Techniques of Speech Data Collection

Version 1.0

Moritz Kaiser, Florian Schiel

Ludwig-Maximilians-Universität München

Technical Document No. 9
November 2005

Technique of Data Collection

November 2005

Moritz Kaiser, Florian Schiel

IPSK LMU München
Schellingstr. 3
80799 München

Tel.: (089) 2180-5751

E-Mail: schiel@bas.uni-muenchen.de;
ariser@bas-uni-muenchen.de

This technical document belongs to sub-project 1:
Multimodale Erkennung und Modellierung

The technical document belongs to a research project that was supported with funding from the Federal Ministry of Education and Research under the funding number 01 IMD 01. The responsibility for the content lies with the authors.

Inhaltsverzeichnis

1 Introduction

This document gives a brief description of the recording and server techniques used in the data collections within the Smartweb project. The data collection comprises requests to a computer information system (Smartweb) by telephone that are recorded with a mobile phone in the field ('handheld') or on a moving motorbike ('motorbike'). The handheld recordings are partly augmented by the video signal of the UMTS devices showing the face of the speaker ('video').

This document does not explain the arrangement and the procedures of the data collection and the design of the prompting. Please refer to SmartWeb TechDok No 6 for a detailed description.

The technique of the data collection is roughly based on the decisions of the Data-Workshop in Munich (22.07.04) - see the appropriate protocol -, on the pilot study (see SmartWeb TechDok No 1) and on correspondence with interested partners. In this document we do not document the decision process for the technical design but merely describe how the recordings were technically realized.

2 Basic Recording Structure for all Collections

2.1 ISDN Server and QXML Interpreter

Primary goal of the SmartWeb collection was to produce realistic speech sampled by a standard PDA-like device and transferred via a standard UMTS network. Therefore the first goal was to set up an ISDN server that allows us to

- captured the digitised speech stream as delivered by the UMTS network
- control the recording session by prompting the speaker

To this end we used a standard ISDN interface connected via a Fritz ISDN card to a PC. As a control and recording software we used a self-developed Pseudo-Voice-XML interpreter based on the Common ISDN API (CAPI). The dialogue is controlled by a XML script language QXML, which supports a small subset of the VoiceXML standard. QXML is far away from a full VoiceXML implementation, but it satisfies all the needs in this context. For more detailed description of the used CAPI classes and the implementation of the server software please refer to Appendix 6.1.

For the context of this report it is sufficient to list the capabilities of the server:

- Handles up to 2 calls in parallel; separate scripts are possible, thus enabling us to run different data collection types at the same time.
- On certain events during the processing of the XML document the server is able to fetch a new document from a standard HTTP Server. Along with the GET-request rich information can be submitted to the HTTP Server. Hence a big variety of sessions and Prompt sequences can be applied depending on caller ID and other measures.
- Prompting by re-playing off-line synthesised speech prompts in two different voices. We used the ATT German synthesis to produce a male voice

representing the SmartWeb system and a female voice representing the 'instructor' that prompts the speaker (see SmartWeb TechDoc No 6 for a detailed description of the 'situational prompting').

- Recording for a fixed maximum length after a prompt beep. A silence detection algorithm searches for a period of time where the level is below a threshold and terminates the recording prematurly. (see appendix 6.1 for details). Signals are stored to disk as being delivered by the UMTS network: 8kHz, ALAW.
- Recording of the UMTS signal for the total length of the connection.
- Receiving DTMF (Dual Tone Multi-Frequency) signals as input. The server can authenticate speakers by requesting a defined ID number for the recording session.
- Handling of DTMF (Dual Tone Multi-Frequency) signals. By pressing number buttons on the UMTS device the speaker may send signals to the server. The server can interpret these signals and start, pause or reset the QXML interpreter. We use this technique to control the session flow during 'motorbike' recordings (see appendix 6.1 for more details).

Since we do not want to record the same contents for each speaker, the server application is embedded into a structure of other programs that handle the selection of prompt blocks after a random scheme during run-time (for details about prompts and prompt blocks please refer to SmartWeb TechDoc No 6). Therefore each recording session may contain a different subset of prompt blocks. By this an equally distributed selection of contents is guaranteed.

