Freie Universität Berlin
Fachbereich für Mathematik und Informatik
Institut für Informatik

Diplomarbeit

Autonomous Wildlife Monitoring

Design, implementation and evaluation of an embedded platform for nightingale song recording

Bearbeitet von:
Marco Ziegert

Betreut von:
Prof. Dr.-Ing. Jochen Schiller
Dipl.-Inform. Tomasz Naumowicz

04. Januar 2010

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

_____

Berlin, 04.01.2010

Danksagung

Ich möchte Dipl.-Inform. Tomasz Naumowicz für die gute Betreuung während der Umsetzung der Diplomarbeit danken. Eine Abschlussarbeit basierend auf einer reinen Softwareentwicklung hätte möglicherweise einen barrierefreien Weg dargestellt, jedoch hat er mich schon während des Entwurfs mit Begeisterung in meiner Überzeugung unterstützt, diese mit der Implementierung einer neuen eingebetteten Plattform zu verbinden.

Dabei wäre es ohne den Rückhalt meiner Familie nicht möglich gewesen ein Informatikstudium und die hier entstandenen Ideen umzusetzen. Meine Schwester, welche mir mit Ratschlägen aus ihren eigenen Erfahrungen immer zur Seite stand und meinem Bruder, der in mir vor vielen Jahren die erste Leidenschaft für die Informatik geweckt hat.

Die größte Anerkennung und Dank gebührt dabei meinen Eltern, die immer für mich da waren, um mich bei der Umsetzung dieses Traumes von ganzem Herzen zu unterstützen.

Inhaltsverzeichnis

1. Einleitung....1
   1.1. Motivation....1
   1.2. Aufgabenstellung und Lösungsansatz....2
   1.3. Verwandte Arbeiten....3
   1.4. Lernstrategien und adaptiver Wert des Gesangs bei Nachtigallen....5
   1.5. Vorhandene Infrastruktur....6
      1.5.1. Basisstation....9
      1.5.2. Voicelink-Client....11
      1.5.3. Problemstellung....12
   1.6. Grundlagen....13
      1.6.1. Abtasttheorem....13
      1.6.2. Signal-Rausch-Verhältnis....14
      1.6.3. Psychoakustisches Modell....15

2. Systemarchitektur....16
   2.1. Anforderungen....16
   2.2. Evaluierung von Hardwareplattformen....18
      2.2.1. MSB-430H....18
      2.2.2. ARM....19
         2.2.2.1. STM32 PRIMER2....20
         2.2.2.2. MSB-A2....22
         2.2.2.3. Olimex LPC-P2378....23
      2.2.3. Fazit....24
   2.3. Evaluierung von Komponenten zur drahtlosen Datenübertragung....25
      2.3.1. 443 Mhz und 868 MHz ISM-Band....25
         2.3.1.1. Texas Instruments CC1020 / CC1100....25
      2.3.2. 2,45 Ghz ISM-Band....26
         2.3.2.1. Nordic Semiconductor nRF24L01+ / nRF24LU1....26
         2.3.2.2. ZigBee / 802.15.4....26
         2.3.2.3. Bluetooth 802.15.1....27
         2.3.2.4. WLAN 802.11b/g....27
      2.3.3. Fazit....30
   2.4. Analog-Digital-Wandler....31
      2.4.1. Analog Devices AD1871....33
   2.5. Stromversorgung....34
   2.6. Gehäuse....36
   2.7. Erweiterungsplatine....36
      2.7.1. AD-Wandler....36
      2.7.2. Vorverstärker....38
      2.7.3. Mikrofonkompressor....39
      2.7.4. Funkmodul....41
      2.7.5. Schaltplan und Platinenlayout....42

3. Entwurf und Implementierung....45
3.1. Embedded Platform....45
3.1.1. Entwicklungsumgebung....45
3.1.2. Entwurf des Programmablaufs....47
3.1.3. Aufbau und Architektur der Anwendung....49
3.1.4. LPC2378....50
3.1.4.1. Energieeinsparung....51
3.1.5. Dateisystem....53
3.1.6. AD-Wandler....54
3.1.6.1. Ansteuerung....54
3.1.6.2. Konfiguration....54
3.1.6.3. Energieeinsparung....54
3.1.7. Funkmodul....55
3.1.7.1. Schichtenmodell....55
3.1.7.2. Konfiguration....56
3.1.7.3. Ansteuerung....57
3.1.7.4. Kommunikation: Paket/Text....60
3.1.7.5. Benutzer-API....62
3.1.7.6. Energieeinsparung....64
3.1.8. Implementierung des Programmablaufs....67
3.1.8.1. Scheduler....67
3.1.8.2. Zeitplansynchronisation....68
3.1.8.3. Aufnahme....69
3.1.8.4. Aufnahmeversendung....69
3.1.8.5. Verlustfreie Kompression....70
3.2. Erweiterungsservice der Basisstation....72
3.2.1. Entwicklungsumgebung....72
3.2.2. TCP-Server....72

4. Auswertung....74
4.1. Testaufbau....74
4.2. Testlauf....76
4.3. Schaltungsanalyse....78
4.4. Tonqualität....81
4.5. Energieverbrauch....84

5. Zusammenfassung und Ausblick....87
5.1. Ausblick....88

Begriffserklärung....92

Anhang....93

Anwenderhandbuch....93

Entwicklerplatine – Erstkonfiguration.....93
WLAN-Modul – Erstkonfiguration.....93
WLAN-Modul – Rettungsmodus.....94
Basisstation Konfiguration.....96

Hinweise für Entwickler.....96

1. Einleitung

1.1. Motivation

Eingebettete Systeme ermöglichen uns im Alltag und Berufsleben gestellte Anforderungen einfacher und präziser zu erfüllen als je zuvor. Sie sind meist klein, leicht, sehr energieeffizient und vor allem spezialisiert auf ihre jeweilige Aufgabe. Ihre Geschichte beginnt mit dem ersten bekannten eingebetteten System, dem Apollo Guidance Computer. Entwickelt von Charles Stark Draper am MIT Instrumentation Laboratory, stellte er der Besatzung sowohl im Kommandomodul als auch in der Mondlandefähre ein Wege-Such- und Diagnosesystem zur Verfügung. ([HIST0] S.5ff) Dank Massenproduktion von Mikrocontrollern sind heutzutage eingebettete Systeme weit über die Raumfahrt hinaus im Einsatz.

Doch auch heute stoßen unterstützende Systeme noch in wissenschaftlichen Projekten an ihre Grenzen, wenn es auf Formen von menschlicher Beobachtungsgabe oder Interpretation ankommt. Des weiteren begründet sich der eigentliche Erfolg einer Idee oder eines Produkts durch seine Tauglichkeit im täglichen Einsatz und diese ist in den meisten Fällen sehr eng mit der Benutzerinteraktion und der dadurch entstehenden Komplexität verbunden. Um so individueller ein Gerät agieren soll, um so größer ist die Wahrscheinlichkeit, dass es meist umständlich eingerichtet und konfiguriert werden muss. Nicht nur, dass dem Anwender gewohnte PC-Eingabegeräte wie Tastatur und Maus fehlen, so muss meist auch auf detailliertes visuelles Feedback mittels einer Bildschirmausgabe verzichtet werden. Die Nutzergruppe muss deshalb meist aufwendig geschult werden, um das System mit diesen speziellen Vorkenntnissen überhaupt bedienen zu können.

Eine zweite Kategorie bilden daher eingebettete Systeme die auf jede direkte Benutzerinteraktion verzichten. Sie reagieren autonom auf vordefinierte Muster ihrer Messwerte. Wir finden sie zum Beispiel in steigender Zahl in Kraftfahrzeugen und ermöglichen damit überhaupt erst die Funktion von Antiblockiersystem, elektronischen Stabilitätsprogrammen, Traktionskontrolle oder dem Auslösen von Airbags bei schweren Unfällen.

Ein solch nahtloses einfügen in unsere Umwelt ist verständlicherweise die wünschenswerteste Variante, da sie die Fehlerquelle Mensch umgeht. Die Grundbedingung für die erfolgreiche Etablierung eines eingebetteten Systems ist somit, dass es vom Anwender meist nur sehr wenig oder keine Vorkenntnisse verlangt und durch seinen Einsatz einen echten Nutzwert darstellt, in dem es eine Problemstellung dauerhaft genauso gut oder besser lösen kön-

nen als der Mensch. In den letzten Jahren wird nun angestrebt diese immer noch bestehende Lücke in einigen wissenschaftlichen Bereichen durch Neuentwicklungen, basierend auf eingebetteten Systemen, zu schließen.

Die AG Verhaltensbiologie der Freien Universität Berlin sammelte in mehreren Feldversuchen Gesangsaufnahmen der Nachtigall. Das Ziel ist aus den erlangten Strophen (siehe Abbildung 1.1a) des Singvogels, von denen er „mehrere Tausend produzieren kann und die eine verblüffende Vielfalt von mehr als 200 verschiedenen Strophentypen umfassen können“ [VB09], Rückschlüsse auf die dazugehörigen Lernstrategien zu ziehen. Diese Aufnahmen wurden bisher bei langwierigen Vor-Ort Terminen, teils bei Nacht oder in den frühen Morgenstunden von Studenten und Mitarbeitern der Forschungsgruppe, mit Hilfe von Richtmikrofonen und handelsüblichen digitalen Aufnahmegeräten erstellt. Danach werden sie manuell am PC nachbearbeitet, Strophen kategorisiert und bestimmten Standorten und Tieren zugeordnet werden.

1.2. Aufgabenstellung und Lösungsansatz

In Abstimmung mit der AG Verhaltensbiologie soll ein Konzept erarbeitet werden, um einen weiteren Ausbau des Projekts „Lernstrategien und adaptiver Wert des Gesangs bei Nachtigallen“ zu ermöglichen. Die Eignung des Konzepts soll durch Aufbau eines Prototypen und eine Beispielimplementierung bewertet werden. Hierzu soll es einem Forscher durch das neu entwickelte autonome System ermöglicht werden langfristige Beobachtungen, in Form von mehrstündigen Tonaufnahmen, ohne seine Anwesenheit durchführen zu können.

Aufnahmestandorte aus den verschiedensten Gründen wartungsfrei über den gesamten Zeitraum einer Saison zu betreiben, stellt besondere Herausforderungen an das zu entwickelnde Hardware- und Softwarekonzept. Stromsparende, batteriebetriebene Systeme mit Laufzeiten von mehreren Monaten oder sogar Jahren sind uns zum Beispiel in Form von Sensorknoten wie sie die Scatterweb Plattformen MSB-430 [BKLS07] und MSB-A2 [BWB08] darstellen bekannt. Beispielhafte Anwendungsfälle sind die Langzeitüberwachung, auch in teils schwer zugänglichen Umgebungen, in denen z.B. mehrere Messungen pro Sekunde durchgeführt, ausgewertet und per Funk weitergeleitet werden sollen. Die Anforderungen an deren Rechenleistung, Speicherkapazität und Datendurchsatz sind zunächst gering, programmiertechnische Optimierungen in diesen Bereichen bergen allerdings erhebliche Energiespareffekte.

Qualitativ hochwertige und verlustfreie Tonaufnahmen erzeugen allerdings ein außerordentlich hohes Datenaufkommen, für das Sensorknoten nicht konzipiert wurden. Hier soll geprüft werden in wieweit bestehende Hardware trotzdem beibehalten, optimiert oder durch neue Komponenten ersetzt oder ergänzt werden muss. Beim Entwurf der Software muss darauf eingegangen werden, dass Funkverbindungen immer wieder gestört sein können, diese muss darauf angemessen reagieren können und darf während des gesamten Betriebs nicht in einen undefinierten Zustand geraten. Wichtig ist vor allem, dass bereits fertiggestellte Aufnahmen in keinem Fall verloren gehen dürfen. Eine besondere Herausforderung ist die nahtlose Integration in eine bestehende Infrastruktur die sinnvoll mit einbezogen, aber so wenig wie möglich verändert werden soll. Einem Nutzer an einem Voicelink-Client soll dabei in der täglichen Anwendung und Programmierung verborgen bleiben, dass es sich hierbei um eine spezielle Mikrofonstation handelt, die sich in der Realisierung von den anderen Mikrofonen im Voicelink-Netzwerk unterscheidet.

Diese Diplomarbeit beschränkt sich daher auf den Entwurf, Implementierung und Evaluierung eines eingebetteten Systems zur Gesangsaufnahme von Nachtigallen, welches das parallel entworfene System von Basisstationen von Frank Beier erweitert.

1.3. Verwandte Arbeiten

Autonomous Monitoring of Vulnerable Habitats

Die University of Oxford hat, zusammen mit der Freien Universität Berlin und Microsoft Research, das Projekt „Autonomous Monitoring of Vulnerable Habitats“ ins Leben gerufen. Das Ziel ist mit Hilfe von Wireless Sensor Networks (WSN) Abläufe und äußere Einflüsse, die die Population der Seevogelart Manx Shearwater auf Skomer Island positiv oder negativ beeinflussen, leichter zu identifizieren und zu verstehen. Diese Erkenntnisse könnten dazu dienen veränderte Verhaltensweisen der Vögel zu erklären und Schutzprojekte besser zu koordinieren. [AMVH]

Das Augenmerk liegt darauf, u.a. mittels angebrachter RFID-Sender die Verhaltensweisen der Seevögel in der Nähe des Nestes zu beobachten. Eine Aufzeichnung von Tondaten, wie es in dieser Arbeit geplant ist, erfolgt nicht. Allerdings ist auch hier die Nutzung einer Infrastruktur vorgesehen.

MSB-430H Implementierung eines Audiolinks

In meiner Studienarbeit, welche zusammen mit Frank Beier erstellt wurde [MF08], konnte die neuere Highspeed-Version des MSB-430 auf ihre Audiotauglichkeit überprüft werden. Hierzu wurden die 12-Bit Werte des internen AD-Wandlers mittels ADPCM verlustbehaftet auf 4-Bit komprimiert und als Live-Stream an eine zweite MSB-430H Station übertragen werden.

Hier stellte sich bereits frühzeitig heraus, dass die Verluste der Datenpakete, welche über den vorhandenen CC1100 gesendet wurden, nur zu einer durchschnittlichen Zuverlässigkeit führen. Diese ist abhängig von einer guten Sichtverbindung und einer geringen Distanz zwischen Sender und Empfänger. Weiterhin zeigte sich, dass die Leistung des MSP430 Mikrocontrollers nicht ausreichte, um Tondaten mit einer höheren Abtastrate als 8 kHz zu komprimieren, was allerdings für die damalige Anwendung – Übertragung von Sprache – völlig ausreichend war.

Durch dieses Projekt konnten nachhaltig Rückschlüsse auf das Verhalten des MSB-430H und des CC1100 gezogen werden, welche in die Evaluierung der Systemarchitektur einfließen konnten.

Autonomous Wildlife Monitoring - Design, implementation, and evaluation of a wireless solution for nightingale song recording

Die Diplomarbeit von Frank Beier stellt für die vorliegende Diplomarbeit die vorhandene Mesh-Infrastruktur bereit. Deshalb soll an dieser Stelle nur auf die ausführliche Beschreibung in Kapitel 1.5. - Vorhandene Infrastruktur - verwiesen werden.

1.4. Lernstrategien und adaptiver Wert des Gesangs bei Nachtigallen

Das bisherige Verfahren zur Aufzeichnung der Messdaten ist sowohl zeitlich als auch personell sehr aufwendig. Ein Ausweitung des Feldversuchs durch Überwachung von weiteren Nestern ist mit den derzeitigen Mitteln somit nur noch in sehr begrenztem Umfang möglich. Bisher in das Projekt einbezogene Nistgebiete sind beispielsweise der Treptower Park in Berlin und der Große Zernsee in Golm bei Potsdam. Diese Areale unterscheiden sich in den Anforderungen voneinander. Während im von dichtem Schilf umgebenen Rand des Großen Zernsees die Nester verschiedener Paare sehr eng beieinander liegen, nistet die Nachtigall im Treptower Park dagegen fast ausschließlich in Sträuchern die sehr weit entfernt und meist durch Rasenflächen voneinander getrennt sind (siehe Abbildung 1.1b). Aufgrund der zentralen Lage ist immer mit Publikumsverkehr zu rechnen, auch in der Nähe von Nestern.

In Hinblick auf die spätere Weiterverarbeitung, wurden die bisherigen Aufnahmen an allen Standorten in sehr hoher Qualität erstellt.

Die Kriterien sind:

- 16 Bit Auflösung
- Abtastrate von mindestens 44 kHz für die Erfassung von Frequenzen bis 20 kHz
- Verzicht auf verlustbehaftete Kompression
- Verwendung von Richtmikrofonen

Die Aufnahmezeit und -länge ist dabei abhängig vom Fortschritt von Partnersuche, Brut und Fütterung der Nachtigall und verlagert sich während der Saison, welche zwischen 6 bis 8 Wochen beträgt (Abbildung 1.1c), von den Nacht- in die Morgen- und Abendstunden.

Mit Hilfe der Software ….. werden Störgeräusche gefiltert und die Frequenzen aus denen eine einzelne Strophe besteht sichtbar gemacht. Das Ergebnis ist in Abbildung 1.1a zu sehen. Wie bereits erwähnt, erfolgt die Zuordnung der einzelnen Strophen, deren Vorkommen später in eine Datenbank eingepflegt werden sollen, allein durch den Nutzer und wird nur durch das charakteristische Frequenzbild erleichtert.

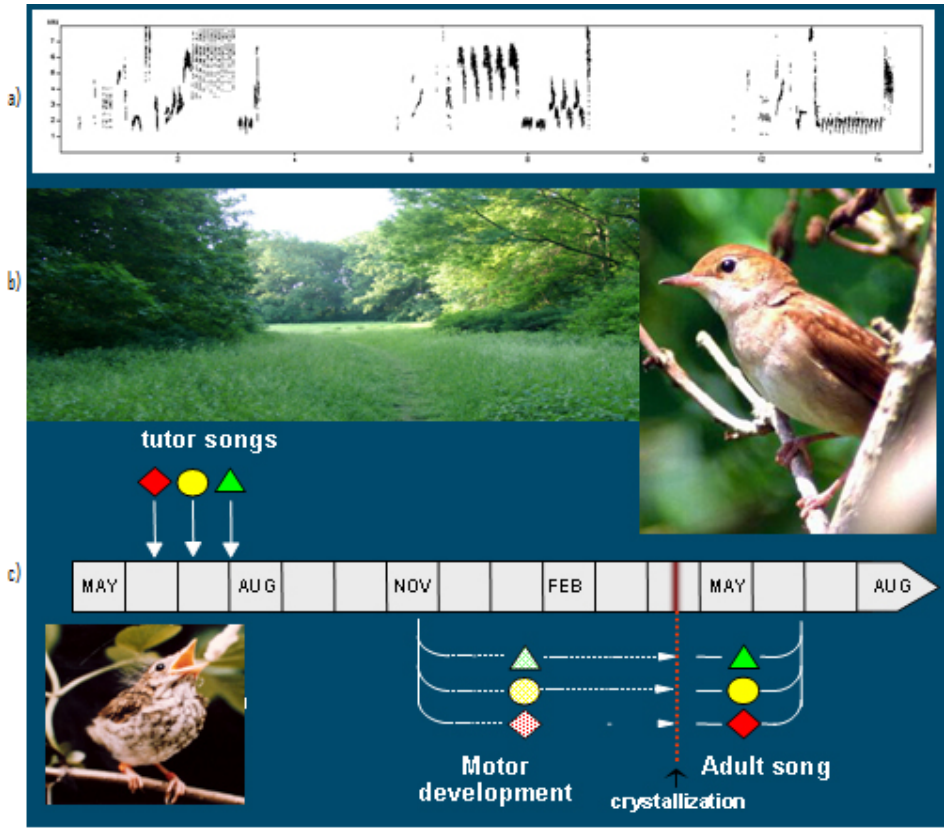


Abbildung 1.1: Nachtigall, a) einzelne Strophen, getrennt durch regelmäßige Pausen b) Nistgebiet Treptower Park Berlin c) Darstellung verschiedener Phasen der Saison [VB09]

1.5. Vorhandene Infrastruktur

Das Ziel war es ein unterstützendes System zu entwickeln, von nun „Voicelink-Netzwerk“ genannt, welches das bisherige Verfahren verteilter Aufnahmen vereinfacht und wieder eine Konzentration auf die eigentliche Forschung ermöglicht. Als erster Schritt ist es deshalb nötig, die Aufnahme von Strophen vollständig von der Anwesenheit von Personen während der Gesangszeiten zu trennen. Dazu soll ein Ad-hoc Netzwerk von „Basisstationen“ aufgebaut werden, die miteinander kommunizieren. Jede Basisstation soll die Anbindung mehrerer Mikrofone erlauben, wobei zwischen drahtgebundenen und drahtlosen Mikrofonen unterschieden wird. Drahtgebundene Mikrofone sind besonders kostengünstig und können direkt von der zentralen Stromversorgung der Basisstation profitieren. Hier wurde mit Hilfe von Extendern eine Lösung gefunden werden, die ohne Qualitätseinbußen Aufnahmen auch über den näheren Umkreis hinaus erlaubt. [FB09] In Fällen in denen sich diese Variante aufgrund von Hindernissen nicht eignet und auch der Aufbau einer neuen Basisstation zu aufwendig wäre, sollen drahtlose „Mikrofonstationen“ verwendet werden. Sie würden sich auch eignen, wenn lediglich ein Nest in einem bestimmten Gebiet anzubinden wäre oder besondere Anforderungen an einen geringen Strombedarf gestellt wer-

den. Derartige Insellösungen könnten zum Beispiel im Treptower Park Verwendung finden, da hier einzelne Nistgebiete teils durch große Rasenflächen voneinander getrennt sind (Abbildung 1.1b).

Sowohl Mikrofon- als auch Basisstationen müssen nach einem anpassbaren Zeitplan agieren. Ein Nutzer soll über den „Voicelink-Client“ diese individuell für jede Station mit Hilfe eines Verwaltungsprogramms pflegen und nach seinen Bedürfnissen verändern können. Im Versuchsgebiet vorhandene Mikrofone sollen automatisch von der Software erkannt werden. Gleichzeitig soll das Verwaltungsprogramm die Möglichkeit bieten, die bereits beendeten Aufzeichnungen aller Stationen an einem beliebigen Zugangspunkt abholen zu können. Abbildung 1.2 zeigt die geplanten Kommunikationswege. Der Client muss dabei nur anwesend sein, wenn neue Zeitpläne eingespielt oder Daten abgeholt werden.

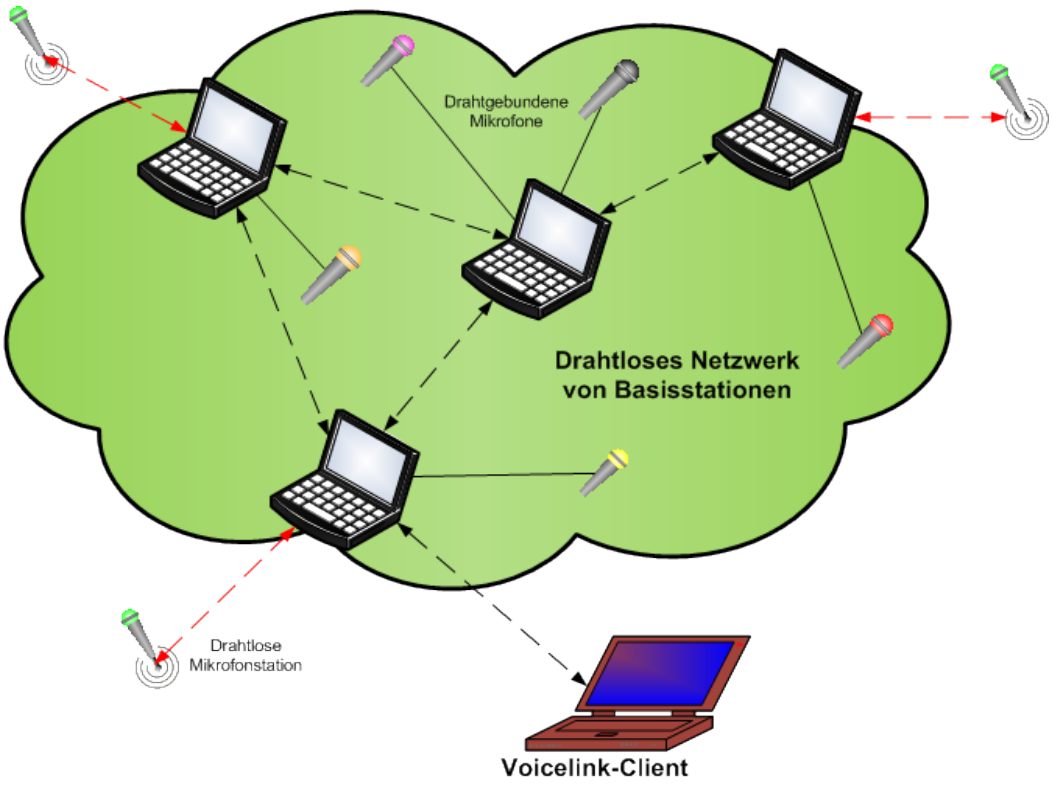


Abbildung 1.2: Im Projektaufbau wird unterschieden zwischen einem Netzwerk von einzelnen Basisstationen und den eigentlichen Mikrofonen. Diese können sowohl drahtlos (rot) als auch direkt (schwarz) angebunden sein. Ein Client übernimmt von außen die Programmierung. Die Umsetzung der Mikrofonstation ist Teil dieser Diplomarbeit.

Die Qualität aller Aufnahmen muss, nach den in Kapitel 1.4 genannten Kriterien von Abtastrate und Auflösung, mit dem bisherigen manuellen Verfahren konkurrieren können. Zusätzlich müssen folgende Eigenschaften für das Voicelink-Netzwerk und jede seiner Stationen gelten:

- Erweiterbarkeit des Systems durch weitere Basis- und Mikrofonstationen

- eine Aufnahme muss zu beliebigen Tages- und Nachtzeiten möglich sein
- die Länge einer einzelnen Aufnahme muss bis zu 8 Stunden betragen können
- der geltende Zeitplan innerhalb des Voicelink-Netzwerks muss konsistent sein
- die Speicherkapazität innerhalb des Netzwerks muss groß genug sein, um Aufnahmen von mehreren Tagen ohne Verlust zwischenspeichern zu können
- wartungsarme und ausreichend dimensionierte Stromversorgung
- Aufnahmen müssen in der Software …. verarbeitet werden können

Außerdem wird damit gerechnet, dass vielerorts auf Richtmikrofone verzichtet werden muss. Einerseits sind sie nicht in ausreichender Stückzahl vorhanden, andererseits wäre eine korrekte und stabile Ausrichtung auf das Ziel sehr aufwändig. Das heißt, dass hauptsächlich omnidirektionale Mikrofone innerhalb eines 5m Schutzradius um das Nest platziert werden müssen. Dieser Schutzradius ist notwendig, um die Nachtigall nicht zu stören und damit möglicherweise von ihrem Nachwuchs zu trennen. Ein Eingriff in diesen Bereich sollte nur einmalig erfolgen und eine sonstige Wartung des restlichen Systems, sofern unvermeidbar, vollständig auf das Gebiet außerhalb dieses Radius verlagert werden.

Bezüglich der örtlichen Bedingungen ergeben sich für den Aufbau des gesamten Systems weitere Einschränkungen und Besonderheiten die beachtet werden müssen. So müssen alle Teilkomponenten wasserdicht oder wenn nicht anders möglich zumindest spritzwassergeschützt sein, da die Erhebung der Daten durchgängig bei jedem Wetter möglich sein muss. Weiterhin hat sich herausgestellt, dass die Forderung einer wartungsarmen Stromversorgung ein großes Problem darstellt. Generell soll das hier gezeigte Konzept des Voicelink-Netzwerks nur ein Grundsystem darstellen. Mikrofon- und Basisstationen sollen so ausge-

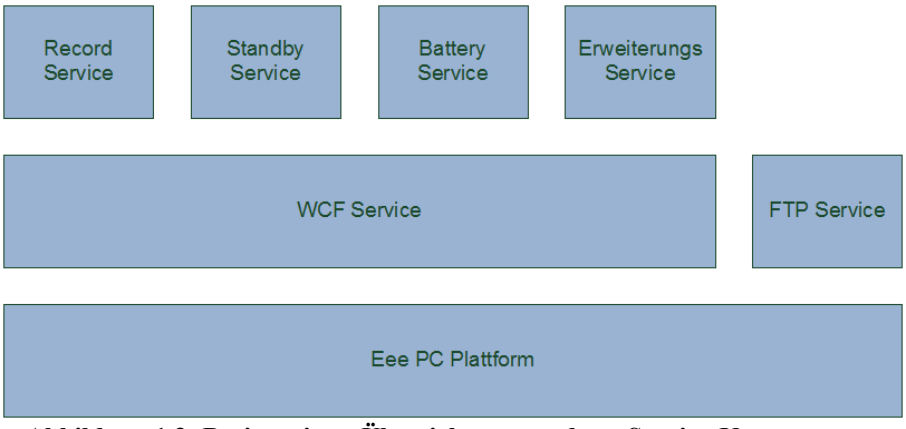


Abbildung 1.3: Basisstation - Übersicht verwendeter Service-Komponenten [FB09]

wählt und umgesetzt werden, dass später weitere Funktionen hinzugefügt werden können. Denkbar wäre die Aufzeichnung von Luftfeuchtigkeit, Temperatur, Sonneneinstrahlung oder die Einführung eines zusätzlichen Aufnahmemodus, bei dem an jeder Station selbstständig entschieden wird eine Aufzeichnung zu starten, wenn Frequenzen in der Umgebung auf Nachtigallgesang schließen lassen. Auch eine Videoaufzeichnung ist denkbar, um das Verhalten bei der Fütterung der Jungvögel zu beobachten.

