem PocketSphinx, welches lokal auf dem Raspberry Pi installiert wird und über die Kommandozeile ausgeführt wird. Die Ergebnisse sind jedoch nicht zufriedenstellend, da die Spracherkennung sehr lange dauert und nie mehr als zwei Worte erkannt werden.

Der zweite Ansatz funktioniert über die Google Speech API, welche über das Internet aufgerufen wird. Es muss immer eine komprimierte Audio-Datei an den Google-Server geschickt werden um nach Rund 30 Sekunden eine qualitativ mittelmässiges Ergebnis zu erhalten.

Um eine gute Spracherkennung zu realisieren ist die Verwendung des Betriebssystem Android unumgänglich. Mit Android haben Entwickler und Nutzer Zugang zu den offline Google Spracherkennungsdiensten. Eine eigene Android Anwendung für ein Spracherkennugssystem kann damit schnell

entwickelt werden. Die Spracherkennungsbox liefert im Vergleich zu den anderen Lösungsansätzen sehr gute Resultate.

7 Ausblick

Eines der wichtigsten Kriterien bei der Auswahl eines geeigneten Einplatinencomputers für ein bestimmtes Projekt ist definitiv, welche Betriebssysteme für den Computer verfügbar sind. Je nach Zielsetzung eines Projektes ist genau abzuwägen, welches Betriebssystem mit seinen verfügbaren Systemfunktionen für das Projekt am besten geeignet ist. Dies ist eine erste, entscheidende Weichenstellung die ein Einplatinencomputer völlig nutzlos oder sehr brauchbar machen.

Hat man die richtige Wahl getroffen, so kann ein Minicomputer sehr viele Möglichkeiten eröffnen. Im Blick auf das Spracherkennungssystem für Schwerhörige könnten beispielsweise noch verschiedene Geräte an die GPIO Pins des Banana Pi angeschlossen werden, um anzeigen zu lassen, ob es an Haustür oder Telefon geklingelt hat.

Das System ist aber keineswegs auf die Funktion der Spracherkennung limitiert. Weitere Anwendungen könnten noch unzälige Dienste bieten. Bewegungsmelder die an die GPIO Pins angeschlossen sind könnten beispielsweise die Bewegungen in einer Wohnung erfassen, um so zu erkennen ob ein älterer Mensch gestürzt ist und nicht mehr aufstehen kann.

Einplatinencomputer haben durchaus ihre Daseinsberechtigung, auch wenn die meisten Anwendungen die damit realisiert werden, momentan eher in die Kategorie “Spielerei” gehören.

Literatur- und Webverzeichnis

[1] Welches Hörgerät: „Im Ohr Hörgeräte“
http://www.welches-hoergeraet.de/im-ohr-horgerate-vorteile-und-nachteile-zu-klassischen-horgeraten-65.html
[Stand: 09.02.2015].

[2] Wikipedia: „Einplatinencomputer“
http://de.wikipedia.org/wiki/Einplatinencomputer
[Stand: 09.02.2015].

[3] Wikipedia: “Raspberry Pi”
http://de.wikipedia.org/wiki/Raspberry_Pi
[Stand: 09.02.2015].

[4]Wikipedia: „Banana Pi“
http://de.wikipedia.org/wiki/Banana_Pi
[Stand: 09.02.2015].

[5] Sebastian Pech: „Sprachsteuerung“
http://www.spech.de/blog/article/sprachsteuerung
[Stand: 09.02.2015].

[6] Sebastian Pech: „Spracherkennung mit Google Speech API“
http://www.spech.de/blog/article/spracherkennung-mit-google-speech-api
[Stand: 09.02.2015].

[7] Raspberry Pi Geek: „Grafische Ausgaben mit dem C-Berry“
http://www.raspberry-pi-geek.de/Magazin/2014/04/Grafische-Ausgaben-mit-dem-C-Berry-Display
[Stand: 09.02.2015].

[8] Android: “SpeechRecognizer”
http://developer.android.com/reference/android/speech/SpeechRecognizer.html
[Stand: 09.02.2015].

[9] Amazon:
http://ecx.images-amazon.com/images/I/41wuZjdQpwL._AA160_.jpg
[Stand: 09.02.2015].