2.2 Example QXML Script

The following shows an excerpt from a typical QXML script as used in the 'handheld' data collection:

```
<?xml version="1.0"?>

<!DOCTYPE qxml SYSTEM "file:///etc/quapi/qxml.dtd">

<qxml version="1.0">

<block><audio src="/raid/r33/Quapi/SW-PROMPTS/xxx-0000pro-010.al"/></block>

<form name="xxx-0000rec-010.al"><record dest="xxx-0000rec-010.al"
name="xxx-0000rec-010" beep="true" maxtime="12000"
finalsilence="2000"/></form>

<block><audio src="/raid/r33/Quapi/SW-PROMPTS/xxx-0000pro-020.al"/></block>

<form name="xxx-0000rec-020.al"><record dest="xxx-0000rec-020.al"
name="xxx-0000rec-020" beep="true" maxtime="12000"
finalsilence="2000"/></form>

<block><audio src="/raid/r33/Quapi/SW-PROMPTS/xxx-0000opr-010.al"/></block>

<block><audio src="/raid/r33/Quapi/SW-PROMPTS/xxx-0000pro-030.al"/></block>

<form name="xxx-0000rec-030.al"><record dest="xxx-0000rec-030.al"
name="xxx-0000rec-030" beep="true" maxtime="12000"
finalsilence="2000"/></form>

...
```

The first `<block>` structure replays a pre-synthesised prompt stored in `xxx-0000pro-010.al`.

The subsequent `<form>` structure records the answer of the speaker after issuing a beep and for a maximum recording time of 12secs. Silence detection requires a minimum silence interval of 2secs before terminating the recording. The signal is stored in the file `xxx-0000rec-010.al`.

The above script has been automatically compiled from a text-form script that describes the individual prompt blocks and a randomly selected number of block numbers during run-time. The script is stored together with the recorded signals for reference (see meta data files of type 'rsc' in the corpus edition).

Authentication is not handled by the QXML script but controlled by the server software before starting the QXML interpreter. Same is true for the interpretation of DTMF signals during the recording session.

2.3 UMTS Network

Two working UMTS networks (G3 standard) were investigated before the start of the collection for their respective suitability: German Telekom and O2.

Both networks turned out to be almost identical (probably caused by the fact that O2 used the cell technology of German Telekom):

Downstream bit rate for data connection was theoretically 384kbit/s, upstream 64kbit/s.

For speech transmission the bitrate lies between 6,6 kbit/s and 23,8 kbit/s when the AMR-WB codec described in “3GPP 26.171 v5.0.0” is in use. If the G3 connection fails, an automatic hand-over is possible to GSM. This feature was disabled for the collection since the connection tends to stay on GSM after a hand-over and we did not want 'mixed' network protocols within a recording session.

The AMR-WB (adaptive multi rate – wide band) codec supports nine different source rates which can be switched every 20 ms.

Video streaming to a server was not possible.

UMTS coverage was (May 2005) concentrated on larger cities but not on major traffic routes. The reason for this is probably that the G3 standard as used at that time was not specified for moving clients faster than 60km/h. In preliminary tests it turned out that it was in fact difficult to find a route through the city of Munich that was constantly covered by G3. This was not a major problem for the 'handheld' collection where we had a defined set of recording locations, but for the 'motorbike' collection where we had difficulties to a find a proper route of suitable length within Munich.

2.4 Recording Unit

The speaker carried a group of interconnected devices that is referred to as the 'recording unit' in the following. The basic recording unit consists of

- a UMTS capable cellular phone Siemens U15 (Nokia 6680 for 'video' recordings)
- a harddisk recorder iHP 140 by iRiver capable of recording 2 channels in un-compressed quality (16bit, 44,1kHz, PCM)
- Primary microphone: a Bluetooth 1.1 headset with directional microphone (??? manufactures and types) or a Bluetooth 1.1 motorbike helmet with 2-capsule microphone array (BMW prototype: (BMW Systemhelm 5 with communication system WCS-1) or a cable connected headset microphone (1 manufacturer and 1 types)
- Secondary microphone: either a standard collar-attached electret-microphone as used for simple speech recordings (iRiver) or a laryngal neck microphone (KEP 33S)
- a Bluetooth 1.1 receiver (B-Speech) connected to the UMTS cellular and to the harddisc recorder (not used when a cable connected primary microphone was used)
- analogue transceiver circuit (needed because the signal levels of microphones and recording units differ considerably; see Appendix B for details of the circuit).