1.5.1. Basisstation

Als Hardwareplattform wurde der Asus EEE PC 701 ausgewählt. Auf Basis des .NET Framework 3.0 wurde eine Software entworfen, die sich in Form verschiedener Dienste direkt in das verwendete Betriebssystem Windows XP integriert.

Alle vom Nutzer getroffenen und von der Station selbst generierten Informationen werden in Form von XML-Dateien gespeichert. Einige von ihnen, wie z.B. Aufnahmezeiten, werden redundant auf jeder Basisstation vorgehalten, auch wenn sie für die aktuelle Station zunächst scheinbar nicht direkt relevant sind. Sie werden dazu genutzt andere Stationen die vorübergehend nicht erreichbar waren, nachträglich zu synchronisieren. Diese Aufgabe übernimmt ein Service auf Basis der Windows Communication Foundation (WCF). Wird beispielsweise eine Aufnahme über die Verwaltungssoftware des Voicelink-Clients programmiert, so wird die Anfangs-, Endzeit und die global eindeutige Identifikationsnummer des Mikrofons in der dazugehörigen XML-Datei abgelegt. Wurde diese Datei an zumindest eine der Basisstationen übertragen, so können später hinzugekommene Stationen über den WCF-Service diese Änderungen erkennen und die Daten mit ihrer lokalen Version abgleichen.

Weitere Services und ihre Aufgaben im Einzelnen:

- Record Service:
    - Vorhalten des aktuellen Zeitplans
    - Starten von Aufnahmen
    - Speicherung von Dateien im RIFF-WAVE Format
    - Beenden der Aufnahme und anschließende verlustfreie Kompression mittels WavPack
    - Bekanntgeben der fertigen Datei

- Standby Service
  - Aktivieren und Deaktivieren des Standby Stromsparmodus
  - Erkennen aktueller Aufgaben, z.B. Aufnahme, Kompression, Dateiübertragung und damit aussetzen des Stromsparmodus
  - Bereitstellen einer „Service Time“ welche in einem vorher durch die Verwaltungssoftware definierten Zeitraum alle Basisstationen aktiviert, um zum Beispiel an einem Zugangspunkt durch einen Voicelink-Client neue Zeitpläne einzuspielen oder Aufnahmen abholen zu können
- Battery Service
  - Meldet den aktuellen Stand der angeschlossenen Batterie an die Verwaltungssoftware
  - Daten werden dazu aus einem USB-Voltmeter ausgelesen (nicht implementiert)
- FTP Service
  - Filezilla FTP-Server
  - Verwaltungssoftware kann auf diesen zugreifen, um Aufnahmen an einen Voicelink-Client zu übertragen
- Erweiterungs Service
  - Der Erweiterungs Service steht beispielhaft dafür, dass dem System weitere Funktionen durch weitere Services hinzugefügt werden können ohne Bestehende ändern zu müssen. Denkbar wäre hier die Unterstützung einer zeit- oder bewegungsgesteuerten Videoaufzeichnung. Auch der TCP-Server über den alle Mikrofonstationen ihre Kommunikation mit den Basisstationen abwickeln ist ein Erweiterungs Service.

Da nicht garantiert werden kann, dass sich alle Basisstationen in Funkreichweite befinden, wird auf eine Mesh-Netzwerk Implementierung von Microsoft Research zurückgegriffen. So können auch Stationen die mehrere hops entfernt sind, also in diesem Fall mehrere Sprünge innerhalb des Ad-hoc 802.11b/g WLAN Netzwerks, erreicht werden. Standardmäßig wird die gesamte Kommunikation zwischen den Basisstationen und einem eventuell vorhandenen Voicelink-Client über das Mesh-Netzwerk abgewickelt.

Eine Basisstation kann durch drahtgebundene Mikrofone erweitert werden. Hierzu werden mehrere USB-Soundkarten angeschlossen, deren Reichweite mittels Extendern auf Ethernetkabel-Basis von 5 m auf 50 m gesteigert werden kann. Die Laufzeit einer Basisstation beträgt, bei Verwendung einer Autobatterie mit 45Ah und einer täglichen Aufnahmezeit von 6 Stunden, etwa drei Tage.

Die Auswahl der Basisstationen, deren Software (ausschließlich des Mikrofonstation Erweiterungs Service) und die Umsetzung des Voicelink-Clients ist Teil der Diplomarbeit von Frank Beier. [FB09]

1.5.2. Voicelink-Client

Über den Voicelink-Client sind alle Stationen und die dort angeschlossenen Mikrofone sichtbar. Jeder normale Laptop mit WLAN wird mit Hilfe des Mesh-Netzwerktreibers und der Verwaltungssoftware zum Voicelink-Client. Dazu muss lediglich eine der Basisstationen in Reichweite sein, welche damit zum Zugangspunkt für alle Stationen im Ad-hoc Netz wird. Abbildung 1.4 zeigt wie ein für mehrere Stationen geltender Zeitplan angelegt werden kann. Im Hintergrund ist die Übersicht aller bisher erstellten Aufnahmezeiten zu sehen und ob diese bereits zum Download bereitstehen. Über eine weitere Registerkarte kann die bereits im vorigen Abschnitt genannte Service Time festgelegt werden, ein Zeitraum in welcher alle Basisstationen auch ohne eine anstehende Aufgabe den Standby-Stromsparmodus verlassen. Diese ist notwendig wenn sicher gegangen werden soll, dass alle fertigen Aufnahmen auch für die Verwaltungssoftware verfügbar sind. Dies wäre zum Beispiel nicht der Fall, wenn die hostende Basisstation schläft. Ein Aufwecken mittels einer Wake-On-LAN ähnlichen Methode ist nicht möglich.

Wurden alle nötigen Änderungen vom Nutzer getroffen, werden nach Ablauf von 5 min nach beenden der Verwaltungssoftware oder dem Ende der Service Time (je nachdem welches Ereignis zuletzt eintritt) alle Basisstationen automatisch wieder in den Stromsparmodus versetzt.

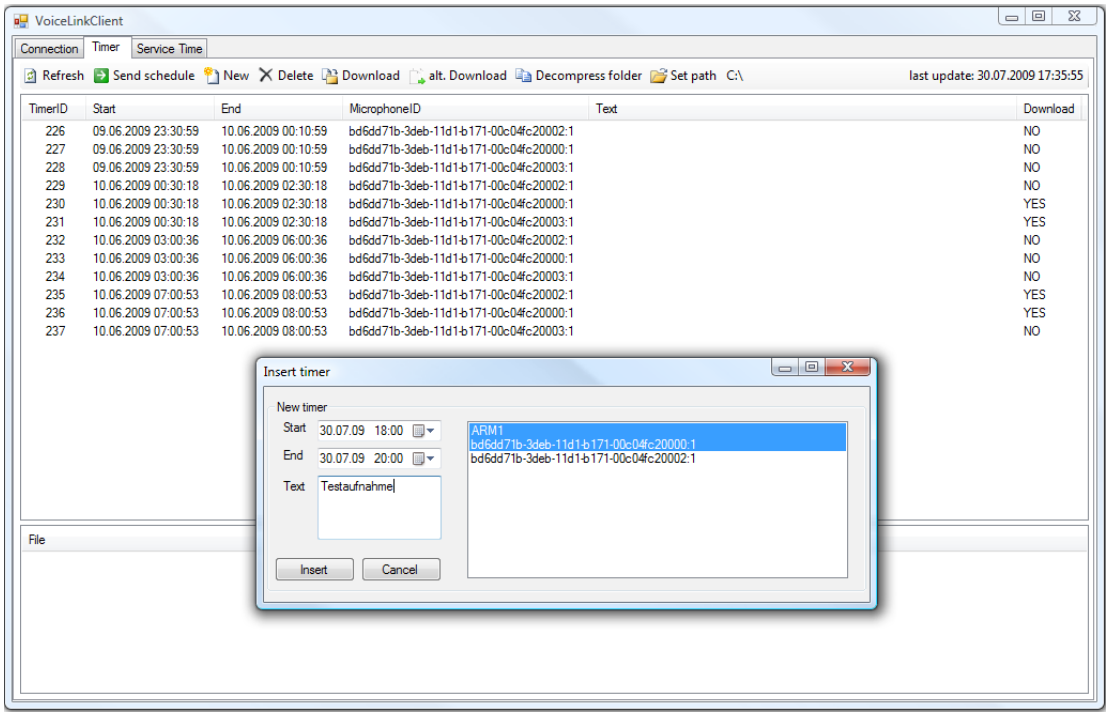


Abbildung 1.4: Anlegen von Aufnahmezeiten in der Verwaltungssoftware [FB09]

1.5.3. Problemstellung

Schon nach Auswahl der Basisstationen war ersichtlich, dass hier mehrere signifikante Nachteile hingenommen werden müssen. So konnte auf Basis von .NET schnell und komfortabel ein verteiltes System entwickelt werden, allerdings fordert die Wahl einer leistungsstarken Standard-Hardware ihren Tribut:

Wartungsaufwand

Aufgrund des hohen Energiebedarfs muss nach 3 Tagen die Autobatterie wieder aufgeladen werden. Diese wiegt je nach Hersteller zwischen 11 und 13 kg, sollen hiervon mehrere an einem Tag getauscht werden, so ist der logistische Aufwand und die Nachbereitung noch immer außerordentlich hoch.

Verlustleistung

Die Hitzeentwicklung der EEE PCs ist so hoch, dass der integrierte Lüfter bereits im Leerlauf durchgehend aktiv ist. Kommen hier zusätzlich Gehäuse hinzu, um sie vor Regen zu schützen, führt der Luftstau dazu, dass der Temperaturanstieg keinen zuverlässigen Betrieb mehr zulässt. Dies führt dazu, dass mögliche Gehäuse eine hohe Wärmeabführung besitzen müssen, welche nur durch teure Spezialanfertigungen erfüllt werden können.

Es ist abzusehen, dass neue effizientere Varianten des EEE PCs diese Probleme in den nächsten Jahren abmildern werden, von der Möglichkeit eine Station über einen Zeitraum von 6 bis 8 Wochen mit der selben Batterie zu versorgen, ist man allerdings weit entfernt.

1.6. Grundlagen

1.6.1. Abtasttheorem

Bekannt als Shannon-Niquist Abtasttheorem, legt es fest, dass ein analoges Signal mindestens mit der doppelten Rate der höchsten Teilfrequenz abgetastet werden muss. Nur so können später alle Frequenzen des analogen Ausgangssignal wiederhergestellt werden. Wird mit einer niedrigeren Rate abgetastet, können die wenigen Abtastpunkte dazu führen, dass der Eindruck einer niedrigeren „Geisterfrequenz“ entsteht (Abbildung 1.5 rechts).

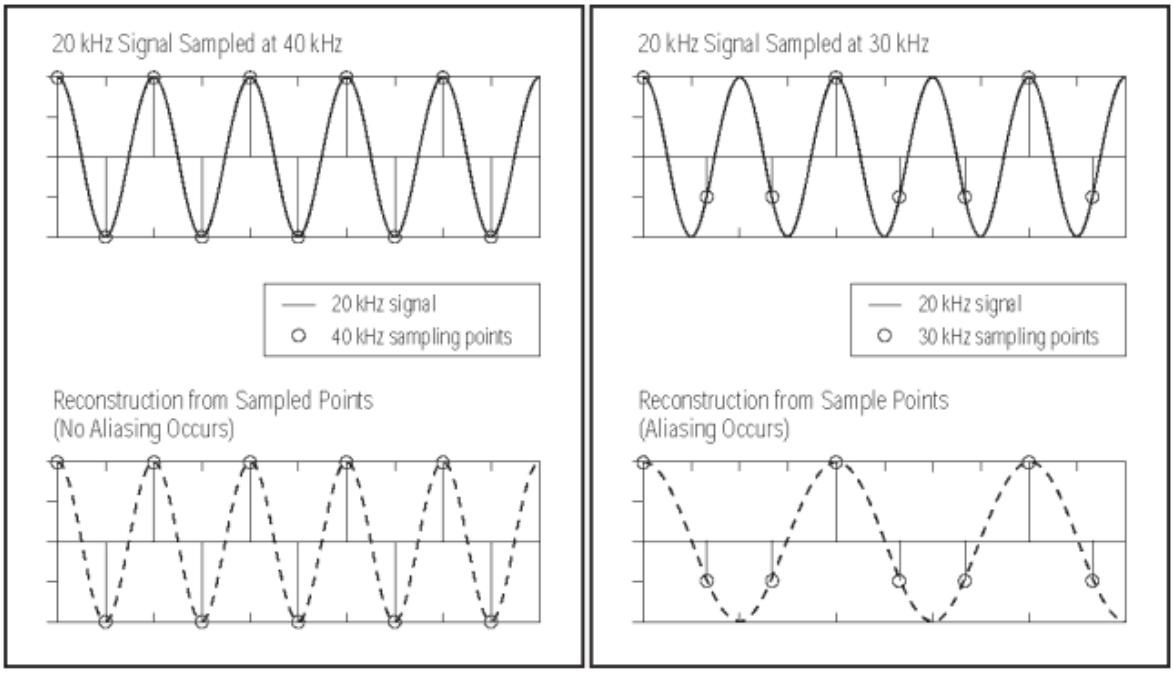


Abbildung 1.5: Bei 40kHz Abtastrate kann die 20kHz Ausgangsfrequenz wiederhergestellt werden, während bei 30kHz Abtastrate der Eindruck der halben Ausgangsfrequenz erweckt wird. [EMB05]

Geht man nun vom hörbaren Frequenzbereich eines gesunden Menschen aus, erstreckt sich dieser von 20 Hz bis 20 kHz. Eine Abtastung mit 40 kHz müsste nun, wie in Abbildung 1.5 links idealisiert dargestellt, mit jedem Abtastpunkt immer ohne jede Verschiebung die Amplitude abbilden, um das Ausgangssignal wiederherstellen zu können. Eine Verschiebung von einer viertel Periode würde hier eine Nulllinie liefern, eine achtel Periode eine 20 kHz Frequenz mit halber Amplitude. Um diese Probleme zu umgehen, wird meist die 2,2 (44,1kHz) oder 2,4 (48kHz) fache Abtastrate genutzt. [EMB05]

1.6.2. Signal-Rausch-Verhältnis

Nimmt man das Dezibel (dB) als Maß der Lautstärke des Schalls zur Grundlage, beträgt der Dynamikbereich des menschlichen Ohrs, also der hörbare Bereich vom leisesten zum lautesten Ton (Schmerzgrenze), 120 dB. In der Audiotechnik sollte dieser Bereich natürlich optimal abgedeckt werden, allerdings muss hier für jedes Bauteil ein Grundrauschen im unteren Messbereich und eine Verzerrung des ursprünglichen Signals im oberen Messbereich angenommen werden. Der nutzbare Teil zwischen beiden Bereichen (Abbildung 1.6) ist nach der Quantisierung des analogen Signals gleich dem digitalen Signal-Rausch-Verhältnis (SNR – signal-to-noise-ratio).

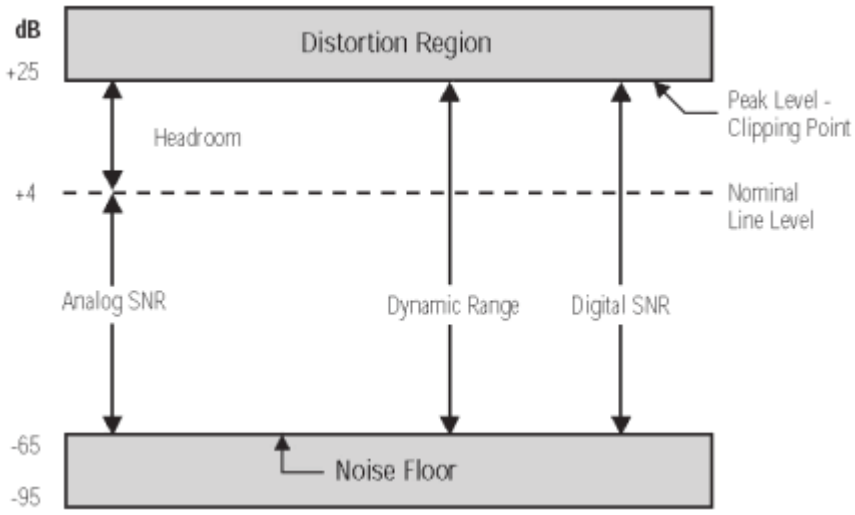


Abbildung 1.6: Nutzbarer Bereich des Signal-Rausch-Verhältnisses und des Dynamikbereichs [EMB05]

Das Dynamikbereich bei Analog-Digital-Wandlern ist eng mit der Anzahl der zur Verfügung stehenden Bits des Datenwortes verknüpft. Als Faustregel wächst für jedes zusätzliche Bit der Dynamikbereich um ungefähr 6 dB, wenn man die Konstante vernachlässigt.

$$\textit{Dynamikbereich}\ (dB) = 6.02\boldsymbol{n} + 1.76 \approx 6\boldsymbol{n}\ dB$$
$$\boldsymbol{n} = \textit{Anzahl an Bits des Datenworts}$$

Ein 16-Bit Analog-Digital-Wandler hat somit maximal einen Dynamikbereich von 96 dB. Wird für diesen AD-Wandler im Datenblatt ein Bereich von 95 dB angegeben, so kann davon ausgegangen werden, dass er ein nur sehr geringes Eigenrauschen und kaum Verzerrung bei der Quantisierung besitzt (und zwar von ungefähr 1 dB), er damit aber den gleichen Dynamikbereich wie das menschliche Ohr erreichen kann. [EMB05]

Die Diskussion vernachlässigt hierbei komplett zusätzliches Eigenrauschen, verursacht durch den analogen Teil der Schaltung vor dem Analog-Digital-Wandler.

1.6.3. Psychoakustisches Modell

Wie bereits beim Abtasttheorem erwähnt, besitzt der Mensch eine gewisse Hörschwelle, die unterhalb von 20 Hz und oberhalb von 20 kHz am stärksten ausgeprägt ist. Diese bezeichnet den Bereich, in dem ein Mensch eine Sinuskurve nicht mehr wahrnehmen kann. Der hörbare Bereich wird wiederum in mit der Frequenz größer werdende Frequenzgruppen unterteilt. Frequenzgruppen können die Hörschwelle anderer Komponenten durch Maskierungs- und Verdeckungseffekte erhöhen: Werden 2 Frequenzen aus der selben Frequenzgruppe zeitgleich aber unterschiedlich laut abgespielt, ist es möglich, dass nur eine davon wahrgenommen wird, wenn sich die Zweite unterhalb der Mithörschwelle befindet. Verlustbehaftete Kompressionsverfahren würden den Abstand beider Signale berechnen und ggf. die Leisere von beiden verwerfen oder nur in vereinfachter Form in den Datenstrom einfließen lassen.

Weiterführende Literatur: [UZ05], [JM09]

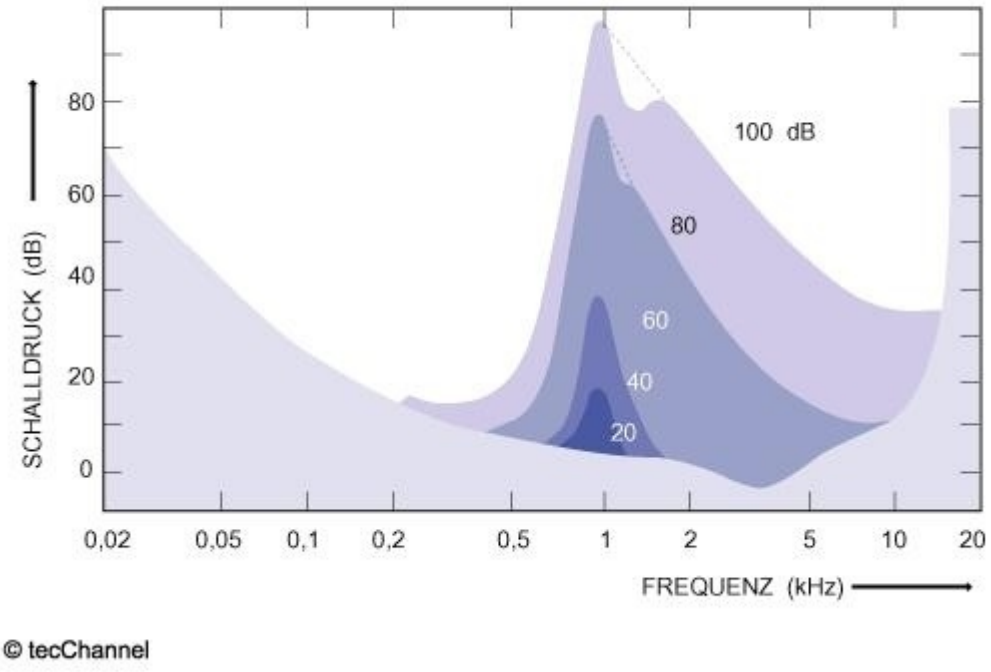


Abbildung 1.7: Hörschwelle über den gesamten Frequenzbereich und Darstellung der Mithörschwelle bei 1 kHz

2. Systemarchitektur

2.1. Anforderungen

Die Mikrofonstation soll die Zeitsteuerung, Aufnahme, Speicherung und Funkübertragung ermöglichen und damit eine Teilmenge der Basisstation-Funktionen. Vor allem soll im Gegensatz zu den Basisstationen mit nur minimaler Wartung während der aktiven Saison der Nachtigall ausgekommen werden und damit einen Mehrwert für die dauerhafte Beobachtung bieten. Sensorknoten mit geringem Energieverbrauch wie beispielsweise der MSB-430H sind speziell für die Langzeitüberwachung entwickelt worden. Die Besonderheit gegenüber normalen Sensornetzen ist in diesem Fall allerdings ein durch die geforderte Aufnahmequalität bedingtes konstant hohes Datenaufkommen. Hier muss eine Hardwareplattform ausgewählt werden, welche den benötigten Datendurchsatz zu jeder Zeit garantieren kann:

Abtastrate	Auflösung	Datenaufkommen pro Sekunde
44kSPS	16-Bit	86kB
48kSPS	16-Bit	94kB

Tabelle 2.1: Datenvolumen pro Sekunde ohne Kompression

Entsprechend ergeben sich mehrere Abhängigkeiten, die bei der Auswahl der Komponenten betrachtet werden müssen:

	Abtastrate	Auflösung	Datendurchsatz pro Sekunde	Energieverbrauch
ADC	x	x	x	x
Vorverstärker		x		(x)
Mikrocontroller	x		x	x
Massenspeicher			x	x
Funkübertragung			x	x

Tabelle 2.2: Abhängigkeiten innerhalb des Systementwurfs

- ADC
  - Der Analog-Digital-Wandler ist neben der Vorverstärkerschaltung die Kernkomponente für die spätere Aufnahmequalität. Ist es noch relativ einfach Modelle mit der gewünschten Abtastrate und Auflösung zu finden, müssen diese Datenmengen auch über eine geeignete Schnittstelle an den Mikrocontroller weitergegeben werden können.

- Vorverstärker
    - Die geringen Wechselspannungen von einigen Millivolt am Ausgang handelsüblicher (Elektret-)Mikrofone müssen für den in Relation großen Spannungsbereich am Eingang des Analog-Digital-Wandler vorverstärkt werden. Genauer gesagt handelt es sich hierbei um einen Spannungsverstärker, mit einer Verstärkung vom mehreren 100 bis 1000fachen der Eingangsspannung. Die zu erwartende Energieaufnahme ist hier selbst im unoptimierten Fall eher gering.
- Mikrocontroller
    - Im üblichen Fall wird das Eintreffen digitalisierter Audiosignale durch Auslösen von Interrupts dem Betriebssystem bekanntgegeben. Je nach ausgewählter Schnittstelle kommen hier im schlechtesten Fall somit zwischen 44000/48000 bei zwei Byte und 88000/96000 Interrupts bei einem Byte pro Übertragung zu Stande. Pro Interrupt sind zudem mehrere Takte für die Speicherung der Variablen auf dem Stack nötig, Speicherung der eintreffenden Daten noch nicht eingerechnet. Die Anforderungen an den Datendurchsatz und die Rechenleistung sind somit auch hier außerordentlich hoch.
- Massenspeicher
    - Mit einer bei Sensorknoten üblichen kurzen Zwischenspeicherung erhobener Daten ist es in unserem Anwendungsfall nicht getan. Wenn garantiert werden soll, dass eine Aufnahme auch vollständig den Empfänger erreicht, muss sie zunächst gesichert werden bevor eine Weiterleitung per Funk erfolgt. Hierdurch kann eine Fehlertoleranz gegenüber einem Abbruch der Funkverbindung erreicht werden. Ein anfallender Datendurchsatz von bis zu 337,5 MB pro Stunde und ein Gesamtvolumen von bis zu 2,7 GB pro Aufnahme (bei 8 Stunden) muss vom Massenspeicher unterstützt werden können.
- Funkübertragung
    - Aus der Idee zunächst alle Aufnahmen komplett zwischenzuspeichern, ergibt sich die Schlussfolgerung, dass die Funkübertragung keine zeitkritische Komponente mehr einnimmt wie es zum Beispiel bei einem Livestream der Fall wäre. Letztendlich muss hier aber ein Kompromiss zwischen geringem Energieverbrauch und Leistung eingegangen werden.

2.2. Evaluierung von Hardwareplattformen

Der wichtigste Schritt für den Aufbau des gesamten Systems, ist die grundlegende Plattform zu finden, an der mögliche weitere Komponenten angebunden werden können. Sie muss neben unseren Anforderungen eines niedrigen Energieverbrauchs also auch genügend Erweiterungsmöglichkeiten bieten. Beliebte Mikrocontroller zum Einsatz in Sensornetzen sind hierbei der MSP430 von Texas Instruments und die Modelle des unteren Leistungsbereichs der ARM-Kategorie. An der Freien Universität wurden von Dr. Dipl.-Ing Achim Liers mehrere Hardwareplattformen entwickelt, die diese Mikrocontroller nutzen und bereits in vielen Projekten Verwendung finden. Da sie in ihrem normalen Einsatzgebiet der Sensornetze eher mit geringem Datenaufkommen konfrontiert sind, wollen wir sie auf ihre Eignung als verlustfreies Audioaufnahmesystem mit hohem Datenaufkommen untersuchen.

2.2.1. MSB-430H

Der MSB-430H ist die verbesserte (Highspeed-) Version des Vorläufers MSB-430. Das Highspeed bezieht sich dabei allerdings hauptsächlich auf das verbesserte Funkmodul und nicht den Mikrocontroller. Hier wurde vom langsameren CC1020 auf den CC1100 gewechselt, welcher bei ähnlichem Energieverbrauch einen etwa dreifach so hohen Datendurchsatz erzielen kann. MSB steht übrigens hierbei für Modular Sensor Board, bei dem also durch einen modulbasierten Ansatz zusätzliche Hardware über Konnektoren mit den nach außen geführten Schnittstellen des MSP430 Mikrocontrollers verbunden werden kann. Bereits integriert sind ein Temperatur- und Feuchtigkeitssensor, LEDs und ein SD-Kartenleser. Bereits in meiner Studienarbeit [MF08] wurde der MSB-430H Sensorknoten für die Versendung eines Livestreams verwendet. Dabei wurde sowohl die maximale Funklast als auch die maximale Rechenleistung beim Sampling von 8 und 12-Bit Messwerten ermittelt. Der maximale Datendurchsatz des Funkmoduls limitiert oberhalb von 8-Bit (unkomprimiert) und 16kHz Abtastrate, der Mikrocontroller ist bei der Komprimierung von 12-Bit Werten via ADPCM bereits bei 8kHz vollständig ausgelastet. Zwar ist bei letzterem noch einzubeziehen, dass Rechenleistung für die Erstellung der Datenpakete inkl. CRC-Überprüfung verwendet wird. Würden die Daten nun nicht sofort weitergesendet werden, würde hier zwar kurzfristig zusätzliche Rechenleistung für die Kompression freigesetzt werden. Bedingt durch die Ansteuerung der SD-Karte und die Verwaltung des Dateisystems für die Zwischenspeicherung, würden diese Ressourcen allerdings wieder aufgebraucht werden.