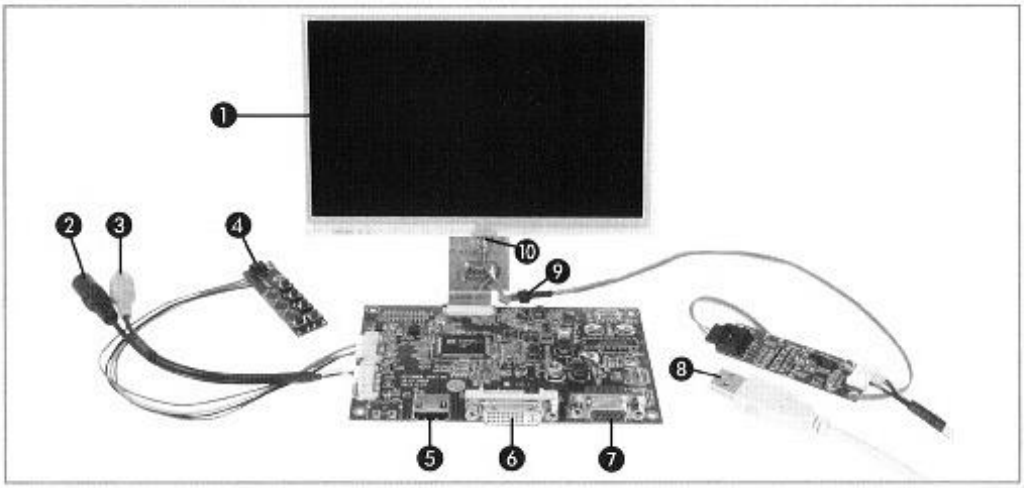
Anhang 1

Symbolerklärung

Das Symbol mit dem Ausrufezeichen im Dreieck weist auf wichtige Hinweise in dieser Bedienungsanleitung hin, die unbedingt zu beachten sind. Des Weiteren wenn Gefahr für Ihre Gesundheit besteht, z.B. durch elektrischen Schlag.

Das Gerät darf nur in trockenen und geschützten Räumen verwendet werden.

Bedienelemente



❶	Display	Anzeige für das eingespeiste Bildsignal.
❷	Hohlbuchse	Für die Spannungsversorgung mit einem 5,5/2,1 mm Hohlstecker (9...18 V-; Pluspol innen).
❸	FBAS-Eingang	Hier wird das FBAS-Bildsignal eingespeist.
❹	Bedienteil	Mit den Knöpfen auf dem Bedienteil können Sie das Set ein- und ausschalten, die Eingänge wechseln und diverse Bildeinstellungen vornehmen.
❺	HDMI-Eingang	Hier wird das HDMI-Bildsignal eingespeist.
❻	DVI-Eingang	Hier wird das DVI-Bildsignal eingespeist.
❼	VGA-Eingang	Hier wird das VGA-Bildsignal eingespeist.
❽	USB-Stecker	Zum Anschluss an den PC.
❾	Verbindungsbuchse	Hier wird das Flachbandkabel des Touchdisplays eingesteckt.
❿	Flachbandkabel	Zum Anschließen des USB-Touchcontrollers.

Inbetriebnahme / Bedienung

Achtung! Das Display-Set sollte vor dem Anlegen der Betriebsspannung berührungssicher in einem Gehäuse verbaut werden, um einen Kurzschluss und die daraus resultierende Brandgefahr zu vermeiden.

Anschluss

- Das Display-Set verfügt über 4 Video-Eingänge mit welchen nahezu jede Video-Quelle angeschlossen werden kann. Nehmen Sie hierfür das passende Kabel zur Hand (HDMI, DVI, FBAS oder VGA), stecken Sie das eine Ende in den Ausgang Ihrer Video-Quelle und das andere Ende in den HDMI- ❺, DVI- ❻, FBAS- ❸ oder VGA-Eingang ❼ des Sets. Es können alle 4 Eingänge gleichzeitig belegt werden.
- Verbinden Sie den Hohlstecker (Pluspol innen!) einer geeigneten Spannungsquelle (Netzteil, Autobatterie usw.; 9...18 V-, mind. 460 mA) mit der Hohlbuchse ❷. Legen Sie die Spannung jedoch erst an, wenn das Set berührungssicher in einem Gehäuse verbaut wurde.

USB-Touchcontroller anschließen

- Verbinden Sie das Flachbandkabel ❿ des Displays mit der 4-poligen Verbindungsbuchse ❾ des USB-Touchcontrollers so, dass die 4 Einkerbungen der Verbindungsbuchse ❾ und die Pins des Flachbandkabels ❿ nach unten zeigen.
- Schließen Sie anschließend den USB-Stecker ❽ des am Touchcontroller angeschlossenen USB-Kabels an einen freien USB-Port Ihres Computers an.