The signal flow is as following:

The primary microphone transmits the close mouth signal via Bluetooth 1.1 to the external Bluetooth receiver (or is directly connected via cable). The signal is split by the analogue circuit and fed into the UMTS cellular phone and into the left channel of the harddisk recorder. The UMTS cellular transmits the signal via G3 protocol and ISDN to the server where the signal is captured and stored 'as is'.

The secondary microphone is connected via a analogue circuit to the right channel of the harddisk recorder.

The external Bluetooth receiver was necessary because the UMTS cellular phone was not capable to deliver the microphone signal to an external recording device.

2.5 Post–processing

Since the individual server recordings are not synchronised with the parallel harddisc recordings in any way, an alignment of the signals after the recording was necessary. We developed a fully automatic alignment technique which described in detail in appendix 6.2.
The result of this post-processing step is a so called marker file (*.mar) which contains the start and end of each individual server recording within the harddisk recording.

3 Specific Setup for Collection 'handheld'

The following details regard only to recording sessions with initial u, the basic 'handheld' recordings.

On the left channel of the hard disk recorder the signal from the headset was recorded and on the right channel the signal from the collar attached microphone was recorded. Diagram 1 shows the relevant influences on the signal during a standard handheld session. Diagram 2 shows the reduced setup when a cable connected headset was used.

Two setups were built to accommodate the slightly different needs for cable and bluetooth connected headsets. For the amplifier circuit for bluetooth setup see Diagram 5. For cable connected setups see Diagram 6.

The server was configured to ask the person under test for an identification number. This test helped to prevent ambiguities between recorded data and the forms filled out by the supervisors.

4 Specific Setup for Collection 'motorbike'

The following details regard only to recording sessions with initial m, the 'motorbike' recordings.

4.1 Recording unit

Speech recording and transmission was achieved with a helmet manufactured by BMW. This helmet used to deliver other signal levels to the bluetooth receiver than the headsets used for the handheld recordings. So an attenuator was attached to the signal flow between the bluetooth receiver and the cellular phone, which reduced the signal level by 6 dB, which can be seen at Diagram 3.

Furthermore, a special device was introduced to allow the driver of the motorcycle to control the recording over some pushbuttons. This special device contained a relay which could break the connection between the cellular phone and the Bluetooth interface in the incoming direction. The relay was triggered by one pushbutton and initiated the phone call for the session. Three other pushbuttons were connected to a DTMF-generator to produce predefined DTMF-codes for the control of the server over an established phone connection.

On the left channel of the hard disk recorder the signal from the helmet was recorded and on the right channel the signal oft the throat microphone was recorded.

4.2 Server configuration

It was impossible for the driver of the motorbike to dial numbers on the cell phone during driving. Hence identification was performed by the supervisor before the session. The recording started without a prompt for recording but with a prompt for a single button press for proceeding.

5 Specific Setup for Collection 'video'

The 'video' collection is basically a variant of the 'handheld' collection. Session have the initial 'i'. In the following we describe the differences to the 'handheld' recording technique mentioned above.
The main difference between the Handheld setup and the video setup is the fact that there was technically no possibility to record the signal transmitted over the bluetooth connection directly to the mobile hard disk as can be seen in Diagram 4. Hence only the sound received over the simple electret microphone was recorded. Thus no amplifiers and attenuators were necessary to match the levels of different signals. The video track was recorded by the embedded camera software (simply called 'Camera') of the cellular phone. To prevent the cell phone from emitting warning sounds during the recording which otherwise would have messed up all audio tracks, the video recording was set mute.

6 Appendix

6.1 QUAPI Server (by Ning Chen)

Structure

Overview

The QUAPI server is a phone dialogue server based on Common ISDN API (CAPI).
The dialogue is controlled by an XML script language QXML, which supports a small subset of
the VoiceXML standard. It is far away from a full VoiceXML implementation.
Although this software needs a JAVA Virtual Machine you will not see
the typical JVM processes, because the JVM is started from C wrapper program.
The server is configurable with the properties file /etc/quapi/quapi.cfg.