Mikrocontroller		TI MSP430F1612
		16-Bit MSP430
Speicher	RAM	5 kB
	Flash-ROM	55 kB
Prozessortakt		Bis 7(10) MHz
Stromversorgung		2,7 – 3,6 V
Energieverbrauch		250µA - 115mA
Transceiver		CC1100 (CC1101)
Frequenzbereich	ISM-Bänder	868 MHz, 315 und 434 MHz
Funkreichweite		>600m
Datenrate (868 MHz)	raw max.	500kbit/s
	real	400kbit/s
Peripherie	On-Chip	12-Bit ADC
		12-Bit DAC
	On-Board	MMA7260Q Beschleunigungssensor Sensirion SHT11 Temperatur und Luftfeuchtigkeitssensor

Tabelle 2.3: Eigenschaften des MSB-430H [BKLS07]

2.2.2. ARM

Die Firma ARM Limited beschränkt sich auf die Entwicklung einer Reihe äußerst erfolgreicher 32-Bit Prozessorkerne die nicht selbst produziert, sondern ausschließlich an verschiedene Unternehmen lizenziert werden. Diese können den Prozessorkern als Grundstein für die eigentliche MCU mit weiterer Peripherie erweitern. Dazu zählen Flash, Caches, RAM, Analog-Digital-Wandler, Controller zur Ansteuerung von LCDs oder Debugging-Optionen für Entwickler. Neben äußerst leistungsstarken Prozessorkernen die z.B. im iPhone Verwendung finden, gibt es auch spezielle Versionen zur nativen Ausführung von Java-Bytecode (Kürzel J). [MSR05] Für unser Einsatzgebiet sind allerdings nur die Mikrocontroller der unteren Leistungsklasse interessant, in Abbildung 2.1 im gelben Bereich eingezeichnet, der ARMTDMI-S und der Cortex M3. Beide können den üblichen 32-Bit ARM-Befehlssatz ausführen, aber auch den kompakteren 16-Bit Thumb-Befehlssatz (Thumb2 beim Cortex M3) welcher eine Untermenge des ARM-Befehlssatzes abbildet. Sein Ziel ist die Codedichte deutlich zu erhöhen, durch den auch größere Projekte in kleine Flashspeicher geladen werden können. Thumb bringt einen Vorteil in der Codegröße, mit dem Nachteil, dass zunächst jeder Befehl dekomprimiert werden muss, welches wiederum zu einem

Geschwindigkeitsverlust führt. [SOC02] Dieser Nachteil wurde bei Thumb2 abgeschwächt, u.a. durch einen erweiterten Befehlssatz. Beide Prozessorkerne nutzen eine dreistufige Fetch-Decode-Execute Pipeline mit effizienter Ausführung kurzer bedingter Befehle ohne die Pipeline leeren zu müssen. Der Unterschied beider Prozessorkerne ist allerdings, dass der Cortex M3 von einer „von Neumann"- zu einer Harvard – Architektur gewechselt ist, welcher durch einen getrennten Befehls- und Datenbus eine schnellere Programmausführung nach sich zieht. Für den Programmierer bleibt diese Änderung allerdings verborgen, in dem sie sich wie eine von klassische Neumann – Architektur verhält. [CM306]

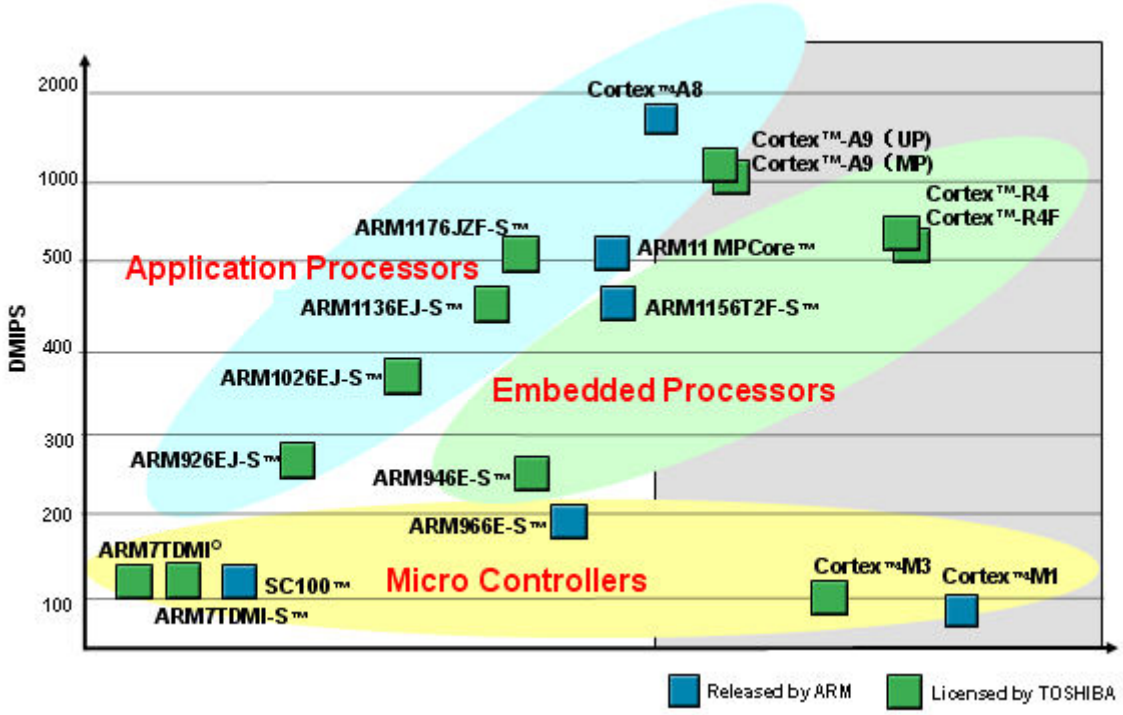


Abbildung 2.1: ARM-Prozessorkerne, dargestellt in Relation zu ihrer vorgesehen Anwendung, ihrer Rechenleistung (y-Achse) und der Releasereihenfolge (x- Achse) [TOS09]

2.2.2.1. STM32 PRIMER2

STMicroelectronics war einer der ersten Lizenznehmer des Cortex M3 Prozessorkerns. In Kooperation mit Raisonance entstand dabei um den Mikrocontroller herum eine Entwicklerplatine, die fast die komplette zusätzlich integrierte Peripherie an beispielhafte Hardware anbindet. Dabei wird sogar ein Betriebssystem bereitgestellt: Circle OS, benannt nach dem Primer1, einem kreisrunden Entwicklergerät. Die Ansteuerung von LCD und Touchscreen, Stromsparmodi der MCU, als auch neuerdings die Unterstützung vom Dateisystem

FAT32 mittels DOSFS sind bereits integriert. Der Primer2 erfreut sich dabei einer stetig wachsenden Gemeinschaft, die das Circle OS um weitere Funktionen erweitert. Die Programmierung erfolgt über die Ride7 Entwicklungsumgebung die kostenfrei zur Verfügung gestellt wird. Das Debugging erfolgt dabei über eine USB-Schnittstelle, über welche auch die integrierte Batterie geladen wird.

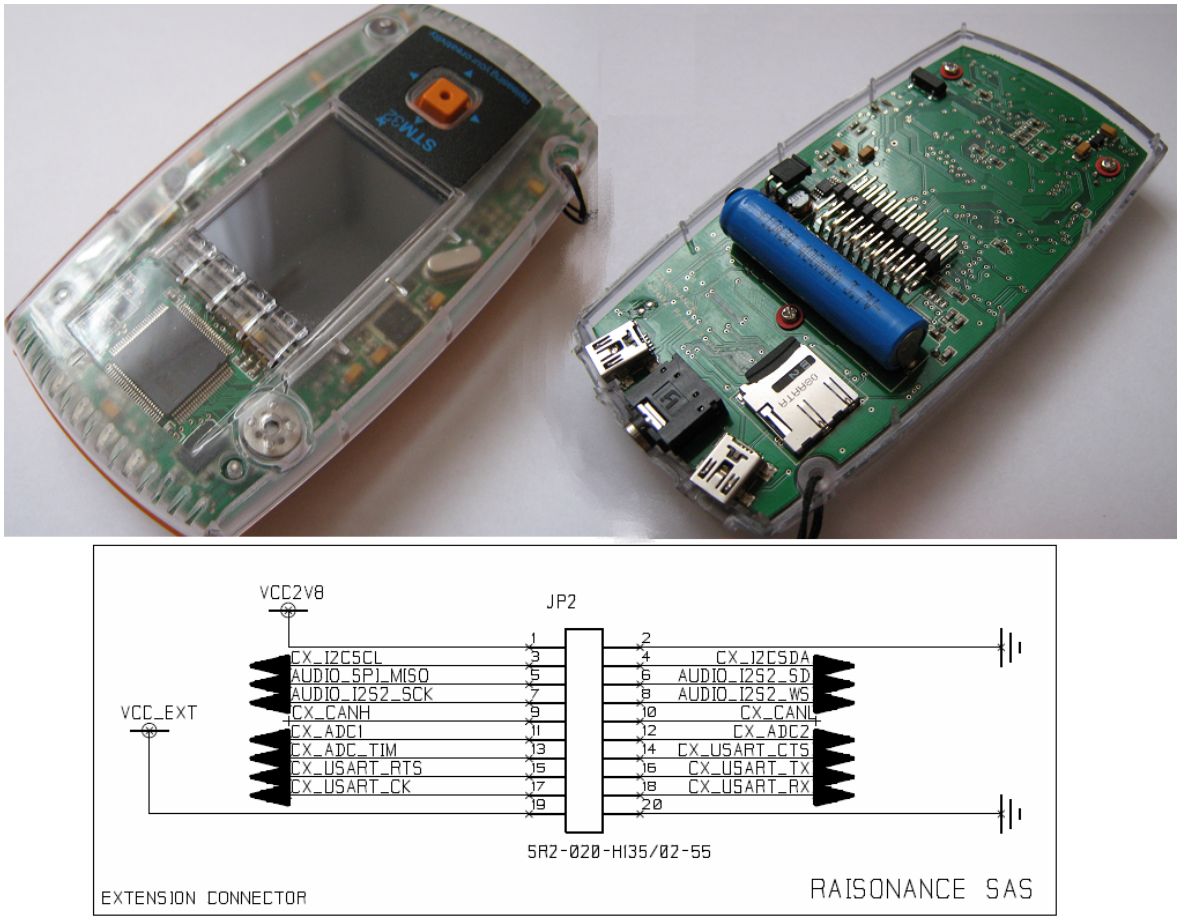


Abbildung 2.2: Vorder- und Rückseite des STM32 PRIMER2. Unten: Nach außen geführte Anschlüsse für die Erweiterung mit zusätzlicher Hardware [RAI09].

Ein großer Nachteil ist allerdings, dass beim Entwurf nicht alle ungenutzten Schnittstellen nach außen an den Extension Connector geführt wurden. So muss sich zum Beispiel I²S die Pinbelegung mit der einzigen erreichbaren SPI-Schnittstelle teilen (Abbildung 2.2).

Mikrocontroller		STM32F103VET6 32-Bit ARM Cortex-M3
Speicher	RAM	64 kB
	Flash-ROM	512 kB
Prozessortakt		Bis 72 MHz
Stromversorgung		5V USB (2,0 – 3,6 V Li-Ion)
Energieverbrauch (nur MCU)		<3 µA - 58mA [ST09]
Transceiver		-
Peripherie	On-Chip	12-Bit ADC 12-Bit DAC
	On-Board	Codec, Beschleunigungssensor, Touchscreen, Joystick, MicroSD

Tabelle 2.4: Eigenschaften des STM32 Primer2 [RAI09]

2.2.2.2. MSB-A2

Das neueste Mitglied der Modular Sensor Board Reihe war zur Einschätzung leider nur als Datenblatt verfügbar. Sehr interessant ist er wegen der hohen Rechenleistung und gegenüber dem MSB-430 weiter entwickelten Funkchip, welcher identisch zu dem des ebenfalls vorgestellten MSB-430H ist. Der MSB-A2 ist mit Sicherheit die zukünftige Plattform für Sensornetze bei denen zusätzlich ein Augenmerk auf Rechenleistung gelegt wird. Diese bietet die Möglichkeit Daten bereits lokal auszuwerten, entgegen einem zentralistischen Prinzip, bei dem alle erhobenen Messwerte zur Auswertung an einen anderen Punkt des Netzes weitergeleitet werden müssen. Hier kann somit bereits am Messpunkt entschieden werden, ob ein Ereignis relevant ist. (vgl. [BWB08]) Nun können ebenfalls aufwändigere Algorithmen (bezüglich Platz und Laufzeit) mit besseren Resultaten wie man sie in der Mustererkennung nutzt, in Betracht gezogen werden. Gleichzeitig wird die Funkübertragung eine Komponente mit großem Einsparpotenzial indem nur noch die Ergebnisse untereinander verglichen werden müssen.

Für unseren speziellen Anwendungsfall stellt die Hardware eine gute Basis dar, die Rechenleistung des Cortex M3 liegt zwar um etwa 35% höher gegenüber dem hier verwendeten ARM7TDMI-S [CM306], lässt aber durch seine hervorragenden Erweiterungsmöglichkeiten mit 47 nach außen geführten I/O-Ports [BWB08] sämtliche Freiheiten bei der Anbindung zusätzlicher Hardware.

Mikrocontroller		NXP LPC2387
		32-Bit ARM7TDMI-S
Speicher	RAM	98 kB
	Flash-ROM	512 kB
Prozessortakt		Bis 72 MHz
Stromversorgung		5V USB (2,7 – 3,6 V)
Energieverbrauch		150µA [LPC87] - 150mA
Transceiver		CC1100 (CC1101)
Frequenzbereich	ISM-Bänder	868 MHz, 315 und 434 MHz
Funkreichweite		>600m
Datenrate (868 MHz)	raw max.	500kbit/s
	real	400kbit/s
Peripherie	On-Chip	10-Bit ADC
		10-Bit DAC
	On-Board	Sensirion SHT11 Temperatur und Luftfeuchtigkeitssensor, MicroSD

Tabelle 2.5: Eigenschaften des MSB-A2 ([BWB08] und [LPC87]). Unterschiedliche Angaben gibt es bezüglich des minimalen Stromverbrauchs, hier wird von NXP mindestens 150µA angegeben.

2.2.2.3. Olimex LPC-P2378

Das Olimex LPC-P2378 ist eine reine Entwicklerplatine, das heißt dass beim Entwurf keinerlei Augenmerk auf Komponenten mit geringer Energieaufnahme gelegen hat, sondern dem Programmierer möglichst viele Möglichkeiten zur Programmierung und Debugging gegeben werden sollte. Es stehen ein UART-, ein JTAG-, Ethernet und USB-Anschluss zur Verfügung. Zur Speicherung von Daten wurde ein SD-Kartenleser angebracht, über den per MCI Daten deutlich schneller übertragen werden können als über SPI. Alle weiteren ungenutzten Schnittstellen wurden nach außen an drei Steckanschlüsse geführt. Der verwendete NXP Mikrocontroller ist im Grunde zu dem des MSB-A2 identisch, besitzt allerdings deutlich weniger RAM und keinen USB Host Controller. Wie beim STM32 PRIMER2 fehlt ein Funkmodul.

Mikrocontroller		NXP LPC2378
		32-Bit ARM7TDMI-S
Speicher	RAM	58 kB
	Flash-ROM	512 kB
Prozessortakt		Bis 60 MHz (Errata beachten)
Stromversorgung		5V USB, JTAG oder 9V Netzteil
Energieverbrauch		150µA - 150mA
Transceiver		-
Peripherie	On-Chip	10-Bit ADC
		10-Bit DAC
	On-Board	Ethernet und USB-Anschluss, SD-Card

Tabelle 2.6: Eigenschaften des Olimex LPC-P2378 [LPC78]

2.2.3. Fazit

Das Konzept des STM32 PRIMER2 ist äußerst interessant, besonders durch sein hervorragendes Betriebssystem und das vorhandene Display. Über letzteres könnte dem Anwender auf Wunsch viel Feedback über derzeitige Aktivitäten gegeben werden. Leider fällt es aufgrund seiner eingeschränkten Erweiterungsmöglichkeiten aus. Durch die Doppelbelegung von SPI und I²S wäre es nur schwer möglich beide Schnittstellen gleichzeitig zu nutzen, um zum Beispiel Tondaten über I²S zu empfangen und über SPI ein Funkmodul anzusprechen. Ohne der Implementierung vorgreifen zu wollen, so ist langfristig die gleichzeitige Aufnahme neuer Daten und Übertragung alter Aufnahmen anzustreben.

Die an der Freien Universität entwickelten MSB-Plattformen sind inzwischen weit verbreitet. Der Anwendungsfall in Sensornetzen bei dem es hauptsächlich auf geringen Stromverbrauch und nicht auf Leistung ankommt, ist wie geschaffen für den MSB-430H. Wie ich allerdings bereits in meiner Studienarbeit [MF08] festgestellt habe, ist er bei hohem Datenaufkommen überfordert. Für unseren Fall ist dagegen der MSB-A2 hervorragend geeignet: Die deutlich höhere Rechenleistung, ein schneller Systembus, über 20-mal größerem RAM und Vielfalt an Schnittstellen ist perfekt geeignet hohe Datenmengen zu verarbeiten – und dies bei kaum höherer Energieaufnahme gegenüber dem MSB-430H – entsprechende Optimierung vorausgesetzt.

Da diese Möglichkeit allerdings noch nicht als fertige Hardware zur Verfügung stand, musste eine Übergangslösung gefunden werden, von der das Projekt später mit wenig Aufwand auf den MSB-A2 portiert werden kann. Aus diesem Grund fiel die Wahl auf das Olimex LPC-P2378, aufgrund der ähnlich guten Erweiterungsmöglichkeiten über externe Schnittstellen und dem fast identischen Mikrocontroller.

2.3. Evaluierung von Komponenten zur drahtlosen Datenübertragung

Für die Kommunikation mit den Basisstationen wird nach einer drahtlosen Übertragungsmöglichkeit gesucht. Hierbei werden aus mehreren Frequenzbereichen beispielhaft stromsparende Transceiver ausgewählt und deren Eigenschaften auf ihre Eignung überprüft.

Anforderungen:

- Übertragungsgeschwindigkeit: mindestens 1 MBit/s
- Reichweite: mindestens 100m
- Robust gegenüber Übertragungsstörungen

Die folgenden Angaben zu Datenraten stellen nur die theoretisch maximal mögliche Leistung dar, die reale nutzbare Bandbreite kann um mehr als 50% darunter liegen.

2.3.1. 443 Mhz und 868 MHz ISM-Band

2.3.1.1. Texas Instruments CC1020 / CC1100

Der CC1020 und CC1100 sind Low-Power RF Transceiver, welche vor allem durch ihren Einsatz in der ScatterWeb MSB-430 Plattform, bekannt sind. Dadurch gibt es bereits eine ausgereifte Implementierungen, die z.B. automatische Paketbestätigung und Sendewiederholung, ein Paketformat und Prüfsummenauswertung nachrüsten. Die niedrigen Datenraten von maximal 0.15 Mbit/s und 0,5 Mbit/s (0,4 Mbit/s typ.) sind allerdings ein Ausschlusskriterium.

- Datenrate: < 0,15 Mbit/s (CC1020), < 0,5 Mbit/s (CC1100)
- Reichweite: bis 100m möglich
- Kollisionserkennung und -behandlung: Software

2.3.2. 2,45 Ghz ISM-Band

2.3.2.1. Nordic Semiconductor nRF24L01+ / nRF24LU1

Nordic Semiconductor bietet mehrere Transceiver im 2,4 Ghz Bereich an. Für den nRF24L01+ und nRF24LU1 [NOS08]werden neben den Entwicklerplatinen des Hersteller von einem externen Anbieter fertige Platinen angeboten, welche eine leichte Anbindung bei einem Prototypenaufbau ermöglichen. Ein Vorteil der Transceiver ist deren integrierte Behandlung von Paketverlusten, Adressauswertung und CRC-Berechnung.

- Datenrate: < 2 Mbit/s
- Reichweite: bis 100m möglich
- Kollisionserkennung und -behandlung per Hardware vorhanden

2.3.2.2. ZigBee / 802.15.4

ZigBee erlaubt den Aufbau eines selbst organisierenden und selbst heilenden Mesh-Netzwerks. Für diesen, eigentlich für die Hausgerätekommunikation entwickelten Standard, sind inzwischen Module in mehreren maximalen Sendestärken zu erwerben, welche Reichweiten bis zu 64km in freiem Gelände und 900m in Gebäuden erlauben. [ZIGB07]Um dies zu erreichen wird dazu das 868 Mhz oder das 915 Mhz ISM Band genutzt, allerdings ist letzteres nur in Nord- und Südamerika zur Nutzung freigegeben. Die ursprünglichen Module im 2,45 Ghz ISM Band teilen sich diesen Bereich mit Techniken wie u.a. Bluetooth und WLAN. Dadurch stellt sich die Frage nach der Störungsempfindlichkeit durch diese weit verbreiteten Techniken, allerdings gibt es hierzu unterschiedliche Angaben. Die ZigBee Alliance verweist ausdrücklich auf die CSMA Technik, welche ein Senden bei belegtem Kanal verhindert, zusätzlich zum Bandspreizverfahren DSSS und Frequenzmultiplexverfahren FDMA. Bei einem Testlauf auf der Hannover Messe wurde hier nur eine Fehlerrate von 2% ermittelt. [ZB] Auf eine Angabe über die verwendete Sendestärke wurde allerdings verzichtet, während in einer wissenschaftlichen Untersuchung [HAUER09] ein Zusammenhang zwischen Sendestärke und Fehlerrate gesehen wird.

Die Nutzung eines ZigBee Mesh-Netzwerks ist eine interessante Option. Ein mögliches Szenario wäre, dass eine feste Station als Gateway zur Außenwelt ausgewählt wird, die alle Daten des Netzwerks vorhält. Dieses Gateway sollte eine schnelle Datenverbindung vorhalten, z.B. per Ethernet oder WLAN, welche eine schnelle Abholung der Daten für den

Nutzer ermöglicht und die insgesamt niedrige Datenrate von maximal 0,125 Mbit/s zwischen den Knoten durch Pufferung ausgleicht.

- Datenrate: < 0,25 Mbit/s (0,125 Mbit/s typ.)
- Reichweite: von 10m bis 64km je nach Sendeleistung bei freier Sicht
- Kollisionserkennung und -behandlung per Hardware vorhanden

2.3.2.3. Bluetooth 802.15.1

Während der Ausarbeitung der Diplomarbeit sind mehrere Bluetooth-Module für den Low-Power Bereich erschienen, die auch höhere Datenraten bis zu 2,1 Mbit/s nach v2.0 EDR ermöglichen, z.B. das Roving Networks RN-41. Die Steuerung von Verbindungsauf- und Abbau erfolgt dabei über AT-Kommandos. Bluetooth stellt aufgrund dessen aktuell möglicher Geschwindigkeit eine interessante Alternative dar, erschienen aber zu spät, um sie in Betracht zu ziehen.

- Datenrate: < 2,1 Mbit/s
- Reichweite: von 10m bis 100m
- Kollisionserkennung und -behandlung per Hardware vorhanden

2.3.2.4. WLAN 802.11b/g

Aufgrund der Komplexität von 802.11b/g und der damit verbundenen Abstraktion von TCP/IP, waren stromsparende Module selten oder nur mit sehr geringer Geschwindigkeit zu erwerben. Trotzdem ist dieser Standard sehr interessant, denn er ermöglicht eine leichte Integration in bestehende Infrastrukturen, bietet gleichzeitig aber auch Ad-hoc Verbindungen an. Der Ad-hoc Modus ist außerdem bereits die Grundlage, auf der Frank Beier in seiner Diplomarbeit [FB09] ein Mesh-Netzwerk zwischen den Basisstationen aufbaut. Somit wäre bei den Basisstationen kein weiterer Hardwareaufwand nötig, um mit den ARM-basierten Plattformen zu kommunizieren.

- Datenrate: < 54 Mbit/s (typ. ~20Mbit/s 802.11g)
- Reichweite: bis 100m möglich
- Kollisionserkennung und -behandlung per Hardware vorhanden

Zur Festlegung wurden zunächst zwei Module verschiedener Hersteller auf deren Eigenschaften überprüft.

Roving Networks WiFly

Das RN-111b lag als Modul zu Tests vor. Es unterstützt nur den 802.11b Standard. Die Anbindung erfolgt über UART mit bis zu 921 kbit/s. Ohne Nutzung der Flusskontrolle ist der Betrieb mit nur 4 Verbindungen (PWR, TX, RX, GND) möglich, aber nicht empfohlen, da dies die einzige Möglichkeit ist eine zuverlässige Übertragung zu gewährleisten. Zum Test wurde das Modul mit einem USB zu RS232 Kabel verbunden. Per Textkommando wurde eine Verbindung zu einem PC aufgebaut und ASCII-Zeichen über ein Terminalprogramm übertragen. Bei ersten Tests wurden aufgrund des bekannten TCP-Acknowledge Problems, bei dem alle WLAN Module nicht mit verzögerten Bestätigungen umgehen können, nur maximal 40 kbit/s erreicht und stand danach nicht mehr zur Verfügung.

- Datenrate: ~ 200 kbit/s

Avisaro WLAN-Modul 2.0

Das Avisaro WLAN-Modul besteht aus einem Basismodul mit ARM7 Mikrocontroller und dem eigentlichen WLAN-Steckmodul. Die gesamte Ansteuerung wird dabei über das Basis Modul abgewickelt, welches zur Anbindung sowohl RS232 (2x), I2C (Master/Slave), SPI (Slave) oder CAN (2x) zur Verfügung stellt. [AVI09] Die eigentliche TCP/IP Kommunikation wird durch einen Text-Kommandomodus und einem neueren Paketmodus fast komplett vor dem Nutzer verborgen. Bei letzterem durch ein Handle-Verfahren, welches auch mehrere Verbindungen gleichzeitig, sowohl über TCP/IP als auch UDP, verwalten kann.

Optional gibt es durch die Modulbauweise die Möglichkeit, den Funktionsumfang durch zusätzliche Steckplatinen zu erweitern.

- Option “Datenlogger”:

  Dabei handelt es sich um einen SD-Karten Aufsatz, welcher es u.a. ermöglicht beliebige (Mess-)Daten mit Hilfe einfacher Basic Skripte aufzuzeichnen. Vom Hersteller wird explizit darauf hingewiesen, dass diese Skripte nicht dafür gedacht sind große komplexe Programme zu entwickeln, ermöglichen aber zum Beispiel die Erstellung einer einfachen Logik, um gepufferte Daten regelmäßig an Server zu übermitteln. Auch ohne zusätzliche Skripte kann über WLAN auf den Inhalt der Spei-

cherkarte über einen integrierten FTP-Server zugegriffen werden. Angaben über mögliche Schreib-und Lesegeschwindigkeiten gibt es hierzu allerdings nicht.

Basic Skripte (Beispiel Fehler: Referenz nicht gefunden) können auch ohne den SD-Karten Aufsatz verwendet werden, sie verlieren dadurch allerdings einen Großteil ihres Funktionsumfangs.

- Option Connector Boards:

  Es sind mehrere Trägerplatinen verfügbar, die einem den ersten Start bei der Nutzung der CAN- oder RS232-Schnittstellen erleichtern. Diese liefern zum Beispiel den fehlenden MAX232-Baustein und einen PC-kompatiblen Anschluss nach. Außerdem wird es damit möglich das WLAN-Modul als Brücke zwischen Ethernet und WLAN einzusetzen.

Das WLAN-Modul 2.0 befindet sich in einer ständigen Weiterentwicklung durch welche stetig neue Verbesserungen hinzukommen, die die Funktionalität erheblich erweitern oder vereinfachen.

- Datenrate: ~ 2,6 Mbit/s (Paketmodus2)

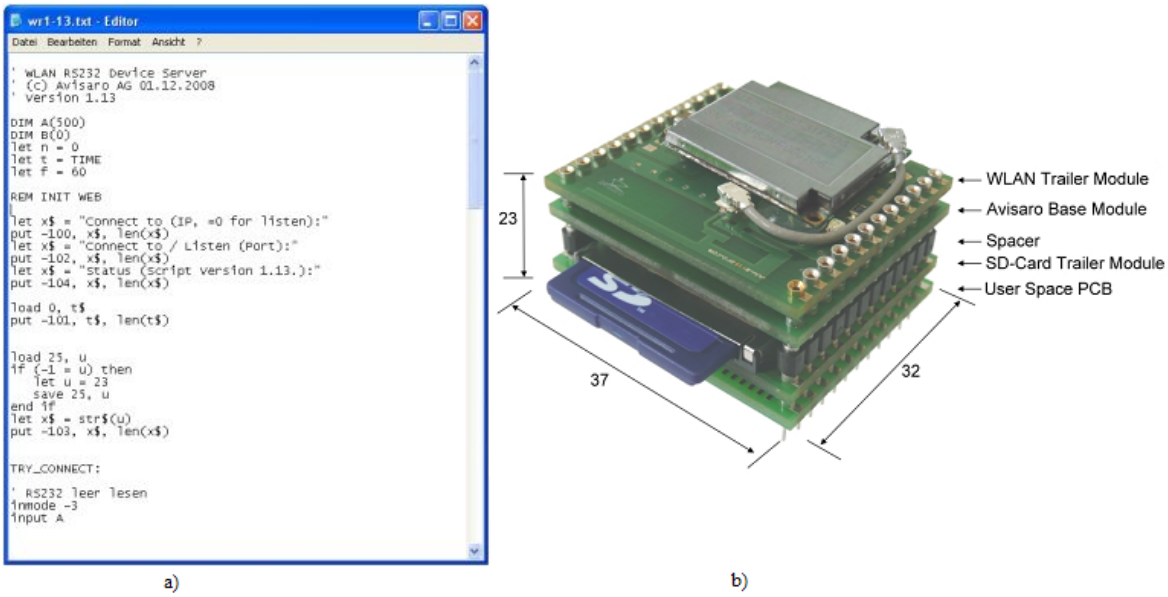


Abbildung 2.3: a) Eine Basic ähnliche Syntax erlaubt die Erstellung einfacher Skripte b) Bestandteile des WLAN-Moduls inkl. möglicher Erweiterungen. [AVI09]

2.3.3. Fazit

Aus der durchschnittlich täglichen Aufnahmezeit von 8 Stunden könnte abgeleitet werden, dass nun 16 Stunden zur Verfügung stehen, um die fertiggestellte Aufnahme an die Basisstation zu übertragen, über welche sie dann später den Voicelink-Clients zur Verfügung gestellt wird. Letztendlich bedeutet aber eine Einsparung bei der Energieaufnahme meist auch eine geringere Leistung bei der Mikrofonstation und damit eine längere Übertragungsdauer. Diese führt zu einem vielfach höheren Mehrverbrauch auf Seiten der Basisstation im selben Zeitraum. Somit kann nicht allein der beste Kompromiss aus geringem Energieverbrauch und Datendurchsatz herangezogen werden, sondern dem Datendurchsatz muss in diesem Fall eine höhere Bedeutung zugeordnet werden. Betrachten wir in Tabelle 2.7 aber zunächst exemplarisch die Energieeffizienz der einzelnen Funktechniken in Relation zum Datendurchsatz, um einen ersten Eindruck zu erlangen:

	CC1100 [CC11]	nRF24Lxx+ [NOS08]	Zigbee [ZIGB07]	WLAN Modul 2.0 [AVI09]
Datenrate	≤ 0,4Mbit/s	< 2Mbit/s	< 0,125 Mbit/s	≤ 2,6 Mbit/s
Energieverbrauch	104mW (3,3V)	120mW	148,5 mW	957mW
Verhältnis	3,85:1	16,6:1	0,84:1	2,72: 1

Tabelle 2.7: Verhältnis von Energieverbrauch zur erzielten Datenrate

Ist das Verhältnis von Datenrate zum Energieverbrauch bei der nRF24Lxx+ Reihe zunächst am Besten, so ist zu beachten, dass hier – im Gegensatz zu den anderen Chipsätzen - keine Erfahrungen zu realen Datenraten zu bekommen waren und diese möglicherweise weit unter den maximalen 2Mbit/s liegen. Außerdem werden später mehrere Mikrofonstationen in ein und dem selben Sendebereich mit Basisstationen kommunizieren. Senden diese zur selben Zeit, beträgt die zur Verfügung stehende Bandbreite jedes Einzelnen nur noch höchstens den Kehrwert der Gesamtanzahl der sendenden Teilnehmer. Hinzu kommen Verluste durch Kollisionen, einerseits verursacht durch weitere 2,4 GHz Funkstandards wie z.B. WLAN (sofern keine freien Nachbarfrequenzen unterstützt werden), als auch durch die Kommunikation untereinander.

Zwar sind Kollisionen nicht ausgeschlossen, das generelle Probleme des verringerten Datendurchsatz tritt somit in der Theorie nur bei den in Betracht gezogenen WLAN-Modulen nicht auf. Ihr derzeit erreichter Datendurchsatz ist nur ein Bruchteil des jeweiligen 802.11b/g Standards. Nehmen wir das schnellere Avisaro WLAN Modul 2.0 zur Grundlage, könnten im Idealfall 5 bis 7 Mikrofonstationen ohne nennenswerte Geschwindig-

keitseinbußen im selben Sendebereich agieren. Ob dies auch in der Praxis der Fall ist, kann allerdings nicht überprüft werden, da nicht genügend Testexemplare zur Verfügung stehen. Dennoch wird davon ausgegangen, dass das Avisaro WLAN Modul 2.0 die beste Möglichkeit darstellt, um Daten an die Basisstation zu übertragen. Sein Datendurchsatz ist im Vergleich am Höchsten, wodurch der ebenfalls hohe Energieverbrauch entschuldigt werden kann. Ein unbestrittener Vorteil ist, dass auf zusätzliche Hardware auf Seiten der Basisstation verzichtet werden kann, da diese bereits integrierte WLAN-Karten besitzen. Sollte sich außerdem später die Möglichkeit bieten das WLAN Modul 2.0 in einer Infrastruktur anstatt des angestrebten Ad-hoc Netzwerks einzusetzen, kann der zusätzliche Infrastruktur Power Save Modus (vgl. [SBBC]) genutzt werden, hier werden Einsparungen von bis zu 65% angegeben. [FLY08]

Das WLAN-Modul 2.0 ist als Standalone-Lösung für das gesamte Projekt der Mikrofonstation leider nicht geeignet, wohl aber als externe Funklösung für das Olimex LPC-P2378. Mit Hilfe der vorgestellten Basic Skripte können leider keine komplexen Programme erstellt werden wie wir sie benötigen. Eine Ansteuerung externer Hardware wird durch die Verhinderung des direkten Zugriffs auf die Firmware verhindert. SPI ist hier fest als Slave konfiguriert wie bei jeder externen Hardware, z.B. Analog-Digital-Wandlern. Mit Hilfe von Basic Skript Befehlen können zwar regelmäßig vom on-board Analog-Digital-Wandler Werte abgefragt werden, die Abtastrate beschränkt sich aber auf maximal 500 SPS. [AVI09] Natürlich besteht hier außerdem die ARM7-typische Beschränkung auf 10-Bit ADCs.

2.4. Analog-Digital-Wandler

Die verschiedenen Kombinationen bezüglich maximaler Abtastrate, Auflösung, Schnittstelle, Energieverbrauch – um nur einige wenige Kriterien zu nennen die man an Analog-Digital-Wandler stellen kann – zieht eine ebenso große Auswahl an Analog-Digital-Wandlern nach sich. Dadurch ist es nahezu unmöglich sich in einem überschaubaren Zeitrahmen einen kompletten Überblick über alle am Markt verfügbaren Modelle einer Kategorie zu verschaffen und das Modell mit den besten Eigenschaften auszuwählen.

Schauen wir uns deshalb zunächst die Eigenschaften der On-Chip-ADCs der möglichen Grundsysteme aus Kapitel 2.2. näher an:

	MSP430x1xx [MSP02]	LPC23xx/24xx [LPC09]	Anforderung [Kap. 1.4]
Auflösung in Bit	≤12	≤10	≥16
Samples/Sekunde	≤200k	≤400k	≥44k
	STM32 Cortex M3 [ST09]	STw5094A Codec [ST08]	
Auflösung in Bit	≤12	≤14	≥16
Samples/Sekunde	≤1000k	≤48k	≥44k

Tabelle 2.8: MSP430F1612 aus dem MSB-430H, LPC2378 vom Olimex LPC-P2378, STM32F103VET6 vom STM32PRIMER2 und dessen zusätzlicher On-Board Codec im Vergleich

Somit ist ersichtlich, dass kein interner Analog-Digital-Wandler der vorgestellten Mikrocontroller unseren Anforderungen genügt. Sie sind auch nicht für diesen Zweck entwickelt worden, sondern um zum Beispiel Werte von Temperatur-, Feuchtigkeits- oder Bewegungssensoren auszuwerten. Eine hohe Genauigkeit von 16-Bit, also immerhin 65536 Abstufungen innerhalb des Messbereichs (meist 0 bis 3,3V) ist in diesen Bereichen natürlich nicht erforderlich.

Um nun eine Auswahl möglicher externer Analog-Digital-Wandler treffen zu können, ist es wichtig eine geeignete Schnittstelle für die Kommunikation mit dem Mikrocontroller auszuwählen. Folgende Schnittstellen sind hierzu gebräuchlich:

- I²C ist ein 2-Signalleitungs-Datenbus wobei nur die Erste für die eigentliche Datenübertragung genutzt wird. Über den Zweiten wird das benötigte Taktsignal für die Auswertung der einzelnen Bits übertragen. Da über diesen Datenbus auch mehr als zwei Teilnehmer kommunizieren können – welche über verschiedene Adressen angesprochen werden – können alle Teilnehmer auch unterschiedliche Taktraten nutzen. Obwohl es hier bereits neuere Spezifikationen gibt, unterstützen die LPC23xx Mikrocontroller aber lediglich 400.000kbit/s. [LPC78]

- I²S ist im Gegensatz zu I²C ein 3-Draht-Datenbus mit klarer Master/Slave Rollenverteilung und nur 2 Teilnehmern. Der eigentliche Datenstrom wird hierbei wieder nur über eine Signalleitung übertragen, während die Zweite für das Taktsignal benötigt wird. Die Besonderheit von I²S als speziellen Datenbus zur ausschließlichen Audioübertragung wird im Nutzen der dritten Signalleitung ersichtlich: Hier wird selektiert ob es sich beim derzeit übertragenen Datenwort um einen Wert des linken oder rechten Audiokanals handelt. Somit kann ohne eine zusätzliche Codierung und

Auswertung in Software zwischen beiden Kanälen unterschieden werden. Unterstützt werden Samplingraten von 16 – 96 kHz bei Mono- oder Stereo, mit Datenwortgrößen von 8, 16 bis 32-Bit. [LPC78]

- SPI (Serial Port Interface) ist ebenfalls ein Datenbus mit einem Master, aber beliebig vielen Slaves. Er basiert auf vier Signalleitungen: zwei Datenleitungen (MOSI und MISO) für die Übertragung in jeweils Hin- und Rückrichtung, eine Signalleitung für das Taktsignal (SCK) und mindestens eine Signalleitung für die Auswahl (SSEL) eines Slave durch den Master. Die Taktraten können dabei von wenigen kHz- bis in den zweistelligen MHz-Bereich variieren. SPI ist somit die Schnellste der drei vorgestellten Schnittstellen.

Nur I²S und SPI bieten uns die benötigten Datenraten, wobei I²S durch seine Spezialisierung auf Tondaten-Übertragung die ideale Wahl darstellt.

Mit Verzicht auf einen womöglich geringeren Energieverbrauch, wurde ein Analog-Digital-Wandler ausgewählt, der schon einige Jahre auf dem Markt verfügbar ist und sich dadurch bereits in mehreren Projekten bewährt hat und somit gut dokumentiert ist.

2.4.1. Analog Devices AD1871

Der AD1871 ist ein 24-Bit Sigma-Delta Analog-Digital-Wandler mit Abtastraten von 32kHz bis 96kHz. Sigma-delta bedeutet hierbei, dass zum Beispiel bei 48kHz nicht 48.000 Einzelmessungen durchgeführt werden, um die gleiche Anzahl an 24-Bit Messwerten zu erzeugen, sondern deutlich mehr. Der Grund ist, dass intern nur ein Wandler mit deutlich geringerer Genauigkeit, z.B. 1-Bit vorhanden ist. Dieser wird pro Messwert beispielsweise 64- oder 128-mal angesprochen, um durch Dezimierung daraus ein PCM-Datum zu generieren. Diese Technik nennt sich Überabtastung (oversampling) und der Vorteil ist, dass einzelne Fehlmessungen auf ein weites Frequenzband verteilt werden, welches das Signal-Rausch-Verhältnis verbessert. [EMB05] Das entsprechend hohe Taktsignal im MHz-Bereich muss von außen (über MCLK Abbildung 2.4) durch einen zusätzlichen Quarz zugeführt werden.

Über SPI kann auf mehrere Konfigurationsregister zugegriffen werden, mit denen wir zum Beispiel Filter oder Auflösung einstellen können, während die PCM-Tondaten über unsere Wunschschnittstelle I²S an den Mikrocontroller übertragen werden. In unserem Fall benötigen wir nur einen der zwei angebotenen Kanäle des Wandlers.

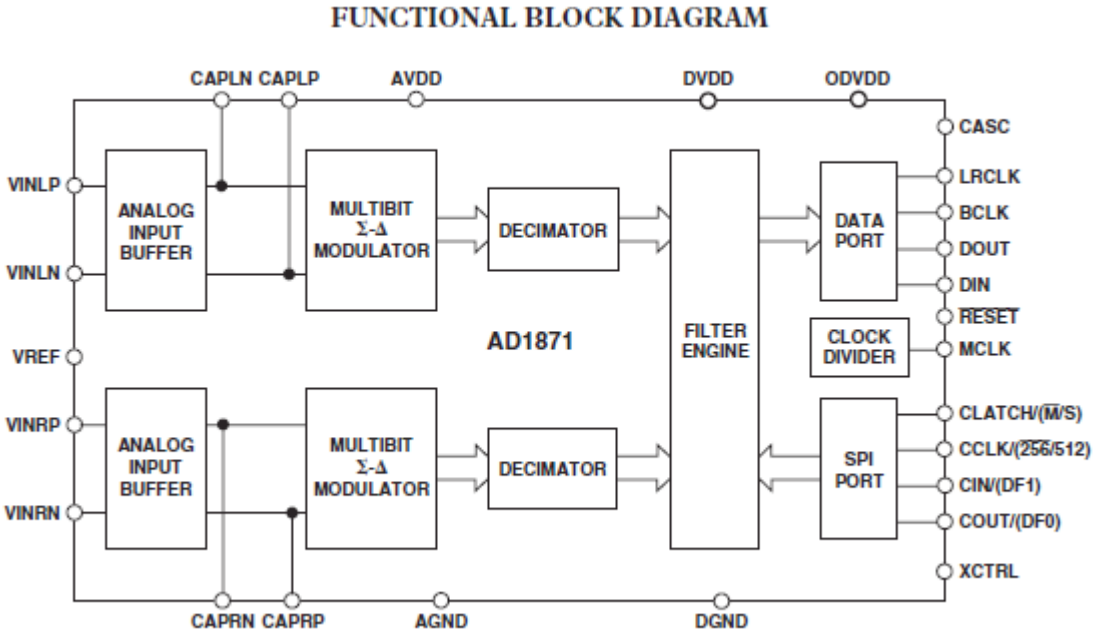


Abbildung 2.4: Über die analogen VIN-Eingänge (in unserem Fall benötigen wir nur den linken oder rechten Kanal) werden die Signale an den Sigma-Delta Modulator übertragen und später als PCM-Daten über I²S an den Mikrocontroller übergeben. [AD02]

Wie bereits erwähnt, wird zu Gunsten hoher Audioqualität auf eine geringere Energieaufnahme verzichtet. Diese beträgt etwa 58mA (Analog Current + Digital Current) bei 5V. [AD02] Sollte sich der AD1871 bewähren, kann auf ein Pin-kompatibles Nachfolgemodell umgestiegen werden.

2.5. Stromversorgung

Für die Nutzung eines Mikrocontrollers ist es im Grunde obligatorisch eine Stromversorgung über AA-Batterien, Lithium-Ion oder Solarzellen anzustreben. Allerdings musste die Idee einer Kombination aus Akkumulator und Ladeelektronik mit Solarzellen schon sehr früh verworfen werden. Das Versuchsgebiet am …. See in Golm ist mit hohem Schilf bewachsen, welches die Solarzellen verdecken und sie somit ineffizient werden lassen würde. Eine genaue Vorberechnung der gewonnen Energie wäre somit nicht möglich und der Ausfall der dazugehörigen Station wahrscheinlich. Ein ähnliches Bild zeichnet sich im Treptower Park ab. Hier sind die Sträucher in denen die Nachtigall nistet, entweder von hohen Bäumen umgeben oder sie sind direkt an angrenzenden Rasenflächen gelegen. Ersteres würde wieder die Frage der Energieeffizienz stellen, doch auch die mit direktem Sonnenlicht versorgten Nistplätze am Rand sind ungeeignet. So würde die spiegelnde Oberfläche der Solarzelle möglicherweise Besucher des Parks anlocken, welche entweder die Nachtigall vermehrt stören oder sogar das gesamte Equipment entwenden könnten.

Erschwerend kommt hinzu, dass sich bei ersten Messungen bezüglich der ausgewählten Entwicklerplatine eine sehr hohe Energieaufnahme im Tiefschlafmodus des Mikrocontrollers zeigte von 80mA bei 5V. Dies wurde in dieser Höhe nicht erwartet, zumal die Stromaufnahme des Mikrocontrollers laut Datenblatt nur etwa 150µA betragen sollte. (vgl. Kap. 4.5. Energieverbrauch)

Die Festlegung auf eine passend dimensionierte Stromversorgung konnte deshalb nicht wie eigentlich geplant erfolgen, so dass als Übergangslösung die selbe Energiequelle gewählt wurde wie für die Basisstationen: Die Nutzung einer 12V Autobatterie mit einem DC-DC Wandler der Marke Voltcraft, Modell SMP-20A, maximale Ausgangsleistung von 18W bei 12V. Die Ausgangsspannung ist dabei regelbar, so dass hier die entsprechende Eingangsspannung des Netzteilanschlusses des Olimex LPC-P2378 von 9V DC verwendet wird. Die maximale Belastbarkeit des DC-DC Wandlers beträgt 1,5A und ist damit ausreichend dimensioniert. Erwartet wird hier (unter Vernachlässigung von Wandlungsverlusten von 9V auf 5V und von 5V auf 3,3V): ca. 2,21W

Komponente	Stromaufnahme	Spannung	Energieaufnahme
Mikrocontroller	150 mA max.	3,3V	495 mW
Analog-Digital-Wandler	58 mA max.	5,0V	290 mW
WLAN-Modul 2.0	290 mA max.	3,3V	957 mW
Entwicklerplatine	~ 80 mA	5,0V	400 mW
		Gesamt:	~ 2,21 W

Tabelle 2.9: Schätzung der maximalen Energieaufnahme, unter Vernachlässigung von Wandlungsverlusten

Dem Autor ist durchaus bewusst, dass die Nutzung einer Autobatterie jedem Grundsatz eingebetteter Systeme widerspricht. Aufgrund der Rahmenbedingungen die die Entwicklerplatine vorgibt, ist somit allerdings zumindest eine Laufzeit von mehreren Wochen möglich.

Wenn die eigentlich angestrebte MSB-A2 Plattform als Prototyp verfügbar ist, sollte hier eine Neubewertung erfolgen. Hier kann entweder weiterhin eine durchgängig verwendete Batterie gewählt werden die dann kleiner dimensioniert werden kann oder im eher unwahrscheinlichen Fall ein Akkumulator der genügend Kapazität hat, um den Nachtbetrieb zu sichern. Am Tage könnte dann mit einer kleinen Solarzelle inkl. Ladeelektronik nachgeladen werden.

2.6. Gehäuse

Um bei Regen einen Kurzschluss zu vermeiden, wurde ein wasserfestes und geschlossenes Gehäuse des Modells Hofbauer Explorer 6 ausgewählt. Aufgrund des geringen Energieverbrauchs ist hier nicht wie bei den Basisstationen von einem Hitzeproblem auszugehen. Trotzdem sollte aufgrund der schwarzen Außenfarbe eine Platzierung in direktem Sonnenlicht vermieden werden. Es spricht nichts dagegen das Gehäuse zu vergraben, um es vor Sonne oder Entwendung zu schützen. Da keine Öffnung nach außen besteht, wurde diese auf der rechten Seite nachträglich hinzugefügt, um das Kabel eines Kfz-Adapters zum Anschluss zwischen Autobatterie und DC-DC Wandler und eine Verlängerung zum Anschluss eines Mikrofons an der Außenseite zu ermöglichen. Die Öffnung wurde danach wieder wasserdicht verschlossen.

2.7. Erweiterungsplatine

Es gilt nun eine entsprechende Verbindung zu schaffen, die die Entwicklerplatine mit dem WLAN-Modul und dem Analog-Digital-Wandler verbindet. Zusätzlich wird eine Verstärkerschaltung benötigt, um die geringe Wechselspannung die ein Mikrofon generiert, auf den Messbereich des Wandlers zu vergrößern. Nur so kann seine Auflösung optimal genutzt werden.

Im Wintersemester 06/07 wurde in einem Projekt am Institut für Energie- und Automatisierungstechnik der TU Berlin, Fachgebiet Elektronik und medizinische Signalverarbeitung, eine abgewandelte Form eines Moodlights entworfen und umgesetzt. [ELE41] Der klassische Gedanke eines Moodlights ist hierbei, dass Lichter auf Geräusche und Musik reagieren und aufgrund von Lautstärke oder Frequenz ihre Farbe ändern oder dimmen. In diesem Projekt wurde zur Signalverarbeitung der AD1871 Analog-Digital-Wandler verwendet, welcher auch bei unserer Mikrofonstation zur Digitalisierung genutzt werden soll. Um den zeitlichen Aufwand gering zu halten, wurde deshalb vom dort verwendeten Platinenlayout der analoge Teil der Schaltung um den AD1871 übernommen. Dieses Layout stellt deshalb die Basis unserer Erweiterungsplatine dar, auf der alle weiteren Komponenten ergänzt werden.

2.7.1. AD-Wandler

Generell wird für den AD-Wandler eine Spannung von 5V benötigt, welche sowohl zur Versorgung des ICs als auch zur Generierung der Referenzspannung benötigt wird. Von

den zwei zur Verfügung stehenden Kanälen des AD1871 wird nur der linke Kanal benötigt. Dieser hat wiederum zwei Eingänge: VINLN und VINLP. Da der differenzielle Modus genutzt werden soll, wird an VINLN eine externe Referenzspannung von etwa 2,25V angelegt. Diese wird durch einen Spannungsteiler von 5V auf 2,5V erzeugt, gefolgt von einem 10µF Elektrolyt-Kondensator gemäß Datenblatt (siehe Abbildung 2.5a). Intern wird nun diese Referenzspannung mit dem verstärkten Mikrofonsignal auf VINLP verglichen. Kann die Differenz laut Datenblatt maximal zwischen -2,828V und +2,828V betragen, so werden wir somit maximal etwa -2,25V bis +2,25V erreichen [ELE41] und damit nicht den kompletten Messbereich ausnutzen. Der Aufwand eine zusätzliche Spannung zu erzeugen wäre hier zu groß.

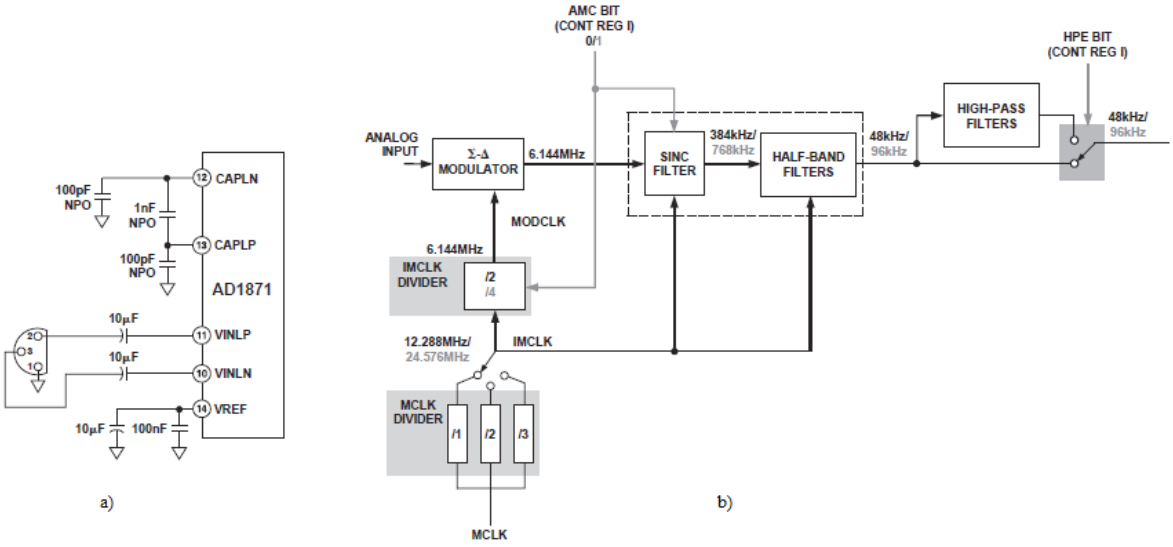


Abbildung 2.5: a) Beschaltung bei differenzieller Nutzung b) Interner Verlauf und Nutzung des externen Taktsignals MCLK und dessen mögliche Teiler

Über MCLK muss ein externes Taktsignal anliegen, welches ein Vielfaches der 256fachen Abtastrate ist, um dank Oversampling ein ideales Signal-Rausch-Verhältnis zu erhalten. Für die selbe Abtastrate von 44,1 kHz wie bei der Basisstationen, müsste hier eigentlich ein Quarz mit 11,2896 MHz gewählt werden, welcher allerdings momentan nur mit drastischem Aufpreis aus Asien bezogen werden könnte. Deshalb wurde ein kostengünstiger 12,288 MHz Quarz gewählt, wodurch die vormals schon mehrfach betrachtete Abtastrate von 48 kHz zu Stande kommt.

Um dem Programmierer völlige Freiheiten bei der Konfiguration des AD1871 zu lassen, wurde der SPI-Port über den EXT2-Konnektor der Olimex-Platine mit SPI1 des LPC2378 Mikrocontrollers verbunden. Die Übertragung der PCM Audiodaten wird über die I²S-Schnittstelle erfolgen.

2.7.2. Vorverstärker

Die Aufgabe des Vorverstärkers ist es, das Signal des Mikrofons, eine Wechselspannung welche üblicherweise im mV-Bereich liegt, auf maximal 2,5V zu verstärken, bevor sie über den im Datenblatt vorgesehenen 10µF Elektrolyt-Kondensator an den Analog-Digital-Wandler Eingang gelegt wird. Nur so ist es überhaupt möglich die meistverwendeten und günstigen Elektretmikrofone für die Tonaufnahme effektiv nutzen zu können. Betrachten wir die dazugehörige Schaltung in Abbildung 2.6: Über den Widerstand R1 bezieht das Mikrofon seine Versorgungsspannung, gleichzeitig bedeutet dies aber auch, dass Wechselspannung und Versorgungsspannung auf dem Weg zum Operationsverstärker voneinander getrennt werden müssen. Dies geschieht durch Kondensator C2, welcher nur den Wechselspannungsanteil passieren lässt und die Gleichspannung herausfiltert. Durch den Spannungsteiler der 47kOhm Widerstände R2 und R3 liegt der neue Nullpunkt der Wechselspannung nun bei 2,5V, also der Hälfte der 5V Versorgungsspannung. Der LMC6494 Rail-to-Rail Operationsverstärker, genutzt als nichtinvertierender Verstärker, verstärkt das Eingangssignal bis zur maximalen Höhe seiner Versorgungsspannung von 5V. Die Verstärkung berechnet sich wie folgt:

$$v = 1 + \frac{R4}{R5 + R10} = 1 + \frac{470\text{kOhm}}{570\text{Ohm} + R10}$$

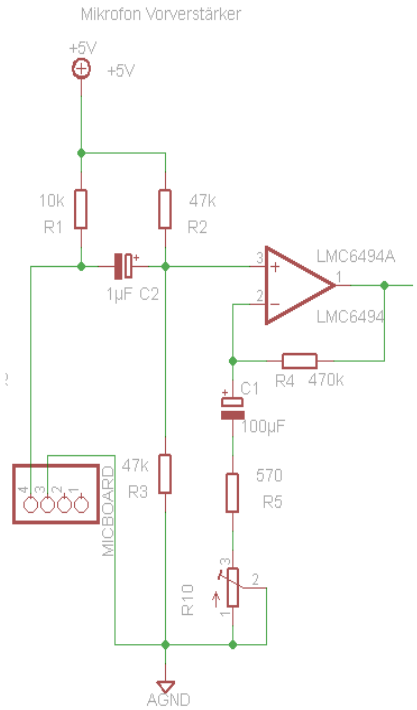


Abbildung 2.6: Verstärkerschaltung, mit regelbarer Verstärkung über Widerstand R10. In Kombination mit dem Mikrofonkompressor wird R1 weggelassen, da dieser die zusätzliche Versorgungsspannung die er bereitstellt nicht benötigt.

Die Verstärkung v kann somit die Eingangsspannung um den Faktor 2 bis ~ 825 verstärken. Welcher Wert letztendlich für R10 gewählt wird, hängt vom Schalldruck ab. Ist dieser sehr hoch, so kann die vom Mikrofon erzeugte Wechselspannung schnell über 20mV steigen und somit zu Übersteuern führen. Geht man aber vom Normalfall aus, beträgt die Wechselspannung ≤5mV (abhängig vom Mikrofon: Modell, Hersteller, interner Verstärkung etc.) und muss somit um das 500fache auf 2,5V verstärkt werden. R10 muss also etwa 400 Ohm betragen. Am Ausgang steht dann eine Wechselspannung zur Verfügung, deren Nullpunkt bei +2,5V liegt, wie zu Beginn unserer Betrachtung gewünscht.

Im Projekt der TU Berlin folgte hier ein Anti-Aliasing-Filter. [ELE41] Aufgrund eines Fehlers beim dortigen Tiefpassfilter zwischen Vorverstärkerschaltung und ADC-Eingang, welcher eigentlich Aliasing-Effekte bei Frequenzen oberhalb von 20kHz vermeiden sollte, lag die Grenzfrequenz bereits bei 200 Hz. Am Oszilloskop und in Aufnahmen waren Frequenzen oberhalb von 200 Hz trotzdem noch wahrzunehmen, auch wenn sie gedämpft in einer 200Hz Trägerfrequenz integriert waren. Dieser Teil der Schaltung wurde verworfen und nicht mehr korrigiert, um die eigentlich korrekte Grenzfrequenz von 20 kHz zu erhalten. Um die Mikrofonstation möglichst universell verwenden zu können, soll es später auch ohne Änderung von Bauteilen möglich sein, Frequenzen oberhalb von 20 kHz zu erfassen, wie sie zum Beispiel Fledermäuse erzeugen. Diese Idee der Anwendung liegt nahe, so bietet der Analog-Digital-Wandler durch eine halbierte Dezimierung der durch Oversampling gewonnenen Messwerte auch eine Abtastrate von 96kHz – wenn auch mit Einschränkung des SNR. Damit könnten in der Theorie auch Frequenzen bis 40kHz erfasst werden, ein entsprechendes Mikrofon vorausgesetzt. Ob dies auch in der Realität möglich wäre, da das Datenblatt des AD1871 explizit auf eine ausschließliche Nutzung bis 20 kHz hinweist, ist unklar. [AD02] Dies soll aber nicht Gegenstand der aktuellen Arbeit sein, sondern nur ein kleiner Exkurs für mögliche künftige Entwickler an diesem Projekt.