Main Classes

There are some classes,which are essential for understanding the internal structure of the QUAPI server.

quapi.Quapi

This is the main thread of QUAPI server.
It will load the C libraries and initialise the CAPI and Connect objects.
It is initialised only once by every new start of QUAPI server.

quapi.capi.Capi

This class is an abstraction of ISDN-Capi interface,which lies in the lowest level.

quapi.capi.Connect

There is only one Connect object in the QUAPI server.
This object has two main tasks. One is to allocate a new connection when a new call comes. The other is to transfer the events from Capi interface to the corresponding connection.
There is one internal data structure included in Connect,which manages all objects of class SpaceInvader. Every SpaceInvader object maps to one connection or session. Its task is to try to break the processing of the QXML-

document when a DTMF-event occurs during the processing of main dialogue QXML-document.

quapi.capi.Connection

Every connection stands for a session. How many connections can be allocated at
the same time depends on the number of D-Channels of the ISDN-Card (generally two).
Every connection is driven by QXML documents, which the QUAPI server retrieves via a cgi-perl script from a separate apache server (except the initial QXML document, which must already be configured before the start of every session). During one session there may exist several handshakes between the two servers.
The main task of every connection is to process all loaded QXML documents. Events which are issued by the CAPI interface, can affect the processing of the QXML documents. Exactly speaking, special QXML elements need to receive special events from the CAPI interface in order to know whether the processing of this element has been finished and it should proceed to the next element. Every connection will last until the processing of all QXML elements has finished, except for a submit element, which triggers the apache server to deliver a new QXML document to be processed subsequently.

quapi.vXML.Parser

This class is the interpreter of a QXML document.
The parser adopts the DOM model to construct a tree structure from every QXML document.

Silence Detection

Description

The silence detector starts after the beep signal as one single thread for every new prompt.
It will detect the end of the speech which the test person has uttered automatically.
When silence has been detected, this thread will call back to the main thread of the program which then continues.

Algorithm

The silence detector extracts three parameters from the audio signal every second: zero crossing rate, energy and standard deviation of energy. The analysis window is 1sec and the
values are based on 100 sampling points. For each of these parameters an upper and lower threshold defines a 'speech range' within which the parameter should be found during regular speech input. If one parameter is found to be outside this 'speech range' for a longer consecutive period than t_dur (usually 3sec), silence is detected. Silence cannot be detected before time window of t_min secs and the recording will stopped latest after t_max secs.

Class

quapi.util.SilenceDetector

Space Invader

Description

The so-called Space Invader of the Quapi server allows the test person to pause and continue the recording session at any time by pressing buttons on the cellular phone. The button signals are simply transferred to the server in the form of DTMF (Dual Tone Multi-Frequency) signals.

Algorithm

The Space Invader will test the status of the QXML interpreter of the Quapi server. When receiving the following DTMF events the Space Invader will do the following:

- DTMF 2 : pause processing
- DTMF 8 : continue processing
- DTMF 0 : reset

Class

quapi.util.SpaceInvader

6.2 Automatic Segmentation (by Ning Chen)

Since the harddisc recordings are not in any way synchronised with the recordings done by the QUAPI server, we developed an automatic segmentation scheme to detect the corresponding time frames of the individual server recordings within the continuous harddisc recording. The result of this segmentation is stored in the so called MAR files (*.mar). The three-column table in the MAR file lists the prompt ID which correspond to the file name of the server recording together with the first and last sample of the segment within the harddisc recording.
Using these values it is possible to cut out corresponding segments from the harddisc recording automatically (for instance for ASR training).

Algorithm

The automatic segmentation scheme uses an iterative cross correlation method as follows:

Initial Information

The IDs and the order of the server recordings are known beforehand (for example this information can be extracted from the QXML file of the recording). We can therefore safely assume that the first recording segment lies within the first 300secs of the harddisc recording.

Windowing

The first server recording is searched for in a window of 300sec from the start of the harddisc recording. The following segments are searched for in a window of 50sec length starting with the end of the previous segment.

Search Algorithm

1. Initialisation:
   Ti = Tinitial (either 0 (first segment) or the end of previous segment)
   STEP1 = 441 (= 10msec)
   STEP2 = 41 (= 1msec)
   F = 9
2. The auto-correlation of the server recording is calculated (AKF). This is basically the sum over all squared sample values within the server recording.
3. Then the server recording is cross-correlated to the harddisc recording starting at Ti sec (KKF). The KKF is normalised by the AKF to avoid loudness dependencies in the following threshold decisions. This results in the decision factor DF.
4. Termination: If DF > 0,20, the search is finished.
   Result: Ti is the start of the found segment.
   If DF is smaller than 0,20 , the search is continued by shifting the server

recording by SHIFT samples to the future:
IF DF < 0,0025 THEN SHIFT = STEP1
ELSE IF DF < 0,0100 THEN SHIFT = STEP2
ELSE IF DF < 1/F THEN SHIFT = 1 (= 22microsecs)
The idea here is that a raising DF indicates an approaching maximum in the KKF. Therefore we downstep the time shifts in the search.
Continue with 3.) until either the terminal condition or the end of the searched window is reached.