2.7.3. Mikrofonkompressor

Nach ersten Testmessungen zeigte sich schnell ein Designfehler in der Vorverstärkerschaltung, welcher zunächst nicht bedacht wurde: Durch SD-Karten Zugriffe wird kurzzeitig ein Spannungsabfall in der Versorgungsspannung erzeugt, welcher durch den Vorverstärker mitverstärkt wird. Dieser Spannungsabfall ist eigentlich sehr gering, tritt aber durch die 500fache Verstärkung deutlich in den Aufnahmen zu Tage (siehe Kapitel Auswertung: Schaltungsanalyse). Um dies zu umgehen, wurde nach einer von der Versorgungsspannung

unabhängigen Vorverstärkung gesucht. Hier wurde der „SMD-Mikrofonvorverstärker 73042 mit Limiter und Rauschsperre“ von ELV ausgewählt. [ELV09] Dieser Platine bietet neben einer Spannungsversorgung über Batterien (2,5V – 5,5V) und einer Transistor basierten Spannungsverstärkerschaltung, zusätzlich den Mikrofonverstärker SSM2167 von Analog Devices [AD09] mit variabler Kompression und Rauschsperre (Schaltbild Abbildung 2.7). Umgesetzt ist hier allerdings nur die Rauschsperre und der Limiter. Die Rauschsperre bewirkt, dass ein Eingangspegel unter -55dB (1mV) oder -40dB (10mV) nicht am Ausgang des SSM2167 ausgegeben wird. Dieser Pegel wird umgekehrt proportional über den Drehpotentiometer R6 eingestellt (Tabelle in Abbildung 2.8a). Auf eine Einstellungsmöglichkeit der variablen Kompression, die der bekannten AGC (Automatic Gain Control) ähnelt, wurde verzichtet. Diese würde über einen variablen Widerstand R7 erfolgen. Hier wurde allerdings ein fester 1kOhm Widerstand verwendet, welcher laut Datenblatt als Null Ohm interpretiert wird und somit eine Kompression von 1:1 erzielt: Ruhigere Bereiche werden somit nicht verstärkt, das Signal bleibt erhalten. Sollte sich die Amplitude des verstärkten Signals allerdings dem Grenzbereich nähern, wird dieses mit einem Faktor <1 ausgegeben (Abbildung 2.8b). Sie wird also limitiert bzw. komprimiert, daher der Name Mikrofonkompressor. Dies verhindert ein Übersteuern an den Eingängen nachfolgender Hardware. Da die eigentliche Anwendung die Nutzung von Mikrofonen an Line-In Eingängen ist, beträgt die maximale Amplitude etwa 0,81V (gemessen, siehe Auswertung) und muss für die Nutzung am Analog-Digital-Wandler noch einmal auf bis zu 2,5V verstärkt werden. Die Platine wird somit vor unsere ursprüngliche Vorverstärkerschaltung gesetzt und deren Verstärkung auf 3,08 gesenkt. R10 (Abbildung 2.6) wird dazu auf 225kOhm erhöht.

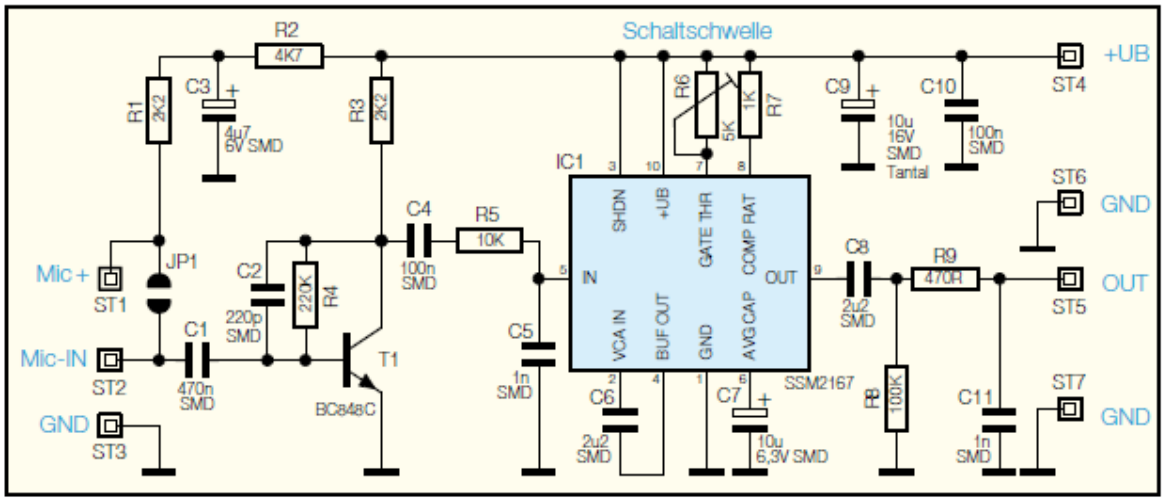


Abbildung 2.7: Schaltbild der SMD-Platine [ELV09] ST5 und ST7 sind im Layout vertauscht.

Noise Gate (dBV)	Value of R_{GATE}
–40	0 Ω (short to V+)
–48	1 kΩ
–54	2 kΩ
–55	5 kΩ

Compression Ratio	Value of R_{COMP}
1:1	0 Ω (short to V+)
2:1	15 kΩ
3:1	35 kΩ
5:1	75 kΩ
10:1	175 kΩ

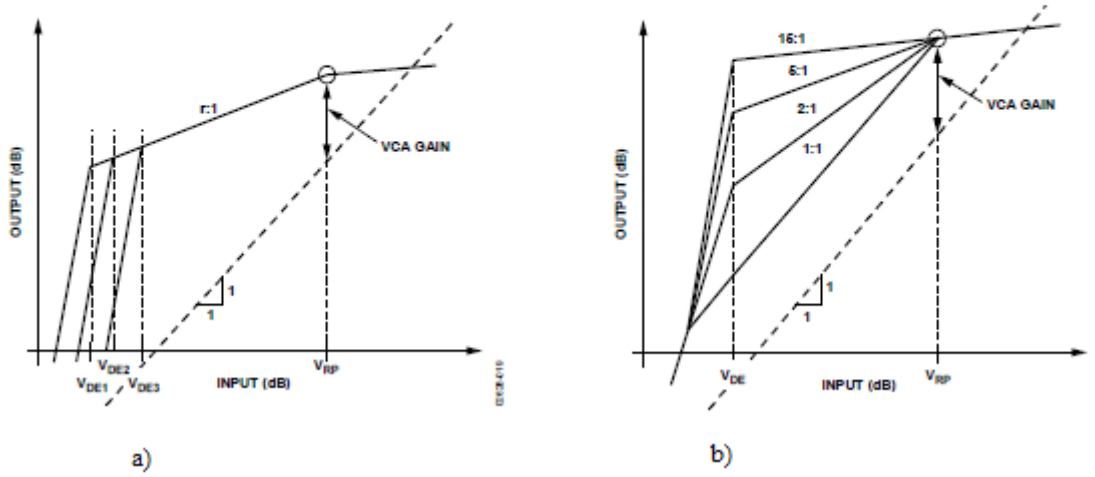


Abbildung 2.8: [AD09] a) Grenzbereich der Rauschunterdrückung, einstellbar über den Drehpotentiometer R6 b) Funktion des Limiters (Kompression 1:1), Kompression kann nur durch Austausch von R7 geändert werden. (siehe Abbildung 2.7)

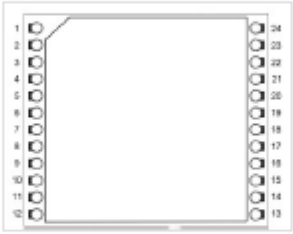
Der Widerstand R1 kann entfernt werden, da die zusätzliche Stromversorgung für Elektretmikrofone nicht mehr benötigt wird. Die Kombination beider Vorverstärker ist natürlich nur eine nachträgliche Notlösung, obwohl mehrstufige Verstärkerschaltungen durchaus üblich sind. Folgt die Migration zur MSB-A2 Plattform, sollten beide Ansätze aber in einer neuen Version der Erweiterungsplatine einfließen, welche schon aus dem Grund nötig ist, da die Pinbelegung der Olimex Extension Connector nicht übernommen werden kann.

2.7.4. Funkmodul

Das Avisaro WLAN-Modul 2.0, genauer dessen Basismodul, wird über die 4 SPI-Signalleitungen MISO, MOSI, SCK und CS (SSEL) für die Datenübertragung an SPI/SSP0 angebunden. Zusätzlich benötigen wir die Reset-Leitung über Pin 14. Die 3,3V Spannungsversorgung wird sowohl mit VCC als auch mit VBAT verbunden, Masse mit GND. Zusätzlich gibt es die Möglichkeit über EINT0 einen externen Interrupt auszulösen, mit dem das Basismodul aus dem Tiefschlafmodus geweckt werden kann. Bei Tests funktionierte dies allerdings nie zuverlässig, so dass EINT0 in unserer Implementierung nicht mehr genutzt wird. Für die Weiterentwicklung ist die Dokumentation von EINT0 aber möglicherweise sinnvoll, so dass sie hier im vom Avisaro stammenden Original wiedergegeben wird: Auf dem Basismodul „befinden sich auf der unbestückten Seite einige messingfarbene Flächen. Zwei große, sechs kleine nebeneinander und ein einsames kleines in der Ecke. Dieses ein-

zelne kleine Plättchen ist EINT0.“ Die Ansteuerung erfolgt über ein Active-Low (wie bei Reset).

No	I/O	Name		Name	I/O	No
1	P	VBAT		VCC (3.3V)	P	24
2	O	LED1		n.c.	-	23
3	O	LED2		n.c.	-	22
4	I	KEY		n.c.	-	21
5	-	n.c.		n.c.	-	20
6	-	n.c.		n.c.	-	19
7	-	n.c.		n.c.	-	18
8	-	n.c.		n.c.	-	17
9	I	CS		n.c.	-	16
10	O	MISO		n.c.	-	15
11	I	MOSI		Reset	I	14
12	I	SCK		GND	P	13



'I/O': pin is input, output or power, 'n.c.': do not connect, leave open

Abbildung 2.9: Pinbelegung des Avisaro Basismoduls [AVI09]

2.7.5. Schaltplan und Platinenlayout

Auf Basis des vorliegenden Platinenlayouts der Vorverstärkerschaltung für den AD1871 von der TU Berlin [ELE41], wurden die restlichen Komponenten mit Hilfe des grafischen Layout-Editors EAGLE in der Version 5.4.0 Light integriert. Mit der Light-Version können nur Platinen mit 2 Signal-Lagen mit einer maximalen Größe von 100 x 80mm erstellt werden, was für unseren Anwendungsfall völlig ausreichend ist. Generell sollte immer angestrebt werden so wenig Lagen wie möglich zu verwenden, um die Produktionskosten niedrig zu halten.

Generell folgt das Platinenlayout den Design-Empfehlungen von e2v zum Entwurf von Platinen mit analogen und digitalen Signalleitungen [E2V09] welche sich wie folgt zusammenfassen lassen:

- Getrennte Masseflächen für den analogen und digitalen Bereich
- Beide Masseflächen sollten nur in einem Punkt verbunden sein, um den Rückfluss von Spannungen aus dem digitalen Bereich über den Analogen zu vermeiden
- AD-Wandler sollten am Übergang beider Flächen positioniert werden, da sie getrennte Analoge und Digitale Bereiche haben

- Signalleitungen sollten sich nur diagonal Kreuzen (und natürlich nicht auf der selben Ebene)

Eine Bauteileliste befindet sich im Anhang.

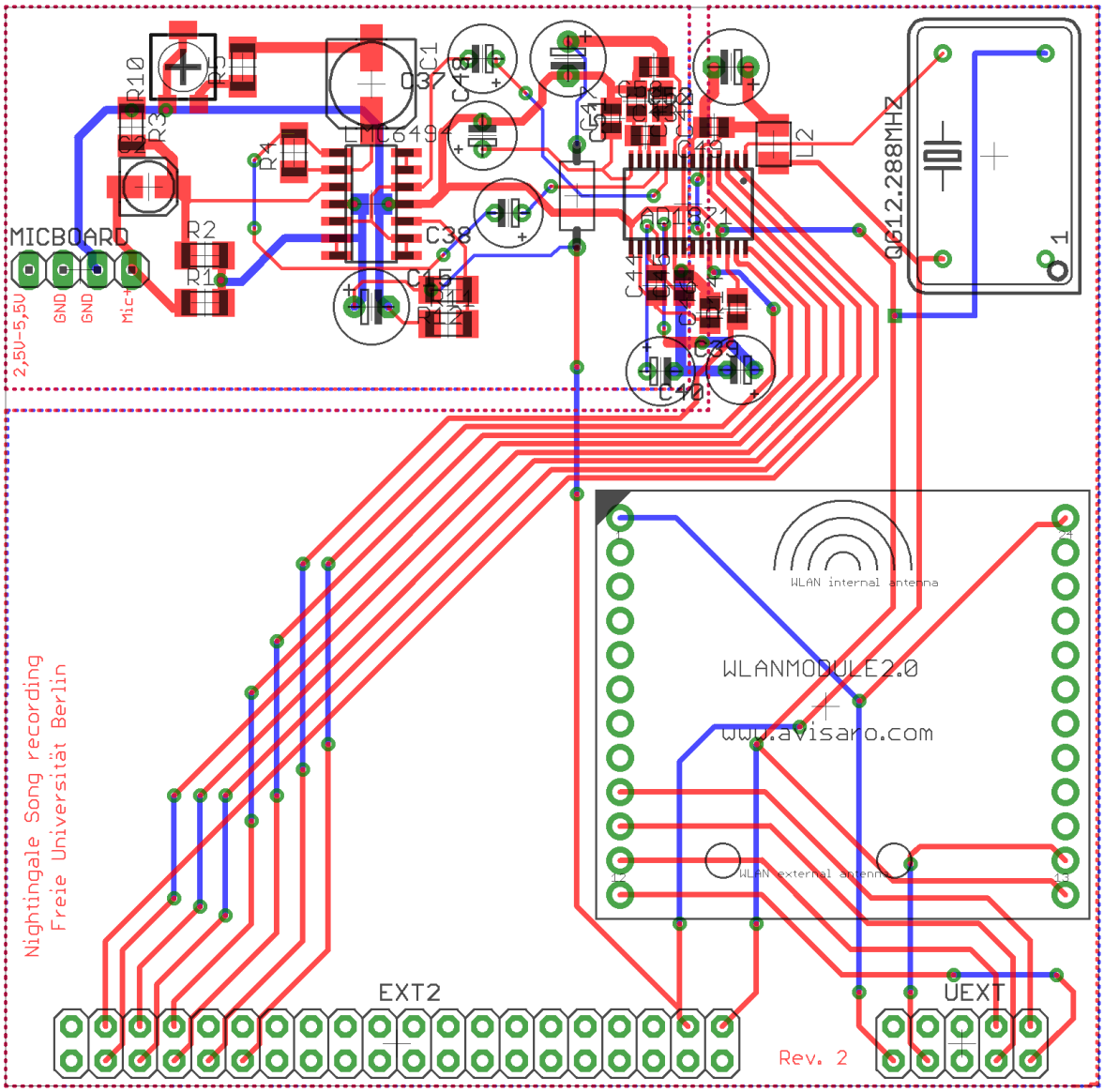


Abbildung 2.10: Gegenüber dem Prototyp fehlerbereinigte zweite Version der Erweiterungsplatine

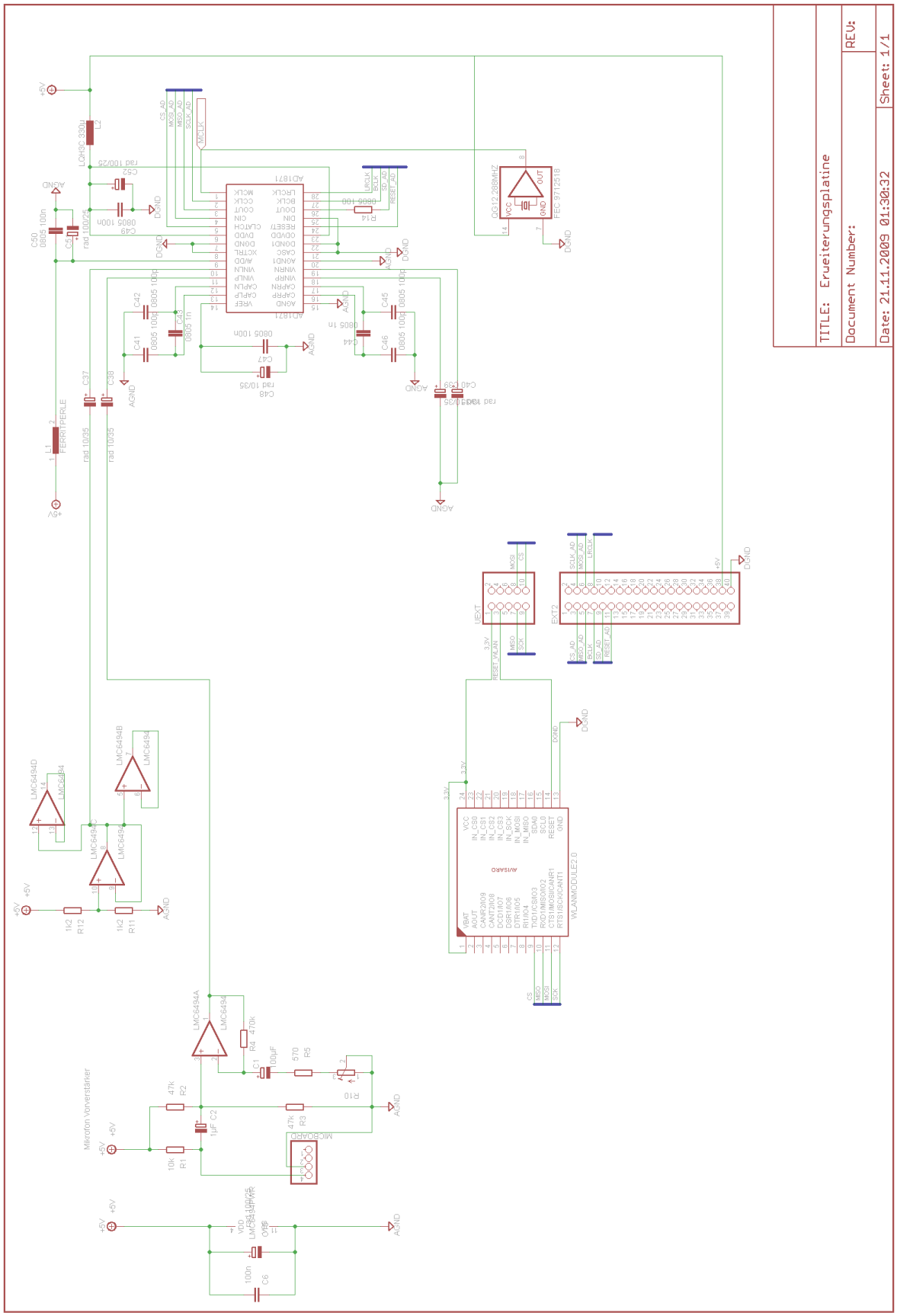


Abbildung 2.11: Zugehöriger vollständiger Schaltplan, basierend auf dem Projekt der TU Berlin [ELE41], ergänzt um das WLAN-Modul, Konnektoren und veränderter Vorverstärkerschaltung.

3. Entwurf und Implementierung

Die Beispielimplementierung des Konzepts besteht aus zwei Teilen:

1. Entwicklung von Firmware und Software für die eingebettete Hardwareplattform
2. Anbindung an die Basisstationen durch Ergänzung eines weiteren Service

3.1. Embedded Platform

Nach der erfolgreichen Zusammenstellung geeigneter Hardwarekomponenten, muss für die Basisplattform nun eine angepasste Firmware entwickelt werden. Diese übernimmt die Low-Level Ansteuerung der Hardware der Basisplattform als auch der externen Komponenten und fasst sie logisch zu Funktionen zusammen. Zur eigentlichen Nutzung durch die Software bzw. Betriebssystem, stellt die Firmware in der höchsten Schicht diese in Form einer API bereit. Aus ihnen werden die Komponenten der zeitgesteuerten Aufnahme und Speicherung auf die integrierte Speicherkarte, Verwaltung des Schedulers (3.1.8.1.), Synchronisation der Zeitpläne mit den Basisstationen, sowie die Übermittlung bereits aufgezeichneter Daten umgesetzt. Um mit der Basisstation Daten austauschen zu können, muss ein Protokoll für eine fehlertolerante Kommunikation entworfen und umgesetzt werden.

3.1.1. Entwicklungsumgebung

Mehrere Hersteller bieten inzwischen speziell angepasste Entwicklungsumgebungen für die ARM Mikrocontrollerprogrammierung, z.B.: *Embedded Workbench* von IAR, *µVision* von Keil oder *Ride7* von Raisonance, um nur eine Auswahl aus den Bekanntesten zu nennen. Die Preise bewegen sich dabei im Bereich von mehreren Tausend Euro, möchte man auf Restriktionen der Codegröße beim Compilieren und Debuggen verzichten. Eine Ausnahme stellt Ride7 dar, welches eine zweistellige Gebühr verlangt – allerdings für jedes verwendete Entwicklergerät welches von der Beschränkung befreit werden soll, wie beispielsweise dem STM32 PRIMER2.

Eine kostenlose Alternative stellt das Yagarto Paket von www.yagarto.de dar, welche für die Abkürzung „Yet another GNU ARM toolchain“ steht und das Ansinnen bereits sehr gut beschreibt. Insgesamt sind drei Komponenten für die Entwicklung notwendig:

- OpenOCD (Open On-Chip Debugger)

  Bietet die Unterstützung verschiedener JTAGs, wie dem verwendeten ARM-USB-OCD von Olimex.

- Yagarto toolchain

  Enthält die Binutils (make, sh...), newlib, GCC und GDB

- Eclipse IDE für C/C++ mit dem Zylin plugin für Embedded Debugging

  Die eigentliche Entwicklungsumgebung (Abbildung 3.1).

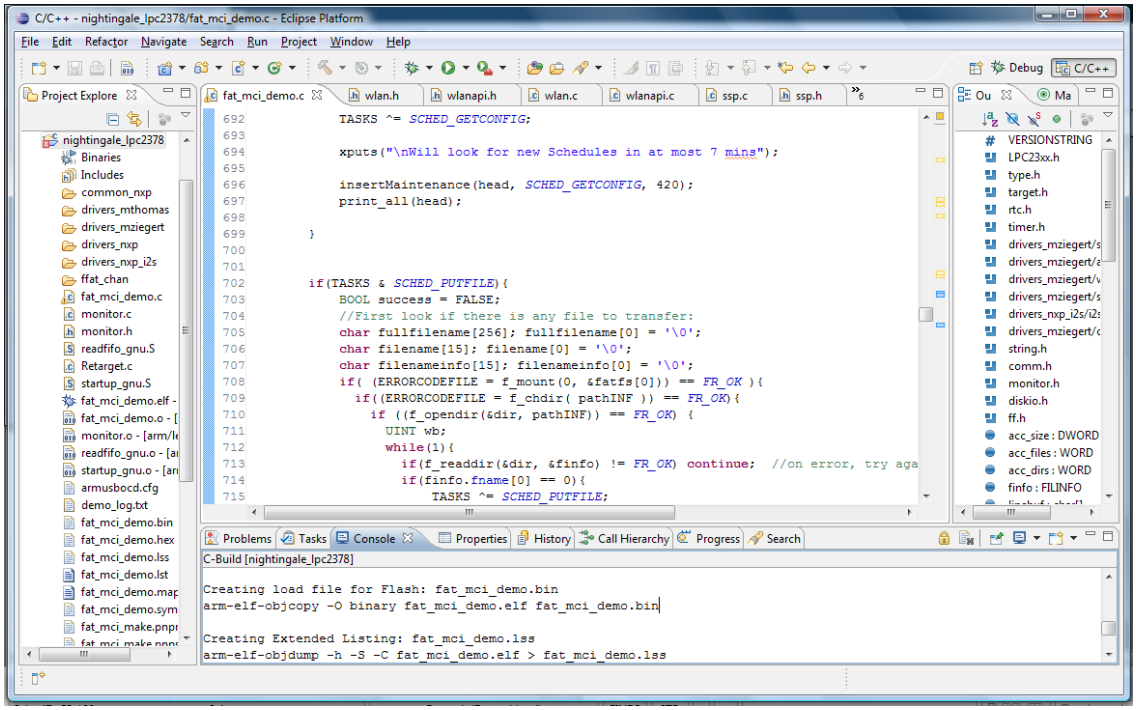


Abbildung 3.1: Eclipse IDE für C/C++ Entwickler mit Zylin Embedded CDT plugin

Die wichtigsten Schritte der Einrichtung der Entwicklungsumgebung werden im „Howto“ der Webseite von Yagarto beschrieben. Abweichungen bei der Konfiguration von Open-OCD und der Initialisierung des Olimex ARM-USB-OCD JTAG Dongle können dem Projektverzeichnis entnommen werden.

Über das Makefile des Projekts kann die zu verwendende Toolchain, mit einzubeziehende Source-Dateien und ggf. Optimierungsoptionen während des Compilierens festgelegt werden. Der verwendete ARM7TDMI-S bietet außerdem die Unterstützung von Thumb, einer Möglichkeit die Codedichte zu erhöhen. Bedingt durch den 32-Bit Befehlssatz können Programme unnötig groß werden, aus diesem Grund werden die häufigsten Befehle auf 16-Bit abgebildet, um eine kleinere Programmgröße zu erhalten (vgl. Kap. 2.2.2.). Diese werden bei der Ausführung dynamisch dekomprimiert, so dass im unoptimierten Fall von einer geringeren Leistung von bis zu 33% auszugehen ist. Aufgrund der ROM-Größe von 512kB kann in unserem Fall auf Thumb verzichtet werden, so dass durchgehend der normale ARM-Befehlssatz verwendet wird.

3.1.2. Entwurf des Programmablaufs

Für den Entwurf der Ablaufbedingungen müssen wir uns zunächst einen Überblick über die Aufgaben innerhalb des gesamten Programmzyklus verschaffen. Hierzu sind im einzelnen nötig:

1. Die **Aufnahme** – diese beinhaltet das Empfangen von Tondaten vom externen AD-Wandler und deren Speicherung auf SD-Karte.
2. Die **Zeitplansynchronisation** – erfolgt durch die Anforderung für die Mikrofonstation relevanter Aufnahmezeiten von einer erreichbaren Basisstation. Hierbei wird ebenfalls die Systemzeit abgeglichen.
3. Die **Aufnahmeversendung** – findet bei Vorhandensein mindestens einer beendeten Aufnahme statt und erfolgt durch die Übertragung der Datei an eine erreichbare Basisstation.

Diese drei Aufgaben können als die relevanten Zustände unserer Anwendung gesehen werden und wurden bereits nach ihrer absteigenden Priorität aufgeführt. Eine Aufnahme hat somit immer Priorität vor der Zeitplansynchronisation oder der Versendung einer Aufnahme. Alle Zustände werden nach Verlassen zurück in den selben Grundzustand überführt:

4. Der **Grundzustand** – ist der Start- und Endpunkt jeder Aufgabe. In ihm sollte keinerlei Funktion stattfinden.

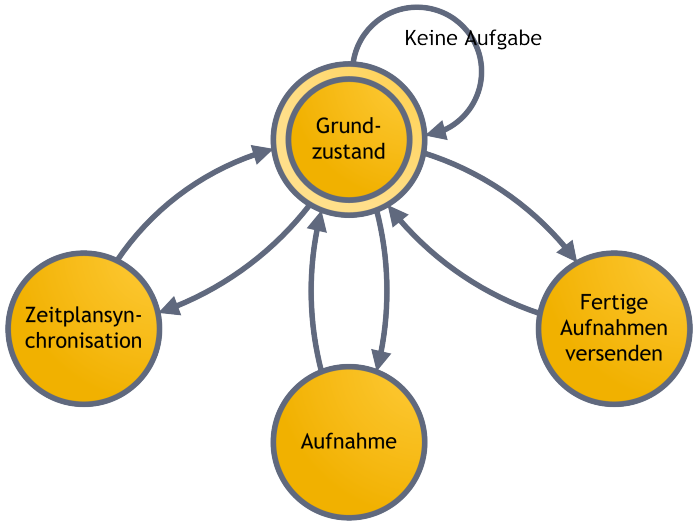


Abbildung 3.2: Vereinfachtes Zustandsdiagramm des Programmablaufs

Abbildung 3.2 stellt das daraus resultierende Zustandsdiagramm dar. Durch die geringe Anzahl an Zuständen bot es sich an – auch um den Zeitrahmen des Projekts nicht zu sprengen – auf ein Betriebssystem welches mehrere Prozesse gleichzeitig verwalten kann zu

verzichten und einen seriellen Programmablauf zu wählen. Sollten also mehrere Aufgaben zur selben Zeit anstehen, so erfolgt die Abarbeitung nacheinander. Diese Entscheidung bringt wichtige Vorteile mit sich:

- Die Anzahl an Seiteneffekten beim Zugriff auf gemeinsame Ressourcen wird in einem überschaubaren Rahmen gehalten, was den Aufwand der Umsetzung verringert und die Zuverlässigkeit des Systems erhöht.
- Entgegen einer quasi parallelen Ausführung verschiedener Aufgaben, bleibt der RAM-Bedarf bei einer seriellen Ausführung fest kalkulierbar (in sofern keine dynamischen Arrays angelegt werden). Wie bereits betont, ist unser Anwendungsfall aufgrund des sehr hohen Datenaufkommens gegenüber typischen Sensornetzprojekten eher unüblich. Daher ist es besonders wichtig, dass möglichst viel RAM zur Pufferung genutzt werden kann, um zu verhindern, dass der Datenstrom zu jedweder Peripherie aufgrund von kurzzeitigen Engpässen (z.B. Sektorwechsel im Dateisystem) abreißt.

Nachteilig ist hingegen, dass eine manuelle Priorisierung von Aufgaben umgesetzt werden muss, um die zeitnahe Ausführung von Aufnahmen zu garantieren. Anderenfalls könnte zum Beispiel das Versenden einer fertiggestellten Datei den Beginn einer nachfolgenden Aufnahme deutlich verzögern – wenn nicht sogar komplett verhindern. Dies kann auch indirekt über eine wartende Zeitplansynchronisation geschehen, indem ein noch nicht bekannter Aufnahmezeitraum bereits bei der Basisstation zur Abholung bereit steht, aber von der Mikrofonstation durch die blockierende Wirkung ihrer lang andauernden derzeitigen Übertragung nicht abgefragt werden kann. Die Zeitplansynchronisation selbst ist hingegen bezüglich dem Beginn einer Aufnahme als unkritisch anzusehen, da davon auszugehen ist, dass der Abgleich eines Zeitplans nur wenige Sekunden in Anspruch nimmt. Eine Verzögerung in diesem Rahmen ist als vertretbar anzusehen. Eine Ausnahmebehandlung ist hierdurch nicht notwendig.