5. If the end of the search window is reached without any terminal condition, the search parameters are adjusted as follows:
   STEP1 = STEP1 / 2
   STEP2 = STEP2 / 3
   F = F – 1
   Ti = Tinitial
6. If F is equal 5 (after 4 unsuccessful searches) the algorithm is aborted with an error messages. In this case most likely there is a problem with the recorded signals.

Results

The segmentation scheme as described above turned out to be very robust. Even in the heavily distorted UTMS Motorbike Recordings the segmentation did not fail. The only failures of the algorithm turned out to be empty recordings (that is the user did not utter anything).

6.3 Block diagrams

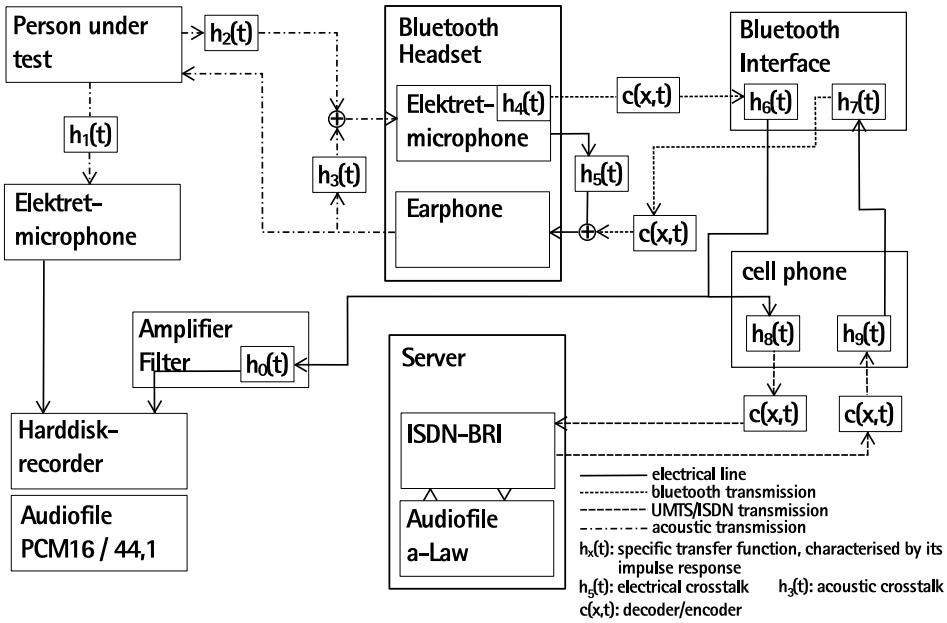

Diagram 1
Block diagram of handheld setup with bluetooth headset

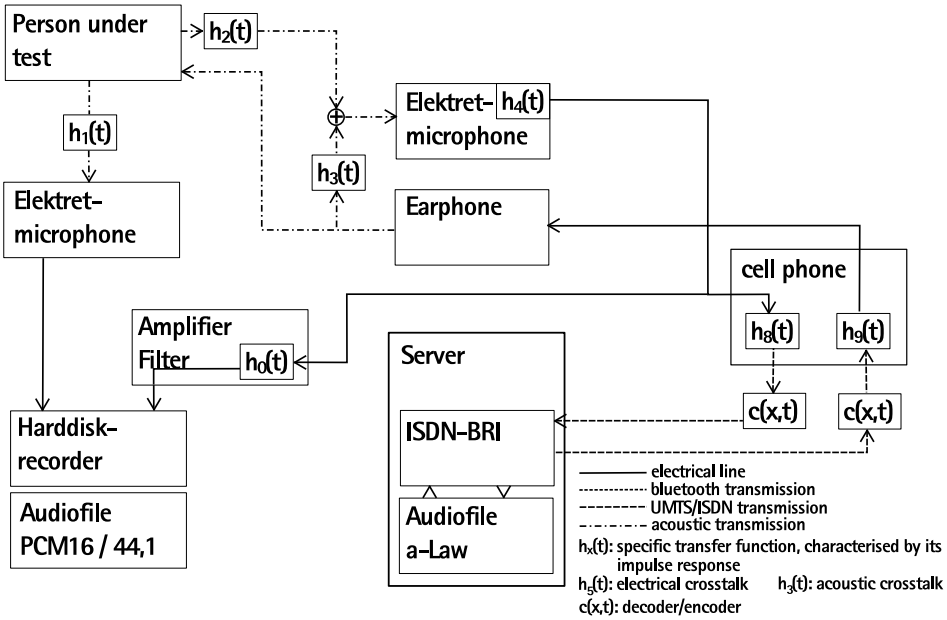

Diagram 2
Block diagram of handheld setup with cable connected headset

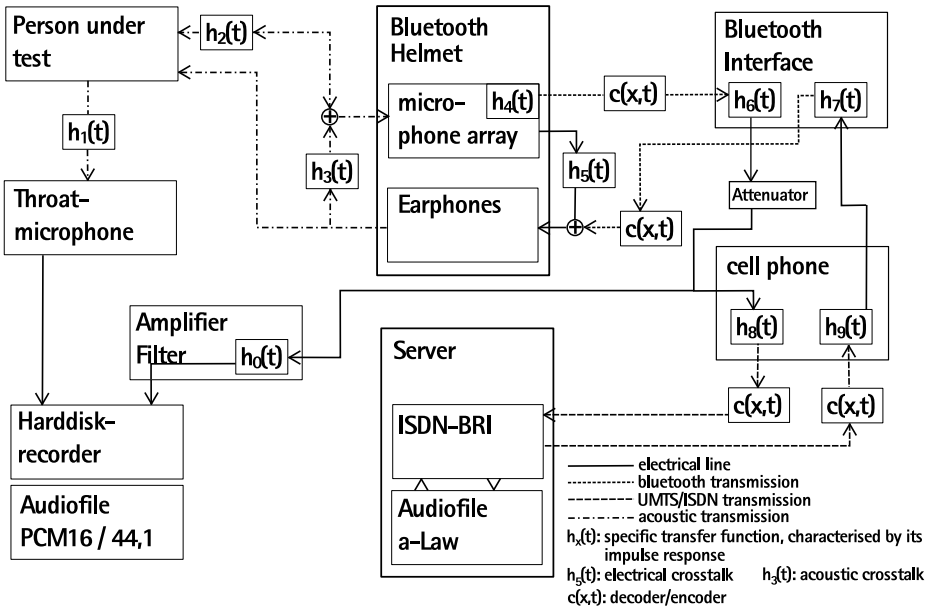

Diagram 3
Block diagram for motorbike setup

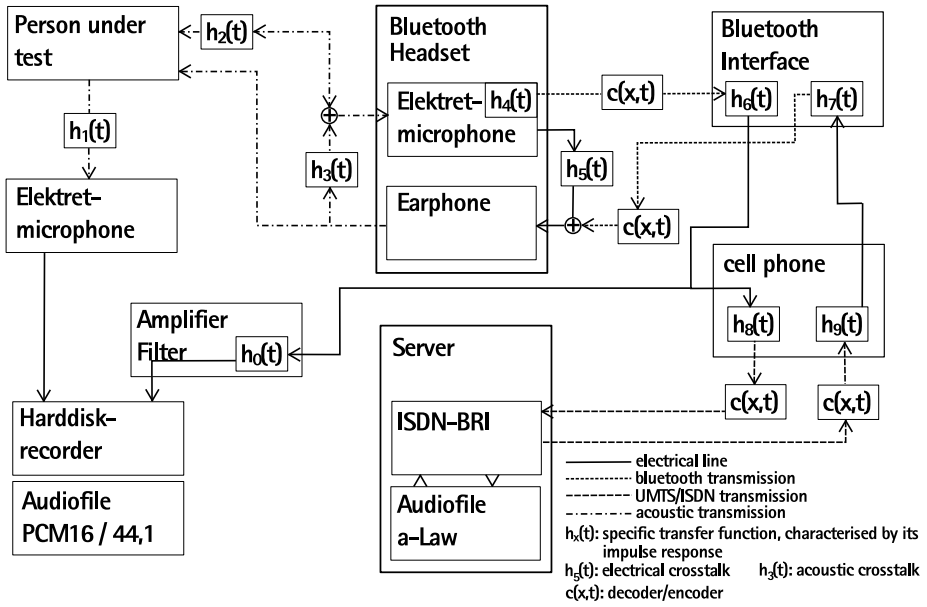

Diagram 4
Block diagram for video setup

6.4 Schematics

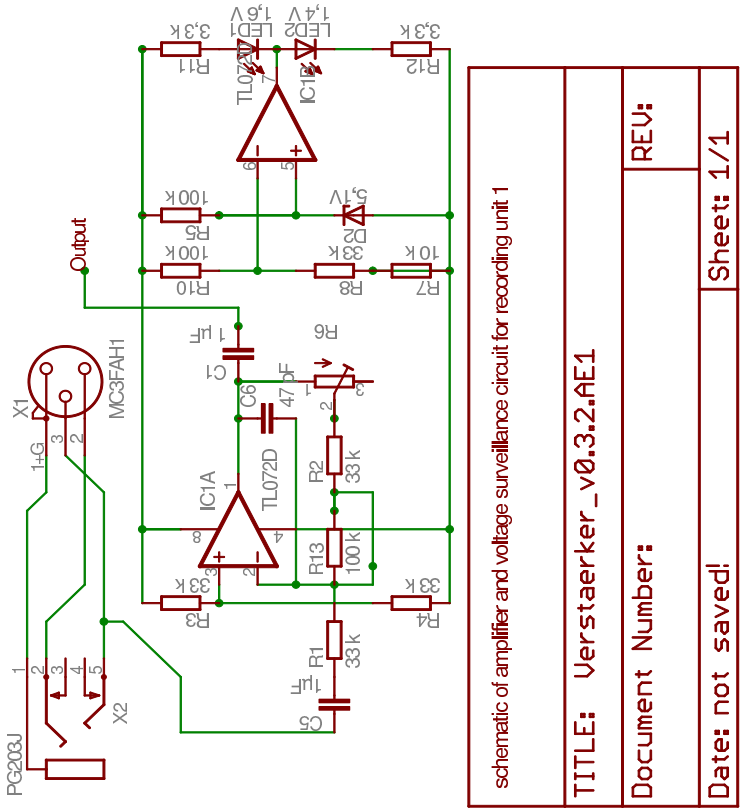

Diagram 5

Technique of Data Collection

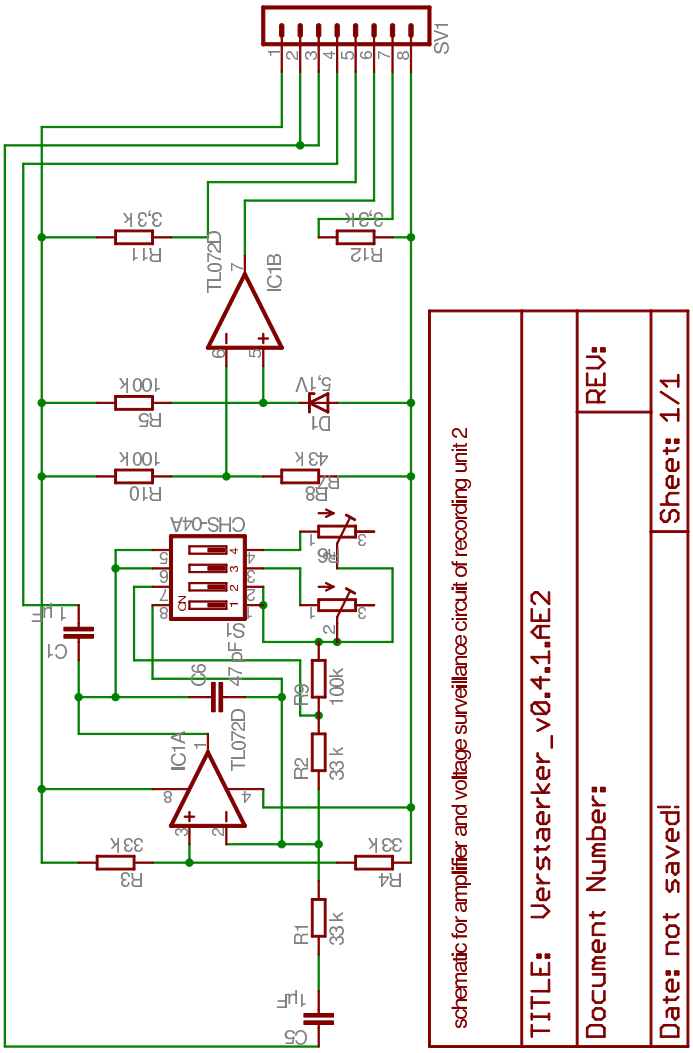

Diagram 6