Somit können wir schlussfolgern, dass bei einem seriellen Programmablauf, das Versenden von Aufnahmen als einzige Aufgabe durch höherpriorisierte Aufnahme- oder Zeitplansynchronisationsaufgaben unterbrechbar sein muss. Dies stellt keine besondere Herausforderung an unseren Entwurf dar, denn da jedwede Kontaktaufnahme und Datenübertragung mit einer Basisstation drahtlos erfolgt, ist hier ohnehin eine Fehlertoleranz einzuplanen. Das Unterbrechen einer Dateiübertragung muss somit - neben dem Verlust der TCP-Ver-

bindung - ebenfalls um das Auftreten höherpriorisierter Aufgaben erweitert werden. Unser Programm soll daraufhin ganz normal in den Grundzustand zurückfallen, von dem aus in jeden anderen Zustand gelangt werden kann – bevor ihm später wieder die Möglichkeit gegeben wird die Dateiübertragung fortzusetzen.

Diese Aufgaben müssen somit zu jedem Zeitpunkt ausgelöst werden können. Dies geschieht durch die einzig nebenläufige Komponente, die jeden unserer Zustände kurzzeitig unterbrechen kann: Dem **Scheduler**. Dieser stellt unsere Ablaufsteuerung dar, indem er uns zu vorher festgelegten Zeitpunkten über anstehende Aufgaben informiert. Die Zeitpunkte können sowohl über die Zeitsynchronisation programmiert worden sein, also auch innerhalb des Programmablaufs der einzelnen Zustände (z.B. Basisstation war nicht erreichbar zur Dateiübertragung, versuche es erneut in 7 Minuten).

3.1.3. Aufbau und Architektur der Anwendung

Unsere klar getrennten Aufgaben und die dazugehörigen Zustände erlauben uns die Entwicklung der Anwendung sinnvoll aufzuteilen. Möchte man sich außerdem die Möglichkeit offen halten, später doch noch auf ein Betriebssystem zurückgreifen zu können, ist es empfehlenswert Module entsprechend der zu nutzenden Hardwareressource aufzubauen. So wird es später möglich konkurrierenden Prozessen Zugriff auf die selbe Hardware zu geben, indem für sie im einfachsten Fall nur Zugriff auf die höchste Schicht besteht – in dieser können später auch notwendige Semaphoren integriert werden, um den Kommunikationsablauf bei parallelen Zugriffen nicht zu stören.

Entsprechend unseres seriellen Programmablaufs können wir allerdings vorerst einen konkurrierenden Zugriff mittels Semaphoren vernachlässigen und nutzen die Eigenschaft der einfachen Wartbarkeit von Modulen und ihre Möglichkeit sie vollständig oder in Teilen austauschen zu können, sollten externe Hardware oder Schnittstellenbeschreibungen wechseln.

Abbildung 3.3 zeigt die gewählte Unterteilung der implementierten Module im Gesamtzusammenhang der Software als Schichtenmodell. Der Zugriff auf das Dateisystem (grün) erfolgt durch eine nahezu unveränderte Implementierung von Martin Thomas [MT09], welche auf dem FatFs-Modul von ChaN basiert [ChaN09] (vgl. Kap. 3.1.5. Dateisystem) und ebenfalls ein Demoprogramm enthält, auf dem die spätere Anwendung aufgebaut wurde. NXP Semiconductor stellt Codebeispiele für den Low-Level Zugriff auf Schnittstellen bereit, welche für unseren Anwendungsfall angepasst wurden.

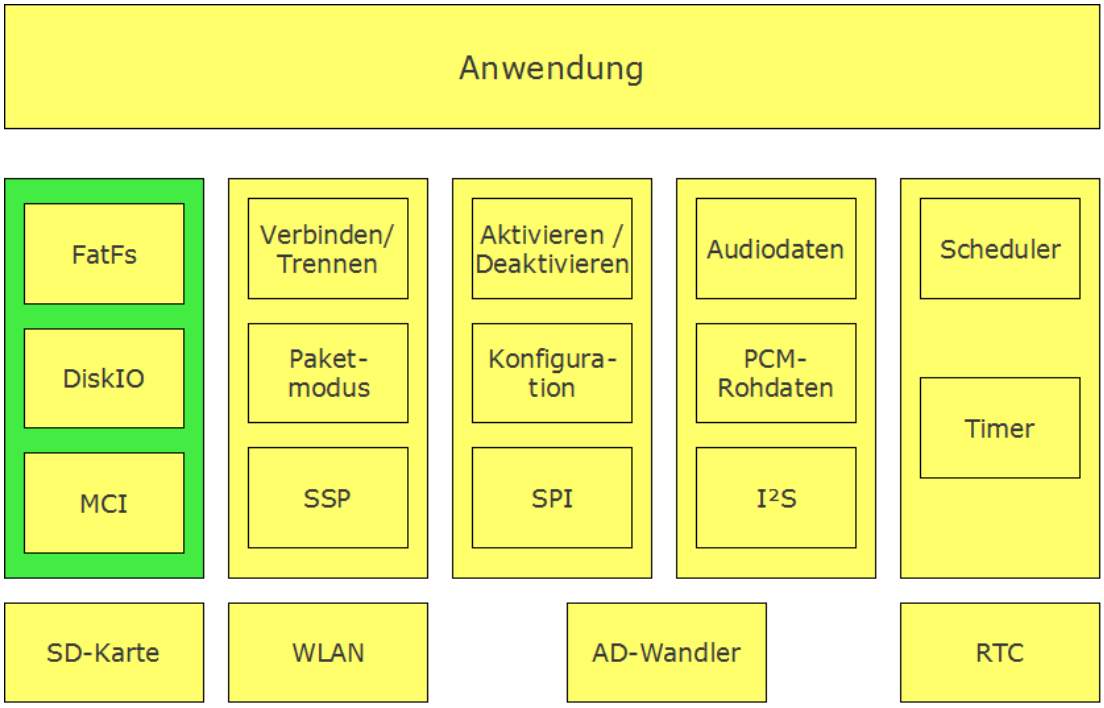


Abbildung 3.3: Darstellung des gewählten Schichtenmodells

3.1.4. LPC2378

Wie in Kapitel 2.2.2. bereits dargestellt, besteht ein ARM Mikrocontroller nicht nur aus dem eigentlichen Prozessorkern, sondern aus einem Bussystem, Arbeitsspeicher und Peripherie (Abbildung 3.4). Da die Festlegung der Taktraten äußerst flexibel ist, soll für die spätere Weiterentwicklung ein kurzer Überblick über die verwendeten Einstellungen gegeben werden.

Als externer Taktgeber steht ein 12 MHz Quarz zur Verfügung. Durch ihn können Frequenzen generiert werden, die zum Beispiel Taktraten für den USB-Betrieb (48 MHz) ermöglichen. Hierzu wird ein PLL von 288 MHz erzeugt, aus dem der Takt des Prozessors und der Peripherie abgeleitet wird. Besondere Rücksicht muss auf die Angaben im Errata Sheet [NXPERR09] genommen werden. So handelt es sich bei der von Olimex verwendeten Version ders NXP LPC2378 um eine sogenannte Null-Serie. Dies ist die erste verfügbare Revision des Mikrocontrollers (derzeit gibt es 0, A,B und D), so dass das gesamte Errata auf diesen anwendbar war. So ist hier die PLL auf 290 MHz begrenzt, I²S ist nicht DMA-fähig und der Prozessortakt sollte für einen fehlerfreien Betrieb nicht 60 MHz überschreiten.

Die gewählten Einstellungen in Tabelle 3.1 sollten für eine spätere Integration von Wavepack genügend Rechenleistung bieten.

Komponente	Takt
Ext. Quarz	12 MHz
PLL	288 MHz
APB	24 MHz
Prozessor	48 MHz
MCI (Speicherkarte)	24 Mhz
SPI/SSP (WLAN-Modul)	6MHz
SPI (AD-Wandler)	12 kHz
I²S	extern

Tabelle 3.1: Gewählte Einstellungen des Mikrocontrollers und der Peripherie

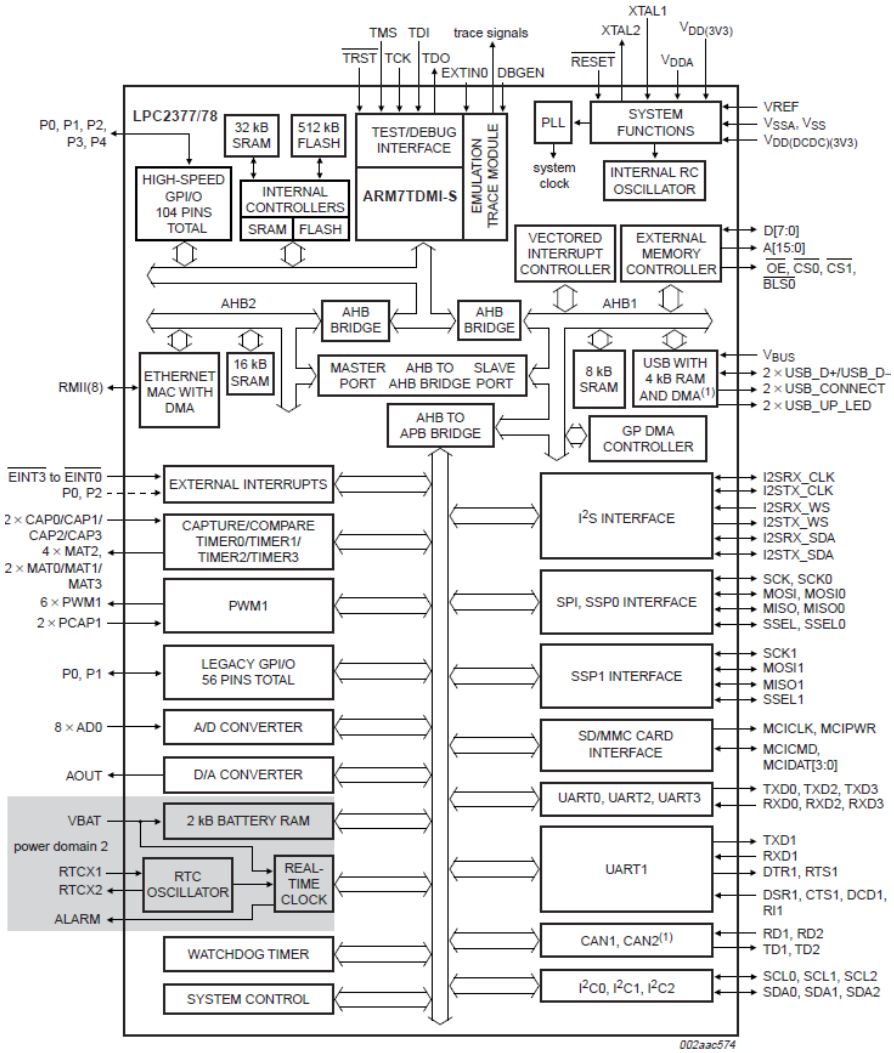


Abbildung 3.4: Aufbau des NXP LPC2378 inkl. Der unterschiedlichen Bussysteme [LPC78]

3.1.4.1. Energieeinsparung

Die ARM7 Mikrocontroller sind in der Vielfalt der Einsparmöglichkeiten dem langsameren MSP430 sehr ähnlich. Sie lassen sich in drei Bereiche einteilen (vgl. [HW08] Kap. 2.4, und [LPC09] Kap. 8):

1. Nutzung der integrierten Energiesparmodi
2. Taktgeschwindigkeit, sowohl des eigentlichen Prozessorkerns als auch des Systembus
3. Deaktivieren von ungenutzten On-Chip Komponenten

Energiesparmodi

- Normalbetrieb

Der Prozessorkern, der Systembus und somit alle gewünschten Peripheriekomponenten sind aktiv. Dies ist der normale Zustand wenn Berechnungen durchgeführt werden und repräsentiert dementsprechend den höchsten Energieverbrauch. Unter Vernachlässigung zusätzlich aktivierter Peripherie beträgt dieser 15mA bei 10Mhz und 63mA bei der maximalen Taktrate von 72Mhz, der Systemtakt sei hier gleich dem Prozessortakt. [LPC78]

- Leerlaufmodus (Idle)

Im Leerlaufmodus wird lediglich der Prozessorkern des ARM7 angehalten, der Takt des Systembus und jede von ihm abhängige Peripherie bleiben aktiv und können dazu genutzt werden den Prozessorkern durch von ihnen generierte Interrupts wieder aufzuwecken. Dieser Modus ist somit besonders geeignet, wenn auf externe Eingaben gewartet wird. Das Einsparpotenzial ist allerdings eher moderat.

- Schlafmodus (Sleep mode)

Neben dem Prozessorkern wird nun auch der Systembus von der Taktquelle getrennt. Dadurch werden alle von ihm abhängigen Peripherien angehalten und Systemtakt-Teiler werden zurückgesetzt. Soll der vorige Zustand wieder hergestellt werden, muss eine Aufwachsequenz verwendet werden sobald der Betrieb durch vom Systemtakt unabhängige Interrupt generierende Peripherie wie die RTC (Real Time Clock) oder externe Interrupts aufgeweckt und wieder fortgesetzt wird. Die Inhalte des RAM bleiben dabei erhalten.

- Tiefschlafmodus (Power-down mode)

Zusätzlich zu allen Stromsparfunktionen des Schlafmodus wird nun auch die (interne) Taktquelle selbst angehalten. Ebenfalls wird der Flash-Speicher von der Stromversorgung getrennt. Der Energieverbrauch beträgt im Idealfall noch 150µA, also

ein Hundertstel des Normalbetriebs bei 10Mhz. [LPC78] Da sich jede Taktquelle zunächst stabilisieren muss, ist allerdings die Startzeit bis wieder mit Rechenoperationen begonnen werden kann gegenüber dem Schlafmodus länger.

Zieht man den tieferen Vergleich zum MSP430, sollte demnach der normale Betriebsmodus in dem sich ein batteriebetriebener Sensorknoten mit ARM7 Mikrocontroller zwischen 99,9% und 99,999% der Zeit befinden sollte, einer der beiden Schlafmodi sein. (vgl. [LB04] Kap. 4.8) Durch die Deaktivierung des System- und des Peripherietaktes ist dies allerdings während einer Aufgabe nicht praktikabel. Aus diesem Grund ist lediglich der Grundzustand praktikabel für einen Stromsparmodus und wurde beispielhaft durch den Leerlaufmodus umgesetzt. Dieser setzt ein, wenn der Scheduler signalisiert, dass derzeit keine Aufgabe ansteht (TASKS-Register ist leer). Ein tieferer Schlafmodus sollte in einer späteren Weiterentwicklung folgen. Durch die Neuinitialisierung der Peripherie wurde zunächst zugunsten eines fehlerfreien Betriebs auf den Schlaf- und den Tiefschlafmodus verzichtet.

Taktgeschwindigkeit

Wenig überraschend ist der Takt des Prozessor direkt proportional zur Energieaufnahme. Hier wurde deshalb auf 48MHz begrenzt – Leistung für eine spätere Kompression mit Wavepack sollte somit vorhanden sein. Weitere Einsparungen können allerdings auch durch einen größeren Teiler bei der Festlegung des Systembus-Taktes erzielt werden. Hier sollte vor Allem bezüglich der anzubindenden Peripherie und des Datendurchsatzes Leistung gegenüber erzielbaren Einsparungen abgewogen werden.

On-Chip Peripherie

Nicht genutzte Peripherie sollte deaktiviert werden. Gerade für Anwendungen mit erforderlicher langer Laufzeit wird dies gern vergessen. Zu beachten ist außerdem, dass der ARM-Mikrocontroller gegenüber dem MSP430 interne Pull-up Widerstände besitzt, welche möglicherweise deaktiviert werden sollten.

3.1.5. Dateisystem

Für die Unterstützung des Fat16/32 Dateisystems wurde auf das fertige FatFS Modul von Martin Thomas zurückgegriffen. [MT09] Dieses nutzt die MCI-Schnittstelle für eine schnelle Datenübertragung zur SD-Karte. Im Gegensatz zur SPI-Schnittstelle können hier

4 Bits pro Takt übertragen werden bei bis zu 25 MHz und somit Datenraten im einstelligen Megabyte Bereich erreicht werden.

FatFS unterstützt das Mounten des Dateisystems, die Verwaltung mehrerer geöffneter Dateien, Schreib- und Lesevorgänge und lange Dateinamen (worauf aus Lizenzgründen allerdings absichtlich verzichtet und auf die altbekannte 8.3 Namenskonvention gesetzt wurde). Da dieses Modul in sich abgeschlossen und von ChaN [ChaN09] ständig mit Updates versorgt wird, soll hier auf eine nähere Darstellung des Aufbaus verzichtet werden. Als Hinweis soll allerdings nicht unerwähnt bleiben, dass in der Implementierung des Lesens von Speicherbereichen größer als 1023 Byte ein Fehler befand, der zum Überschreiben der ersten 512 Byte geführt hat. Dieser wurde im Rahmen dieser Diplomarbeit entdeckt und ist inzwischen in das Projekt von Martin Thomas mit eingeflossen.

3.1.6. AD-Wandler

Im Allgemeinen ist die Ansteuerung eines AD-Wandlers nicht sonderlich komplex. Die Besonderheit des AD1871 ist allerdings die Nutzung zweier Schnittstellen. SPI wird für die Konfiguration der Einstellungen verwendet, während I²S ausschließlich für die Übertragung von Audiodaten vorgesehen ist.

3.1.6.1. Ansteuerung und Konfiguration

Grundsätzlich können über Control Register die Standardeinstellungen des AD-Wandlers verändert werden. Hierzu zählt. u.a. die Nutzung von 16-Bit, die Stummschaltung eines Kanals oder die Aktivierung des Energiesparmodus. In Abbildung 3.5 ist das übliche Vorgehen aufgezeigt: 16-Bit werden in das SPI-Register geschrieben. Während die vier höherwertigen Bits die Adresse des Registers beinhalten, gibt Bit 11 bei einem HIGH einen Schreibvorgang an. Die Antwort kann (sofern es ein Lesevorgang war) sofort ausgelesen werden. Auf die Nutzung von SPI soll hier nur kurz eingegangen werden, eine genaue Beschreibung von SPI befindet sich in Kapitel 3.1.6.1. Ansteuerung des WLAN-Moduls.

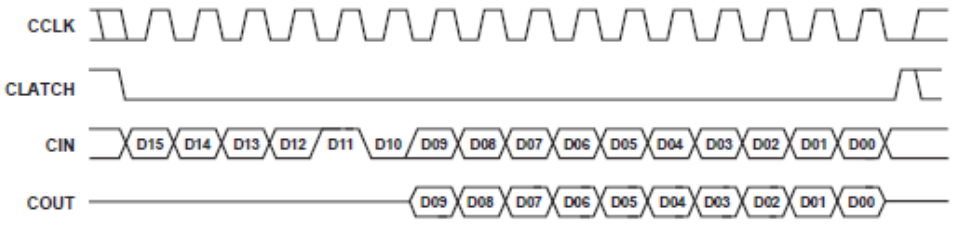


Abbildung 3.5: Schreiben eines Registers: Bits D15 bis D12 enthalten die Adresse, während D11 einen Schreibvorgang signalisiert. [AD02]

Die eigentliche Übertragung der Tondaten erfolgt nun über die I²S (Inter-IC-Sound) Schnittstelle. Das Taktsignal BCLK ist ein Teiler des externen 12,288 MHz Quarzes und wird somit direkt vom AD-Wandler vorgegeben. Der ARM-Mikrocontroller agiert hier somit als Slave. Datenworte werden durch LRCLK voneinander getrennt, deren Länge wie bereits bekannt 16-Bit betragen. (Abbildung 3.6)

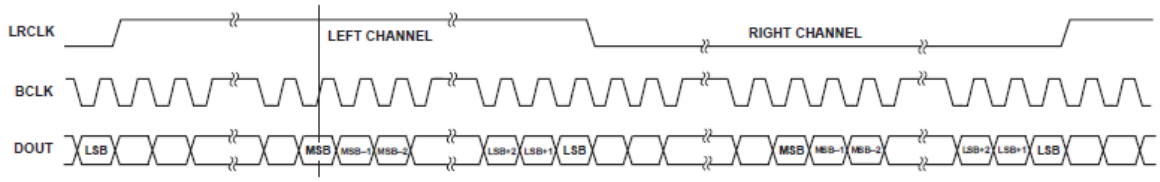


Abbildung 3.6: I²S Audioübertragung von Stereo PCM-Daten. [AD02] Bei der Nutzung von Mono trennt LRCLK lediglich die Datenworte.

Empfangene Audiodaten werden von der Interruptroutine aus dem FIFO-Puffer ausgelesen und in einen global sichtbaren Ringpuffer geschrieben. Dieser wird später durch die Anwendung ausgelesen.

3.1.6.2. Energieeinsparung

Der AD1871 besitzt sowohl die Möglichkeit den analogen als auch den digitalen Teil des Bausteins in den Schlafmodus zu versetzen. Hierzu muss Bit 7 in Control Register 1 auf HIGH gesetzt und anschließend Reset dauerhaft auf LOW gezogen werden.

3.1.7. Funkmodul

Für das Avisaro WLAN Modul 2.0 ist nach der rein physikalischen Anbindung, die Implementierung der System-Bus Kommunikation und darauf aufbauend auch die Umsetzung der bereitgestellten API des Moduls nötig. Über diese API kann u.a. die Kommunikation überwacht, Änderungen an den Einstellungen vorgenommen oder das gesamte Modul in den Stromsparmodus versetzt werden. Dieser Modus ist der Grundzustand des Funkmoduls. Dies ist ohnehin sinnvoll, da von außen keine Verbindungsversuche erwartet werden sollen: Alle nötigen TCP-Verbindungen werden von der Mikrofonstation selbst initiiert. Die Basisstationen müssen somit keine aufwändige Verwaltung aller umliegenden Mikrofonstationen führen oder diese gar regelmäßig aktualisieren. Die Mikrofonstation kann somit nicht nur völlig flexibel entscheiden mit welcher Basisstation sie sich verbindet, sie kann auch in allen anderen Zeiträumen das Funkmodul deaktivieren um Energie zu sparen, da sie nicht empfangsbereit für eingehende Verbindungen sein muss.

3.1.7.1. Schichtenmodell

Für eine klare Strukturierung, wird die Implementierung der WLAN-Kommunikation wie in Abbildung 3.7 dargestellt in drei Schichten unterteilt:

- Die **Low-Level Ansteuerung** – beinhaltet die Ansteuerung, Konfiguration und dem Byte-weisen Senden und Empfangen über die SSP/SPI-Schnittstelle des ARM Mikrocontrollers zum WLAN-Modul. Kapitel 3.1.7.3.
- Die **Kommunikationsschicht** – baut auf der Low-Level Ansteuerung auf und setzt die vorgegebene API des WLAN-Modul Herstellers um. Diese umfasst sowohl Befehle im Paket- als auch Textmodus und gibt die Möglichkeit zur Konfiguration, Statusabfrage und Datenübertragung an einen verbundenen Kommunikationspartner (in unserem Fall eine Basisstation). Kapitel 3.1.7.4.
- Die **Benutzer-API** – fasst Befehle der Kommunikationsschicht logisch zusammen, um dem Programmierer bequem Zugriff auf die häufigsten Anwendungsfälle zu geben, ohne dass sich dieser beispielsweise um eine Fehlerbehandlung und Überwachung des Verbindungsauf- und abbaus, als auch die Datenübertragung selbst kümmern muss. Kapitel 3.1.7.5.

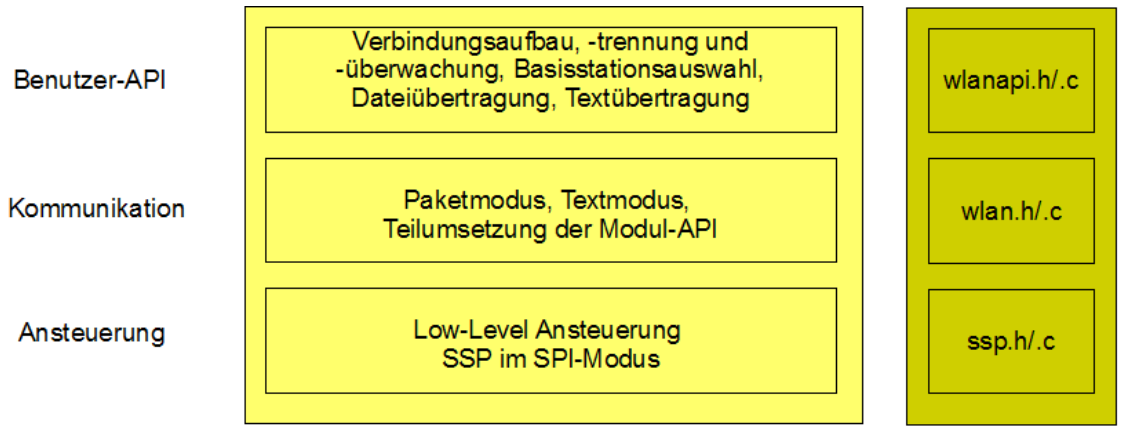


Abbildung 3.7: Schichtenmodell der WLAN-Komponente

3.1.7.2. Konfiguration

Unabhängig von der Implementierung muss ein WLAN-Modul vom Auslieferungszustand durch eine Erstkonfiguration in einen für uns ansprechbaren Zustand versetzt werden. Dies erfolgt über ein übersichtliches Webinterface. Alle Einstellungen die direkt nach dem Einschalten des Funkmoduls gelten sollen, können hier vorgenommen werden. Dies betrifft die Wahl der Schnittstelle SPI, die Einstellungen zu CPOL und CPHA, als auch die Wahl der Verschlüsselungstechnik, des Schlüssels (welcher Hex-codiert eingegeben werden

muss) und der IP-Adresse. Bereitgestellte Firmware-Updates können über ein Untermenü eingespielt werden.

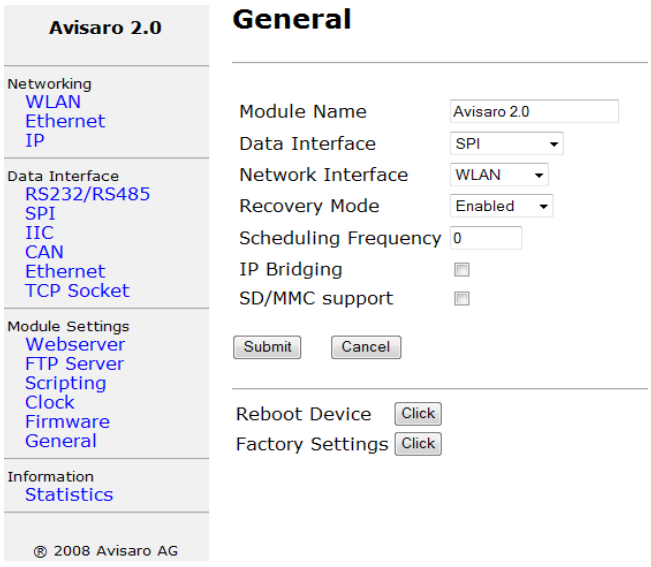


Abbildung 3.8: Die Erstkonfiguration des WLAN-Moduls ist über ein Webinterface möglich.

Sollten hier falsche Werte verwendet worden sein, können alle Einstellungen auch über den Text und Paketmodus geändert werden (siehe Kapitel Kommunikation: Paket/Text) wenn zumindest die SPI-Schnittstelle korrekt ausgewählt und konfiguriert wurde. Eine Übersicht aller Werte ist in Tabelle 1 im verfassten Anwenderhandbuch im Anhang zu finden. Die Geschwindigkeit betreffend ist der wichtigste Punkt die „Scheduling Frequency“. Der Hersteller nutzt für die Umsetzung von Threading eine RTOS Variante und bestimmt mit dieser Variable die Größe der Zeitslots in Millisekunden die jedem Prozess maximal zustehen, bevor ihm alle Ressourcen entzogen und dem nächsten Prozess übergeben werden. Dabei ist es egal ob dieser Prozess wirklich Rechenzeit benötigt oder nicht. Welches Scheduling-Verfahren genau verwendet wird, zum Beispiel nach Round Robin welches den Zeitslot auch vorzeitig freigeben kann, ist nicht bekannt. Allein der unnötige Kontextwechsel könnte allerdings ausreichend dafür sein, dass bei ersten Versuchen nur sehr geringe Geschwindigkeiten über eine TCP/IP-Verbindung erreicht werden konnten. Die Standardeinstellung verschwendet somit Rechenleistung, die für den TCP/IP-Netzwerkverkehr benötigt wird und sollte auf 0 ms geändert werden, wodurch RTOS von nun an die Größe und Verteilung der Zeitslots dynamisch anpasst. Dies brachte bei Tests einen Geschwindigkeitsgewinn von 400%.

Als Kompromiss muss die Wahl der WLAN-Verschlüsselung gesehen werden: Da die WLAN-Karten der Basisstationen lediglich WEP im Ad-hoc Modus unterstützen, muss auf

das sichere WPA2-Verfahren des Avisaro Moduls verzichtet werden. Der WEP Schlüssel wird ebenfalls im Anhang veröffentlicht, da der zeitliche Aufwand deutlich höher ist ihn in diesem Dokument nachzuschlagen als ihn mit einem entsprechenden Tool zu „knacken“.

3.1.7.3. Ansteuerung

Wie bereits aus dem Platinenlayout ersichtlich, wird die – nach MCI – Schnellste der möglichen externen Schnittstellen für die Kommunikation zwischen der LPC2378 basierten Entwicklerplatine und dem Funkmodul verwendet. SPI (Serial Port Interface) basiert auf 4 Signalleitungen: MOSI (Master Out, Slave In), MISO (Master In, Slave Out), SCK (Serial Clock) und SSEL (Slave Select). Die Namensgebung von Master und Slave weist bereits darauf hin, dass es sich um ein hierarchie-basiertes Bus-System handelt. Es können theoretisch beliebig viele Slaves an einem Bus teilnehmen, die allerdings lediglich von maximal einem Master über mehrere SSEL-Verbindungen verwaltet werden können. Die Aufgabe des Masters übernimmt in unserem System der ARM LPC2378 und wird somit auch das Taktsignal SCK vorgeben. Setzt dieser eine SSEL-Verbindung auf LOW, wird der damit verbundene Slave mit Hilfe des Taktsignals die seriell übertragenen Bits auswerten und ist ansonsten bei SSEL=HIGH an den Datenein- und ausgängen hochohmig. Für die Auswertung nach Vorgabe des Taktsignals haben sich 4 Modi durchgesetzt die sich aus der Permutation von CPOL (Clock Polarity) und CPHA (Clock Phase) ergeben. CPOL bestimmt ob der Grundzustand des Taktsignals, wenn keine Übertragung aktiv ist (SSEL=HIGH), HIGH oder LOW ist. CPHA, ob ein Bit an der ersten oder zweiten Flanke auf den eigentlichen Datenleitungen MISO und MOSI übernommen wird. Während der Master somit über MOSI ein Byte an den Slave sendet, empfängt er vom Slave die selbe Anzahl an Bits über MISO. [LPC09] In Abbildung 3.9 ist das zeitliche Zusammenspiel abhängig der verschiedenen Modi übersichtlich dargestellt. In unserem Fall unterstützen sowohl Master als auch Slave alle 4 möglichen Modi. Es hat sich herausgestellt, dass auf dem Avisaro WLAN-Modul 2.0 ebenfalls ein ARM Mikrocontroller eingesetzt wird, ein NXP LPC2366. Nach Rücksprache mit Avisaro über mögliche Geschwindigkeitsoptimierungen stellte sich heraus, dass das Modul erst nach mindestens 4 empfangenen Bytes oder einem Timeout eine Weiterverarbeitung ausführt. Dieses Verhalten weist auf die Verwendung von SSP (Synchronous Serial Port) hin, welches zwar vollständig SPI kompatibel ist, im Gegensatz dazu aber mehrere Vorteile besitzt. U.a. erlaubt es eine höhere Taktung von SCK bis zur Hälfte des APB (Advanced Peripheral Bus) und bietet getrennte, jeweils 8 Werte große FIFO-Puffer für den Datenempfang- und versand. „Normales“ SPI nutzt für diese Aufgabe dagegen

lediglich ein einzelnes Register, welches – um eine Schreibkollision zu vermeiden – nach jedem Schreibvorgang ausgelesen werden sollte. Bei der Nutzung von SSP ist ein Auslesen erst dann zwingend notwendig, wenn der Empfangs-FIFO voll ist. Ein Interrupt wird allerdings bereits dann ausgelöst, wenn mindestens 4 Werte empfangen wurden oder ein Timeout auftrat.

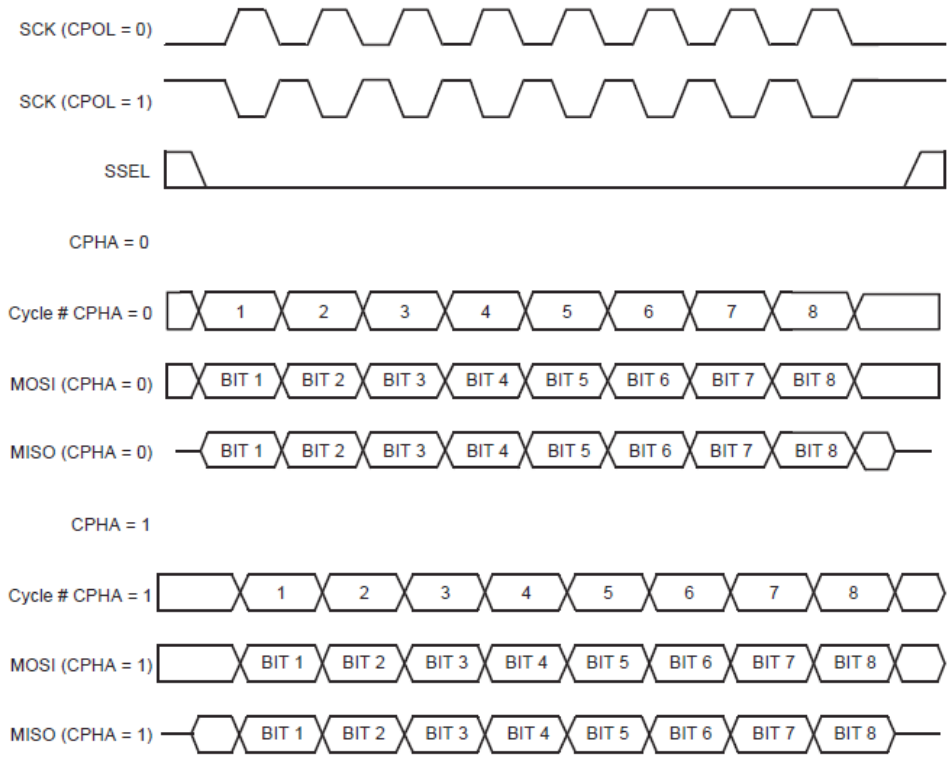


Abbildung 3.9: Darstellung der Kombinationen von CPOL und CPHA und deren Auswirkungen auf den SPI-Transfer [LPC09]

Diese Vorteile, welche zu einem deutlich höheren Datendurchsatz führen, waren entscheidend dafür, dass für die Kommunikation mit dem Funkmodul ebenfalls auf SSP im SPI-Modus zurückgegriffen wird. Hierzu wurden folgende Funktionen implementiert:

- ```
  BOOL SSP0Init( void );
  ```

  Aktivierung der Schnittstelle; Konfiguration von CPHA, CPOL, Wortgröße (8-Bit) und Taktrate.

- ```
  void SSP0Send( BYTE *buf, DWORD length );
  ```

  Senden eines Datenblocks mit vorgegebener Länge. Dabei wird die selbe Anzahl an Bytes (wenn auch teilweise zeitversetzt) wieder ausgelesen, auch wenn diese noch keine relevante Antwort auf unsere Anfrage enthalten (können). In diesem Fall werden Füll-Bytes empfangen.

- `void SSP0Receive( BYTE *buf, DWORD length );`

  Empfangen eines Datenblocks mit vorgegebener Länge. Dabei wird zuvor die selbe Anzahl an Füll-Bytes gesendet, um den Datentransfer zu initiieren.

Auch wenn SPI generell für den Vollduplex-Betrieb vorgesehen ist (und wir ihn auch so verwenden, indem wir ein Byte senden und auch auslesen), nutzen wir effektiv nur Halbduplex. Dies ist dadurch bedingt, dass das Modul aufgrund der komplexen API natürlich zuerst eine vollständige Anfrage abwarten muss, bevor eine Antwort generiert und dann gesendet werden kann. Steht noch keine Antwort zur Verfügung, wird 0xFF (typisches Byte-Stuffing) gesendet. Deshalb kann bei **`SSP0Send`** auf einen Verweis zu einem Rückgabepuffer verzichtet werden, da keine relevante Antwort zu erwarten ist, in sofern man gewisse Grundsätze einhält. Auf diese wird im nächsten Abschnitt eingegangen.

3.1.7.4. Kommunikation: Paket/Text

Nach der vollständigen Umsetzung der Low-Level Ansteuerung, muss darauf aufbauend die Syntax des Funkmoduls nachgebildet werden. Das Avisaro WLAN Modul 2.0 unterstützt zwei vom Aufbau her unterschiedliche API, die sich von den umgesetzten Funktionen allerdings nur im Kern überschneiden und hauptsächlich gegenseitig ergänzen.

Der **Textmodus** ist im Grunde eine Altlast der Vorgängervariante WLAN Modul 1.0. Diese unterstützte ausschließlich den Textmodus, welcher – aus nachvollziehbaren Beweggründen – hauptsächlich wegen der Kompatibilität übernommen wurde. Entwicklern wird somit der Umstieg durch den Erhalt früherer Bibliotheken erleichtert. Vorteilhaft ist die für den Menschen intuitive Syntax, welche eine erste Konfigurierung mittels Kommandozeile ermöglicht und dabei auch tiefergehende Änderungen im System erlaubt. Hier kann zwischen einer Statusabfrage und der Änderung von Einstellungen unterschieden werden:

- Anfragen werden generell mit der Nennung der betroffenen Komponente begonnen und mit einem Fragezeichen „?“ abgeschlossen. In seltenen Fällen wird ggf. noch ein weiteres Argument ergänzt.

  Bsp. [AVI09]: `IP? 0` → Abfrage IP-abhängiger Einstellungen (0 – WLAN / 1 – Ethernet) führt zu folgender Antwort:

```
192.168.0.74
255.255.255.0
192.168.0.1
192.168.0.1
OFF
```

```
192.168.0.88
255.255.255.0
192.168.0.1
192.168.0.1
10
```

Die oberen vier Zeilen entsprechen der im Flash gespeicherten IP-Adresse, des Subnets, des Gateways und des Nameservers. `OFF` indiziert, dass kein DHCP verwendet wird. Das Modul kann alternativ sowohl als DHCP-Client als auch DHCP-Server konfiguriert werden. Darunter sind die aktuell verwendeten IP-Einstellungen aufgeführt. Diese können sich aufgrund der Nutzung von DHCP von den Werten im Flash unterscheiden oder weil Änderungen im Flash vorgenommen wurden (Änderungen werden erst nach einem Neustart mittels `RESTART` übernommen). Die letzte Zeile zeigt den maximalen Abstand von TCP keep-alive Paketen in Sekunden an.

- Änderungen benötigen ebenfalls mindestens ein, meist mehrere Argumente. Dies würde am Beispiel der IP-Adresse folgendermaßen aussehen:

  ```
  IP LOCAL 192.168.0.50 0
  ```

  Die Antwort würde daraufhin `ERR_OK (0)` lauten wenn das Kommando akzeptiert wurde, gefolgt von einem Command-Prompt „<cr> <lf> >“. War der Befehl nicht erfolgreich, wird an Stelle von `ERR_OK (0)` einer von 39 Fehlercodes ausgegeben (plus zwei weitere bei Kommunikationsfehlern auf der SPI-Schnittstelle).

Generell müssen alle Eingaben mit einem *carriage return* und *line feed* (<cr> <lf>) beendet werden, um das Eingabeende zu signalisieren.

Der **Paketmodus** stellt die sinnvolle Weiterentwicklung des alten Textmodus dar. Durch ihn wird ein festes Paketformat festgelegt, welches nicht nur von Statusabfragen, Befehlen, als auch von den jeweiligen Antworten des Moduls eingehalten wird.

Header	Length		Payload			CRC	
Byte	MSB	LSB	Command	Parameter	Data	MSB	LSB

Abbildung 3.10: Aufbau des Paketformats, sowohl von Anfrage- als auch Antwortpaketen [AVI09]

Wie in Abbildung 3.10 ersichtlich, besteht der Paketkopf aus einem ein Byte großen Header. Er gibt an, ob es sich bei dem Paket um ein Kommando (`0x81`), einem Acknowledgement (`0x84`) oder negativen Acknowledgement (`0x85`) handelt. Das Längenfeld bezieht sich ausschließlich auf die Größe der Nutzdaten (Payload), welche wiederum das eigentliche Kommando (Verbindung öffnen/schließen, Datenübertragung, Statusabfrage, etc. …), eventuelle Parameter und zu übertragende Daten enthält. CRC16 ist optional und wird auf-

grund von (wenn auch marginalen) Geschwindigkeitseinbußen bei der Berechnung nicht verwendet. Bei Bedarf ist allerdings bereits eine entsprechende Implementierung in der crc.h/.c vorhanden.

Um den Kommunikationsablauf zu visualisieren, wurde das bekannte PING-Kommando ausgewählt [AVI09]:

Ping-Anfrage (Hex): `81 00 05 85 C0 A8 00 64 00 00`

`0x81` – Kommando Header
`0x00 0x05` – Länge der Nutzdaten
`0x85` – Kommando ID für PING
`0xC0 0xA8 0x01 0x08` – IP Adresse der Gegenstelle (192.168.1.8)
`0x00 0x00` – CRC

Eine Antwort würde dementsprechend positiv oder negativ ausfallen:

Acknowledge	Negatives Acknowledge
Modulantwort: `84 00 00 A9 41`	Modulantwort: `85 00 00 A9 41`
`0x84` – Ausführung erfolgreich	`0x85` – Ausführung nicht erfolgreich
`0x00 0x00` – Länge der Nutzdaten	`0x00 0x00` – Länge der Nutzdaten
`0xA9 0x41` – CRC	`0xA9 0x41` – CRC

Der Paketmodus stellt durch sein festes Format weitaus geringere Anforderungen an selbst implementierte Parser und wäre somit gegenüber dem Textmodus die bessere Wahl. Unglücklicherweise kommt erschwerend hinzu, dass sich wie bereits erwähnt, beide API nur marginal überschneiden. So enthält das exclusive Paketmodus-Kommando `PUTPACKET2` eine einfache Flusskontrolle, indem im Antwortpaket die verbleibende Größe des Sendepuffers angegeben wird. Ausschließlich über den Textmodus ist allerdings die Aktivierung des für den Energieverbrauch wichtigen Schlafmodus möglich. Dies führt dazu, dass letztendlich beide API unterstützt und implementiert werden müssen. Eine klare Linie – zumal sogar neue Befehle entweder in dem Einen oder dem Anderen, aber nie in beiden Modi zur Verfügung gestellt werden – wäre hier hilfreich gewesen.

Gemeinsam ist allerdings Beiden, dass stets eine Antwort auf eine Anfrage abgewartet werden muss, was somit zu einem Halbduplexbetrieb führt. Ansonsten könnten Inkonsistenzen die Folge sein.

3.1.7.5. Benutzer-API

Wie wir sehen konnten, ermöglicht die umfangreiche API des WLAN-Moduls eine sehr detaillierte Steuerung und Überwachung des Datenverkehrs und des derzeitigen Zustands

des Moduls. Dies führt unweigerlich zum Nachteil, dass diese detaillierte Kontrolle zu äußerst umfangreichen Code führt. Deshalb ist es sinnvoll, wiederkehrende Aufgaben in Funktionen zusammenzufassen und in einer darüber liegenden Schicht der Anwendung zur Verfügung zu stellen. Dies wurde mit Einführung der WLAN Benutzer-API in wlanapi.h/.c umgesetzt, welche von nun an als Einzige von der eigentlichen Anwendung eingebunden werden muss, um Zugriff auf das WLAN-Modul zu erlangen.

Verbindungsaufbau und -abbau

```
BOOL connect (BYTE ip);
```

Die Funktion **`connect`** (Abbildung 3.11, Verbindungsaufbau) erstellt eine TCP-Verbindung zu einer Basisstation und vereint bereits mehrere ansonsten teils sehr umfangreiche Aufgaben:

- Aktivieren des WLAN-Moduls

  Das WLAN-Modul wird aus dem Tiefschlafmodus geweckt, woraufhin es versucht sich mit dem voicelink Ad-hoc Funknetzwerk zu verbinden.

- Auswahl einer Basisstation

  Hier stehen zwei Modi zur Verfügung: Einerseits kann durch die Übergabe eines Wertes zwischen 1 bis 10 die letzte Stelle der IP-Adresse einer Basisstation im Subnetz 192.168.1. angegeben werden, mit der eine TCP-Verbindung aufgebaut werden soll. Ist diese nicht verfügbar, so wird auch keine andere Station ausgewählt. Dies ist sinnvoll wenn es erforderlich sein sollte, lediglich bestimmte Basisstation für ausgewählte Dienste zu nutzen.

  Als zweite Möglichkeit kann stattdessen eine „0“ (Null) übergeben werden. In diesem Fall prüft die Mikrofonstation, ob sich aktive Basisstationen im Bereich 192.168.1.1 bis 192.168.1.10 befinden und wählt die zuerst Verfügbare aus. War die Mikrofonstation bereits vorher erfolgreich mit einer Basisstation verbunden, wird diese favorisiert. Dies ist die standardmäßig verwendete Methode einer Mikrofonstation.

- Verbindungsaufbau

  Ist eine Basisstation in Funkreichweite verfügbar, wird versucht eine TCP-Verbindung auf Port 50 aufzubauen. An diesem wartet der für die Mikrofonsationen zuständige TCP-Server auf eingehende Verbindungen. Über **`PCMD_SSTAT()`** wird der

Fortschritt des Verbindungsaufbaus überwacht, bis der interne Zustand **CONNECTED** erreicht ist. Vor dem Nutzer verborgen, wird diese Verbindung von nun an über einen festen Handler mit einer Nummer zwischen 101 und 200 angesprochen. Kann keine Verbindung hergestellt werden, ist dies über den Zustand **CLOSED_TCP** ersichtlich. In diesem Fall muss der Handler wieder freigegeben werden, was durch die Funktion **disconnect** geschieht.

```
BOOL disconnect ( void );
```

Die Funktion dis**connect** (Abbildung 3.11, Verbindungsabbau) überprüft zuerst, ob bei einer noch bestehenden TCP-Verbindung ungesendete Pakete existieren, wartet bis diese versendet wurden und beginnt dann mit dem Abbau der Verbindung. Nur so kann gewährleistet werden, dass kein Datenverlust eintritt.

```
BOOL send_file ( BYTE * data, WORD length);
```

Datenraten sind u.a. abhängig von der Sendestärke, dem Traffic und den Hardwarevoraussetzungen der Gegenstelle. Um so wichtiger ist eine Überwachung der Verbindung, wenn hohe Datenraten erzielt werden sollen. Hierzu wurde eine Flusskontrolle mittels `PUTPACKET2 (0x8A)` und `SSTAT (0x86)` implementiert, welche sicherstellt, dass ein Pufferüberlauf vermieden und somit keine Pakete verloren gehen können. Idealerweise sollte jeder Datenblock maximal 11.680 Byte groß sein. TCP-Pakete können bekanntlich 1460 Byte Nutzdaten enthalten, das WLAN-Modul kann hiervon je Verbindung maximal acht zwischenspeichern. Wurden diese von unserem Mikrocontroller an das Modul übertragen, kann während des Versendens dieser bereits nahezu parallel damit beginnen, einen neuen Datenblock von der SD-Karte zu laden.

```
BOOL sendtext ( char * text ); und BOOL sendtextln ( char * text);
```

Beide sind für die direkte Ansteuerung des TCP-Servers gedacht und bauen auf **send_file()** auf. Durch sie wird die Übertragung von Textkommandos, ähnlich den Textkommandos des WLAN Moduls selbst, erleichtert. Über sie werden zu Beginn einer Übertragung typischerweise Anfragen zur Uhrzeit, dem Scheduler, aber auch Dateigröße und Dateiname übergeben.

```
BOOL receive ( BYTE * data, DWORD availablelength, DWORD * datalength );
```

Eine Empfangsfunktion wurde nur rudimentär implementiert. Derzeit werden hierüber ausschließlich Handshake-ähnliche Protokolle zum sicheren Verbindungsabbau, die Fortsetzung zuvor unterbrochener Dateien und die Übertragung von für die Station relevanter Aufnahmezeiten abgewickelt. Die maximal empfangbare Größe beträgt ein Paket, also 1460 Byte.

3.1.7.6. Energieeinsparung

Während der Ausarbeitung der Diplomarbeit wurde vom Hersteller ein sehr praktikabler neuer Energiesparmodus hinzugefügt. Dieser kann in zwei Stufen aktiviert werden: Mit Hilfe des Textkommandos `WLAN SLEEP` wird der WLAN-Aufsatz (siehe Kap. 2.3.2.4, Avisaro WLAN Modul 2.0 Aufbau) deaktiviert. Zusätzlich kann mit einem einzelnen `SLEEP` ebenfalls der dort verwendete Mikrocontroller des Basismoduls in den Tiefschlafmodus versetzt werden. Hier kann optional auch eine Zeitangabe angefügt werden, nach welcher sich das Modul wieder selbstständig aktivieren soll. Praktikabler ist allerdings die Variante der on-demand Aktivierung. Diese erfolgt aufgrund eines Fehler, anstatt auf dem eigentlich dafür vorgesehenen EINT0, über die Reset-Möglichkeit. Ein wirklicher Nachteil entsteht hierdurch allerdings nicht. Allein durch `WLAN SLEEP` können 200mA eingespart werden, durch `SLEEP` liegt die maximale Stromaufnahme letztendlich bei unter 1mA.

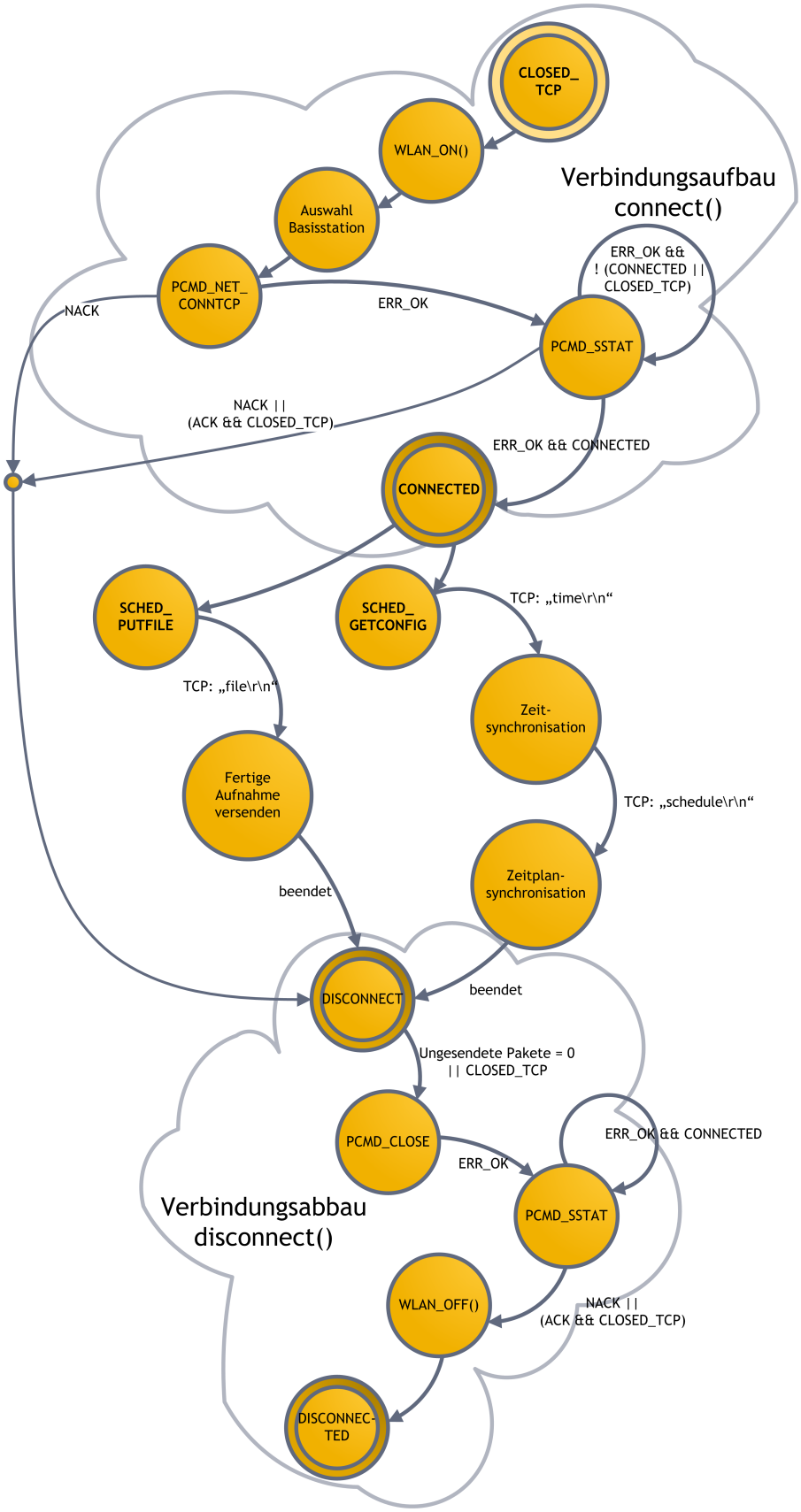


Abbildung 3.11: Schematische Darstellung der Verbindungsverwaltung, in Bezug auf anstehende Aufgaben.

3.1.8. Implementierung des Programmablaufs

3.1.8.1. Scheduler

Der Scheduler ist das Kernstück der Aufgabenverwaltung. Durch ihn werden jegliche Aufnahmen und Aufgaben die eine Kommunikation mit einer Basisstation nach sich ziehen propagiert. Im Grunde handelt es sich dabei hauptsächlich um eine Prioritätswarteschlange mit maximal 50 Elementen vom Typ `struct Schedule_Element`, realisiert als doppelt verkettete Liste.

```
struct Schedule_Element{
      DWORD Sched_id;
      ALARMTime Sched_time;
      BYTE Sched_type;
      struct Schedule_Element *next;
      struct Schedule_Element *prev;
};
```

Diese enthält optional eine ID (zum Beispiel die vom Voiclink-Client vergebene ID einer Aufnahme), ein mit dem Ereignis verknüpftes Datum, die Angabe des Ereignistyps und ein Pointer auf das zeitlich vorige und das nachfolgende Element der Prioritätswarteschlange.

Als Ereignistyp sind derzeit folgende Ereignisse vorgesehen:

```
typedef enum {
      SCHED_HEAD = 0,
      SCHED_STARTRECORD = 0x01,     //Aufnahme starten
      SCHED_STOPRECORD = 0x02,      //laufende Aufnahme beenden
      SCHED_GETCONFIG = 0x04,       //Zeit und Zeitplansynchronisation
      SCHED_PUTFILE = 0x08          //Beendete Dateien übertragen
} SCHEDULE_TYPE;
```

Durch ihren Aufbau können zeitgleiche Ereignisse logisch verUNDet werden. Dies ist nötig, um die Alarmfunktion der RTC nutzen zu können. Diese kann maximal auf sekundengenaue Ereignisse reagieren. Hierzu wird das Datum des Alarms (in unserem Fall die Alarmzeit des ersten Elements in der Prioritätswarteschlange) in die entsprechenden Register der RTC geschrieben und mittels Maskierung der Vergleich für Jahr, Monat, Tag, Stunde, Minute und Sekunde aktiviert. Da die RTC dank ihrer 32 kHz Clock auch dann aktiv ist, wenn sich der Mikrocontroller im Tiefschlafmodus befindet, ist dies neben externen Interrupts die einzige Möglichkeit ihn aus diesem Modus wieder zu wecken. Dies hat somit entscheidende Vorteile für die spätere Implementierung von Stromsparmechanismen.

Für die Wartung der Prioritätswarteschlange bei Änderung der referenzierten Systemzeit, wurden außerdem Wartungsmechanismen eingefügt, welche bestehende Elemente mit der Zeitdifferenz aktualisieren. Diese ermöglichen auch die Erstellung bestimmter Aufgaben in einem selbst gewählten Intervall, indem lediglich die Differenz zur aktuellen Zeit (z.B. sie-

ben Minuten) und der Aufgabentyp übergeben wird. Die dazugehörige Alarmzeit wird daraufhin automatisch berechnet und das Element in die Prioritätswarteschlange eingefügt.

3.1.8.2. Zeitplansynchronisation

Die Zeitplansynchronisation ist zweistufig. Wie in Abbildung 3.11 dargestellt beinhaltet sie u.a. auch den Abgleich der Systemzeit.

Der schematische Ablauf kann folgendermaßen dargestellt werden:

1. Die aktuelle Zeit wird von der Basisstation angefordert. Ist dies erfolgreich dann
2. wird die Prioritätswarteschlange des Schedulers teilweise geleert, indem alle `SCHED_STARTRECORD` und `SCHED_STOPRECORD` Aufgaben entfernt werden. Sollte sich die Systemzeit geändert haben, wird die zeitliche Differenz der verbliebenen Aufgaben bestimmt und neu einfügt. Erst dann wird die neue Systemzeit gesetzt.
3. Ist dies ebenfalls erfolgreich, wird der aktuelle Zeitplan für diese Station angefordert, in das gültige Format konvertiert und in die Prioritätswarteschlange eingefügt. Dabei kann ebenfalls erkannt werden, ob eine Aufnahme bereits aktiv sein sollte (Startzeit in der Vergangenheit, Endzeit in der Zukunft). Hier wird diese Aufnahme sofort nach Beendigung der Zeitplansynchronisation durch setzen des `SCHED_STARTRECORD` -Flags begonnen. Um die Anforderung an den Speicher zu minimieren, wurde die Anzahl der Elemente in der Prioritätswarteschlange auf 50 begrenzt. Hiervon sind 10 für Systemaufgaben (z.B. Aufnahmen senden) reserviert, so dass effektiv 20 Aufnahmen (20 `SCHED_STARTRECORD` und 20 `SCHED_STOPRECORD`) verwaltet werden können.
4. Danach kann mittels Handshaking die Verbindung erfolgreich abgebaut werden. Die Information, dass erfolgreich mit einer Basisstation verbunden werden konnte, wird anschließend dazu benutzt zu überprüfen ob ungesendete Dateien vorhanden sind und eine Aufnahmeversendung ausgelöst.

In Abbildung 3.12 ist der auf Textkommandos basierende Kommunikationsablauf zur Verdeutlichung visualisiert. Hier kann auch das entsprechende Format einer Schedules entnommen werden (Aufnahme-ID, Startdatum, Enddatum).

	Mikrofonstation	**Basisstation**
1	CONNECT()	
2		trägt Station im Mikrofonverzeichnis ein
3	time	
4		2009 11 25 16 53 18
5	ok	
6	schedule	
7		437 2009 11 25 16 46 18 2009 11 25 16 59 18
8	ok	
9		438 2009 11 25 17 05 33 2009 11 25 18 45 33
10	ok	
11		443 2009 11 27 00 00 36 2009 11 27 05 00 36
12	ok	
13		0
14	stop	
15		stop
16	DISCONNECT()	

Abbildung 3.12: Kommunikationsablauf Zeitabgleich und Zeitplanübertragung

3.1.8.3. Aufnahme

Die Aufnahme selbst ist sehr einfach aufgebaut. Bei Erhalt eines *SCHED_STARTRECORD* wird zu Beginn neben dem Mounten des Dateisystems eine Textdatei angelegt, welche die ID der Aufnahme und die reale Startzeit enthält. Darauf folgend wird die eigentliche RIFF-Wave Audiodatei (benannt nach der ID.wav) angelegt und der AD-Wandler aus dem Schlafmodus geweckt. PCM-Audiodaten werden aus dem globalen Ringpuffer ausgelesen, welcher über per I²S generierte Interrupts beschrieben wird. Diese werden in die Datei übertragen. Ausgelöst durch ein *SCHED_STOPRECORD* wird die Aufnahme zeitnah beendet. Das *SCHED_PUTFILE* -Flag wird gesetzt, um zu signalisieren, dass eine fertiggestellte und ungesendete Datei vorliegt.

Als Besonderheit ist beim globalen Ringpuffer zu bemerken, dass sich dieser im 16kByte großen Ethernet-RAM befindet. Diese Größe ist erforderlich um Aussetzer während der Aufnahme zu vermeiden. Gleichzeitig wird der 32kByte RAM in dem sich die Programmdaten befindet entlastet.

3.1.8.4. Aufnahmeversendung

Die Aufnahmeversendung (Abbildung 3.13) erscheint zunächst sehr umfangreich. Neben der Verbindungsherstellung wird die Aufnahme-ID und die reale Startzeit übertragen, aus denen der letztendliche Dateiname generiert wird. Sollte diese Datei bereits vorhanden sein, wird die Fortsetzungsposition der Mikrofonstation mitgeteilt. Aufgrund möglicher Verbindungsabbrüche ist eine sehr ausführliche Fehlerbehandlung notwendig, welche in der schematischen Darstellung in Abbildung 3.13 erfasst ist. Eine genauere Beschreibung des Ablaufs auf Serverseite ist Kapitel 3.2.2. nachzulesen.

3.1.8.5. Verlustfreie Kompression

Grundsätzlich steht mit Wavepack eine verlustfreie Kompression auch für Low-Power Plattformen bereit. Die derzeitige Implementierung erfordert allerdings 48kByte RAM für Quantizierungstabellen und ist somit in der derzeitigen Form nicht in das Projekt integrierbar. Mit dem Umstieg auf den MSB-A2 [BWB08] könnte dieses Problem durch den vorhandenen 98kByte großen RAM umgangen werden. Idealerweise sollte hier allerdings eine neue Kompression ergänzt werden, welche neben fertiggestellten Dateien auch Streams verarbeiten kann, denn bisher wird bei der Initialisierung von einer festen Dateigröße ausgegangen, welche bei einem Stream – also einer laufenden Aufnahme – normalerweise nicht bekannt ist. In der derzeitigen Form wird die Original RIFF-Wave Datei unkomprimiert übertragen und nachträglich auf der Basisstation komprimiert.

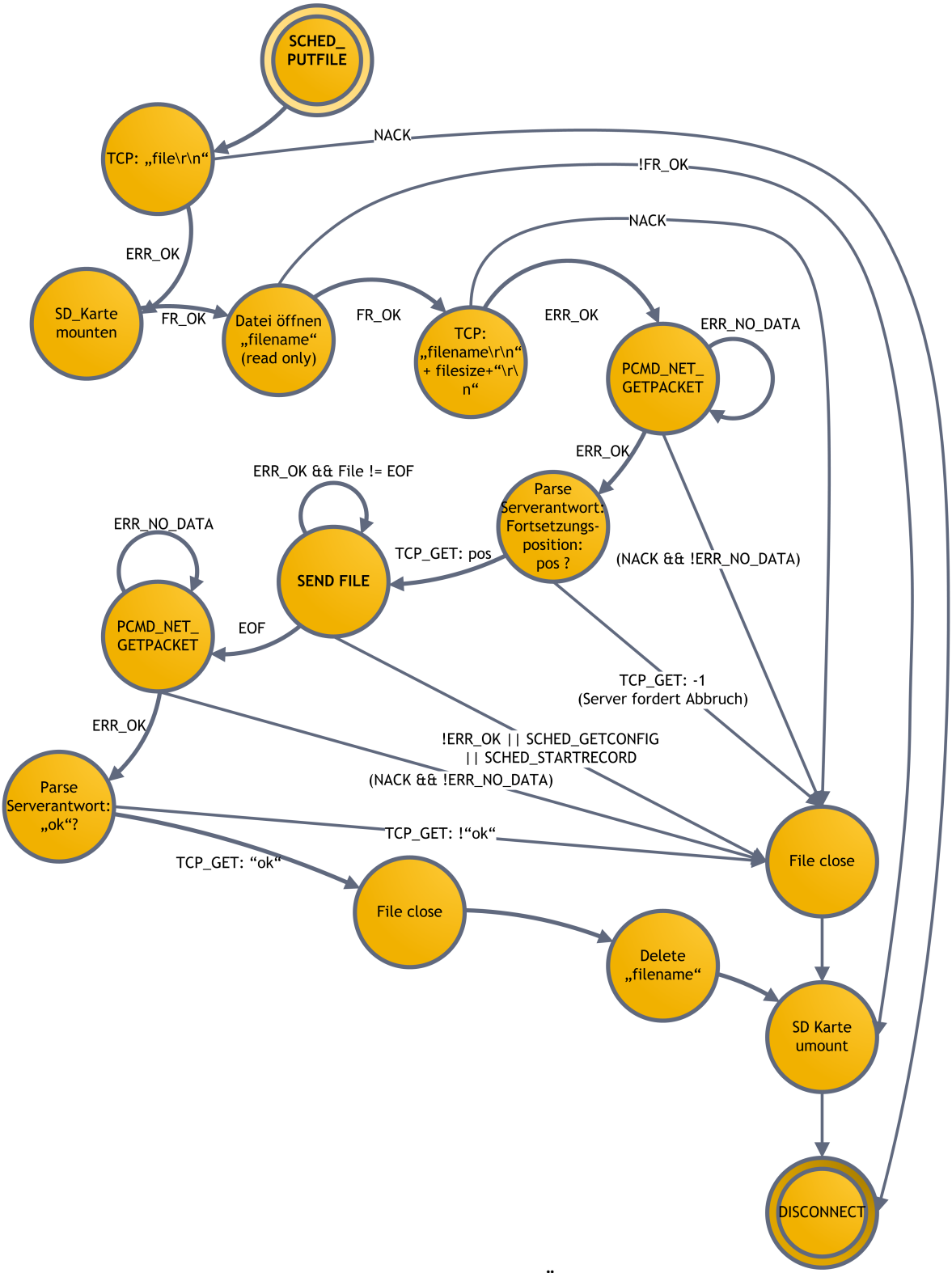


Abbildung 3.13: Protokoll und Fehlerbehandlung bei der Übertragung einer fertiggestellten Datei an eine Basisstation.

3.2. Erweiterungsservice der Basisstation

Als Partner für unser entworfenes Kommunikationsprotokoll stehen uns x86/Windows basierte Basisstationen in Form von Asus EEE PC 701 gegenüber. Auf ihnen ist Windows XP und das .NET Framework 3.5 installiert. Wie in Kapitel 1.5.: „Vorhandene Infrastruktur“ erläutert, sind dort sämtliche Aufgaben (Aufnahme, Verteilung der Zeitpläne innerhalb des Mesh-Netzes,...) als Windows-Service umgesetzt. Dies garantiert, dass abgestürzte Services automatisch von Windows neu gestartet werden. Der Ausfall einer Station wird somit zumindest unwahrscheinlicher (obwohl der Idealfall natürlich die generelle Vermeidung von Abstürzen sein sollte). Vorteilhaft ist vor Allem für den Anwender, dass dieser nicht dafür verantwortlich ist Programme nach dem Systemstart aufzurufen oder diese gar unabsichtlich zu beenden. Aus diesem Grund soll der hier umgesetzte Erweiterungsservice ebenfalls diesem Prinzip folgen. Nähere Informationen zum Entwurf von Erweiterungsservices können der Diplomarbeit von Frank Beier entnommen werden. [FB09] An dieser Stelle soll lediglich ein kurzer Überblick gegeben werden.

3.2.1. Entwicklungsumgebung

Als typisches Element für die .NET Entwicklung wurde Microsofts Visual Studio 2010 in der Professional Edition eingesetzt. Dieses bietet bereits Templates für den Entwurf von Services an. Das Grundprinzip ist hierbei, dass der eigentliche Programmentwurf nach außen lediglich eine Start(), Restart() und eine Stop() Methode besitzt, die nach Systemstart, einem Absturz des Services oder dem Herunterfahren des Betriebssystems durch die Diensteverwaltung aufgerufen werden kann. Für unser Projekt wurde der Dienst entsprechend „ARM_Server“ benannt. Für eine einfache Installation wurde ein Project Installer hinzugefügt. Die Anwendung wird im Anhang beschrieben. Besondere Erwähnung sollte die Tatsache finden, dass der Service lediglich als „Debug“ und nicht als „Release“ compiliert wurde. Über das Programm „Debugview“ können hierdurch die aktuellen Aufgaben nachverfolgt werden.

3.2.2. TCP-Server

Die Kernkomponente des erstellten Erweiterungsservice ist ein TCP-Server. Dieser hat die Aufgabe auf Port 50 eingehende Verbindungswünsche zu bearbeiten, also eine Verbindung zu erstellen und diese auf einen neuen Port zu legen. Dadurch ist es möglich weitere Verbindungsversuche ebenfalls über Port 50 abzuwickeln. Um nicht blockierend zu wirken ist es allerdings notwendig gewesen, die Verbindungen in separate Threads auszulagern. Hier-

durch ist die Anzahl der aktiven Verbindungen lediglich durch die Anzahl der nutzbaren Ports begrenzt.

Nachfolgend wird – allein durch die IP-Adresse identifiziert – überprüft wird ob diese Mikrofonstation in der XML-basierten Datenbank (Abbildung 3.14) bereits bekannt ist. Ist dies nicht der Fall wird diese in der Form „ARMxxx“ (xxx steht für die zwei bis dreistellige IP-Adresse) und dem gesonderten Typ 4 ergänzt (Typ 1 und 2 sind interne und externe Soundkarten der Basisstation). Die Änderung wird durch die bestehenden Services über das Mesh-Netzwerk propagiert.

```
<status_set>
  <status>
    <id>ARM74</id>
    <type>4</type>
    <optional>
    </optional>
  </status>
</status_set>
```

Abbildung 3.14: XML-Format einer eingetragenen Mikrofonstation in der Status.xml

Die Mikrofonstation befindet sich daraufhin – ähnlich einem Hauptmenü – in einem switch-case. Hier sind für die vorgestellten Aufgaben wie der Vergleich der Uhrzeit, der Zeitpläne und dem Upload einer Datei Verweise auf Subroutinen enthalten. Durch ein vereinbartes Handshaking kann bei einem Abbruch der Verbindung auch unterschieden werden, ob dieser ausgehandelt war. Eine Besonderheit bildet hier die Dateiübertragung, welche eine Serverbasierte Uploadfortsetzung anbietet. Hierbei wird der Dateiname verglichen und bei einer bereits existierenden Datei der exakte Fortsetzungspunkt (bisherige Dateigröße in Byte) an die Mikrofonstation übertragen. Alle ankommenden Daten können an die bestehende Datei angehängt werden. Nur so ist es effektiv möglich Dateien in Größenordnungen von 2 bis 2,5 GByte zu übertragen. Vollendete Dateien werden mit Wavepack verlustfrei auf die Hälfte ihrer Größe komprimiert und in die Download.xml eingetragen. Ab diesem Zeitpunkt steht die Aufnahme über die Voiclink-Client Software zum Download bereit.

Im Grunde können beliebige Stationstypen die dieses Protokoll (Details Kapitel 3.1.8.) einhalten, diesen Service nutzen.

4. Auswertung

Um einen Überblick über die Eigenschaften des gesamten Systems zu erlangen, ist eine Analyse der einzelnen Bestandteile des Aufbaus notwendig. Danach sollen mittels eines Testaufbaus die Anwendung im Einsatzgebiet nachgestellt werden.

4.1. Testaufbau

Die Erweiterungsplatine (grün) wurde extern hergestellt und selbstständig bestückt. Die Bauteile wurden hierbei von mehreren Anbietern, u.a. Segor und Farnell, bezogen. Abbildung 4.1 zeigt die erste Revision, mit der die nachfolgenden Tests durchgeführt wurden. Das vorgestellte Layout in Kapitel 2.7.5. enthält bereits eine fehlerbereinigte, zweite Version, in der u.a. einem saubereren Aufbau und der Integration des abstehenden Mikrofonkompressors Tribut gezollt wurde. Das WLAN-Modul ist genauso wie die Erweiterungsplatine über Sockel gesteckt und kann bei Bedarf entfernt und wiederverwendet werden.

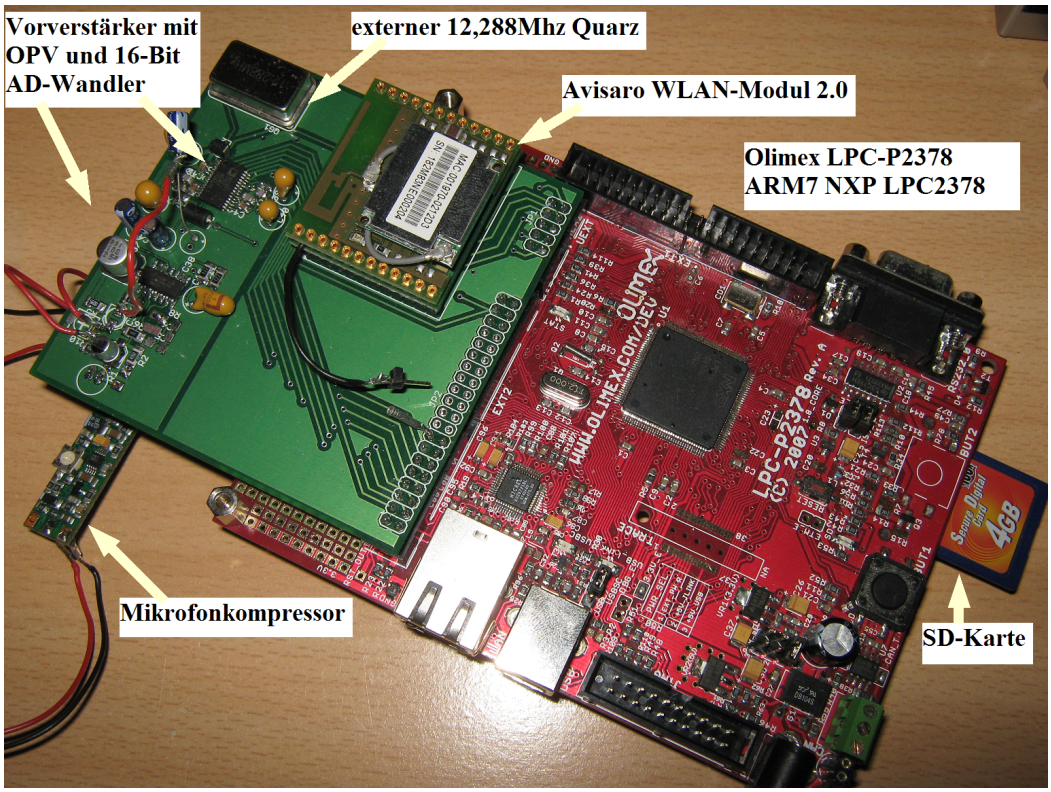


Abbildung 4.1: Evaluationsplatine mit aufgesteckter erster Revision der Erweiterungsplatine (grün)

Die Evaluationsplatine (rot) bietet vielfältige Anschlussmöglichkeiten, u.a. für die Stromversorgung und den Einsatz von SD-Karten. Der Einsatz einer Evaluationsplatine bietet außerdem gegenüber einer Platine für die Serienfertigung hervorragende Debugging-Möglichkeiten über RS232 und die JTAG-Schnittstelle.

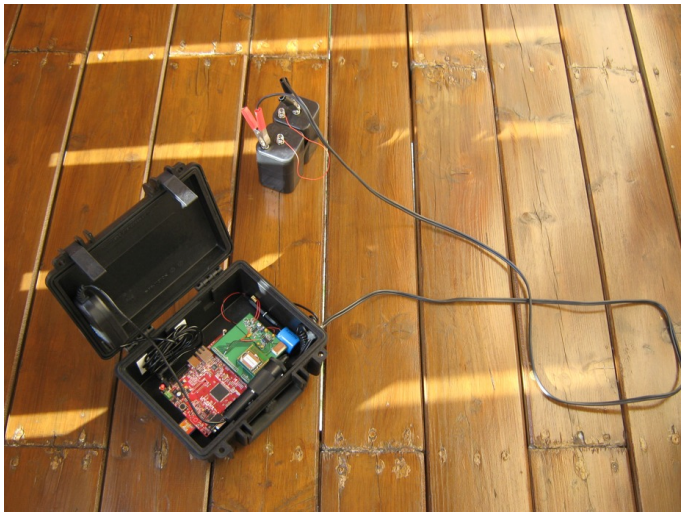


Abbildung 4.2: Outdoortest im wasserdichten Gehäuse

Um den Prototyp vor einem Kurzschluss zu schützen, wurde dieser in einem wasserdichten Gehäuse, dem in Kapitel 2.6. vorgestellten Modell Hofbauer Explorer 6, eingesetzt. Auch der Spannungswandler, welcher die externe 12V Versorgungsspannung zunächst auf 9V reduziert, ist hierbei im Deckel des Gehäuses und somit geschützt angebracht. Die eigentliche Spannungsquelle ist in Abbildung 4.2 beispielhaft repräsentiert durch zwei in Reihe geschaltete 6V, 7Ah Baustellenbatterien für Warnleuchten.

In diesem Fall stellt sich selbstverständlich die Frage, in wiefern das luftdicht verschlossene System von einem Hitzeproblem betroffen sein könnte. Hierzu wurde ein 1-Wire/iButton im Gehäuse platziert um den Temperaturverlauf zu überwachen, welches an einem schattigen Platz positioniert wurde. Die Umgebungstemperatur betrug hierbei 26°C. Wie in Abbildung 4.3 ersichtlich, stieg die Innentemperatur innerhalb von 50min auf 29°C an und blieb konstant über einen Verlauf von über 2 Stunden. Eine Überhitzung (sofern direkte Sonne vermieden wird) ist somit nicht zu erwarten.

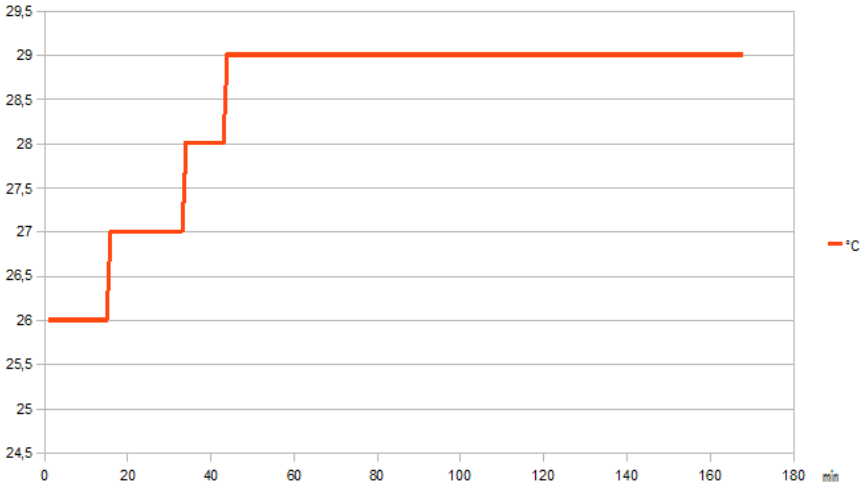


Abbildung 4.3: Innentemperatur des Gehäuses bei Betrieb

Trotzdem war an der Unterseite ein wärmerer Bereich zu ertasten, der deutlich von der gemessenen Innentemperatur abzuweichen schien. Hier konnte ein auf der Olimex-Platine verbauter Einphasen Gleichrichter DB104 identifiziert werden. Der hierüber auftretende Spannungsabfall wird in Wärme umgewandelt, so dass an dieser Stelle lastabhängig eine Temperatur von bis zu 56°C auftritt (Abbildung 4.4). Sollte dies später ein Problem darstellen – oder sollte eine Batterie mit niedrigerer Spannung verwendet werden – so kann der Gleichrichter durch die Nutzung von VIN auf dem externen Sockel der Olimex-Evaluationsplatine umgangen werden. Die Eingangsspannung muss hier mindestens 6,5V betragen [LM06], um die Spannungsversorgung über den nachfolgenden LM1117 von 5V zu garantieren. Allerdings müsste hierdurch auf einen netzteilkompatiblen Anschluss verzichtet werden.

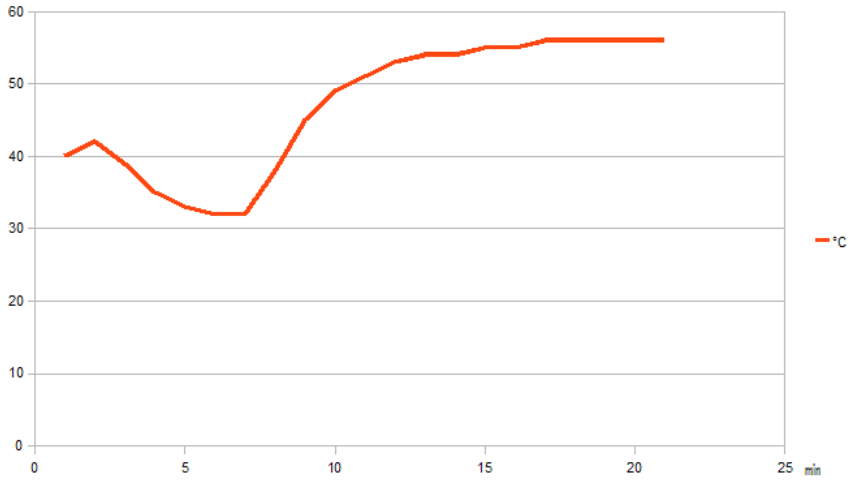


Abbildung 4.4: Temperatur des Einphasen-Gleichrichters DB104

4.2. Testlauf

Für den nachfolgenden Test wurde als Speichermedium auf eine 4GB SD-Karte (non-SDHC) von Transcend eingesetzt, welche in FAT16 formatiert wurde und somit eine nutzbare Kapazität von ca. 3,7GB bietet. Der problemlose Einsatz von 16GB SDHC-Karten mit FAT32 wie sie die Basisstationen zur Speicherung verwenden, konnte aus Zeitgründen nicht abschließend geklärt werden. Erste Tests verliefen allerdings vielversprechend und hinterließen einen positiven Eindruck.

Als Szenario wurde über den Verlauf von drei Tagen verteilt, sieben Aufnahmen geplant. Dabei stand bereits zu Beginn fest, dass die zur Verfügung stehende Aufnahmekapazität nicht ausreichen könnte. Hierdurch soll gleichzeitig das Verhalten des Systems im Grenz-

bereich ermittelt werden, auch wenn dieser im Alltagsbetrieb möglicherweise nie auftreten mag.

Initial meldet sich die Mikrofonstation nach der Inbetriebnahme an einer erreichbaren Basisstation an und trägt sich in der dort vorhandenen Datenbank als gültiges Mikrofon ein. Hierdurch kann dieses in der Voicelink-Client Software ausgewählt und der dazugehörige Zeitplan erstellt werden. Anschließend wird dieser er an die Basisstationen verteilt. Durch den regelmäßigen Abgleich des Zeitplans im Abstand von sieben Minuten, wird einer Mikrofonstation ohne aktive Aufnahmen ein neuer Zeitplan spätestens nach diesem Zeitraum bekannt. Der normale Betrieb ist auf der Olimex-Platine an einer blinkenden Status-LED (im Schaltplan „USB_LINK“ benannt), eine Aufnahme an einer zusätzlichen dauerhaft leuchtenden LED („STAT“) in der Nähe der Erweiterungsplatine zu erkennen (nicht zu verwechseln mit der Gegenüberliegenden („SD“), welche durchgehend aktiv ist und lediglich indiziert, dass die SD-Karte erfolgreich initialisiert wurde).

Tag	ID	Startzeit	Endzeit	Dauer (geplant)	Dauer (real)	Bemerkung
1	457	16:46:18	16:59:18	00:13:00	00:05:55	Beginn 16:53:22
1	458	17:05:33	18:45:33	01:40:00	01:40:05	
1	459	19:00:11	01:00:11	06:00:00	00:06:05	
2	460	02:00:51	10:00:51	08:00:00	04:50:02	Vorzeitig beendet (Speicher)
2	461	12:00:30	13:00:30	01:00:00	00:00:00	Mangels Speicher verworfen
2	462	14:00:09	20:00:00	06:00:00	00:00:00	Mangels Speicher verworfen
3	463	00:00:36	05:00:36	05:00:00	04:50:02	Vorzeitig beendet (Speicher)

Tabelle 4.1: Programmierte Aufnahmen und ihr Verlauf, Angaben in hh:mm:ss

In Tabelle 4.1 können der Zeitplan und die reale Aufnahmedauer entnommen werden. Aufnahme 457 wurde später begonnen, da der Zeitplanabgleich erst nach der Startzeit stattfand. Dies wurde dadurch herbeigeführt, dass die Mikrofonstation erst verspätet eingeschaltet wurde. Hierdurch sollte nachgewiesen werden, dass der zeitlich zurückliegende Aufnahmebeginn erfolgreich erkannt und die Aufnahme zeitverzögert gestartet wird.

Während die Aufnahmen 458 und 459 erfolgreich waren, war durch die fehlende Zeit zum Versenden der fertiggestellten Dateien der Speicherplatz während der Aufnahme mit der ID 460 erschöpft. Diese wurde deshalb vorzeitig – aber korrekt – beendet. Die Datei ist lesbar. Aufgrund der derzeitigen Implementierung konnte die dadurch zur Verfügung stehende Restaufnahmezeit nicht genutzt werden, um vorzeitig mit dem Dateiversand zu beginnen. Dies ist auch korrekt, denn der Dateiversand ist stets mit dem Abgleich des Zeitplans ver-

bunden. Bei diesem würde festgestellt werden, dass derzeit eine Aufnahme stattfinden müsste – und somit zu einer Endlosschleife führen.

Die Aufnahmen 461 und 462 konnten wie erwartet mangels Speicherplatz nicht begonnen werden und wurden verworfen. Die Länge von Aufnahme 463 mag verwundern, ist sie doch identisch mit Aufnahme 460. Hintergrund ist jedoch die interne Entscheidung welche Datei aktuell versendet werden soll. Diese hängt von der Rangfolge im Fat-Dateibaum ab. So konnte bisher Aufnahme 460 als einzige Aufnahme erfolgreich versendet werden, obwohl sie nicht die Erste vom Fertigstellungsdatum, wohl aber im Fat-Dateisystem darstellt. Der durch sie freigegebene Speicherplatz reichte somit exakt aus, um erneut eine 04:50:02 lange Audiodatei anzulegen.

Der durchschnittlich erzielte Datendurchsatz zu den Basisstationen beträgt zwischen 98 kByte/s im schlechtesten Fall und 129 kByte/s maximal. Betrachtet man dies bereinigt von den Lesevorgängen der SD-Karte (wiederholtes Senden eines unveränderten Puffers), konnten 162 kByte/s erzielt werden. Die maximal mögliche Herstellerangabe von 2,6 MBit/s, also rund 325kByte/s konnten leider nicht erreicht werden und bieten somit Raum für Optimierung.

Zur Verdeutlichung: Die Größe einer 6stündigen Aufnahme beträgt 1.977,5 MByte und würde eine Übertragungsdauer zwischen 344 und 262 min erfordern. Im Idealfall, also einem Datendurchsatz von 2,6 MBit/s und der Nutzung einer verlustfreien Kompression (z.B. Wavepack) mit einer Kompressionsrate von 50%, könnte diese auf 52min verkürzt werden.

Letztendlich kann der Testlauf als erfolgreich angesehen werden, der proof-of-concept konnte erbracht werden. Ausnahmefälle wurden korrekt behandelt und Aufnahmen – abhängig vom verfügbaren Speicherplatz – erfolgreich fertiggestellt. In der kommenden Entwicklung sollten bereits begonnene Uploadvorgänge bevorzugt behandelt und der Datendurchsatz optimiert werden.

4.3. Schaltungsanalyse

In der ursprünglich geplanten Version der Erweiterungsplatine mit integrierter Vorverstärkerschaltung, hat sich deren Spannungsversorgung über die von der Olimex LPC-P2378 Platine zur Verfügung gestellte 5V Spannung als Schwachpunkt erwiesen. Beim Entwurf der Schaltung wurde nicht berücksichtigt, dass bereits minimale Schwankungen in der Spannungsversorgung mitverstärkt und in der Aufnahme zu hören sein könnten. Da auf

Stromsparmechanismen des Mikrocontrollers während der Aufnahme verzichtet wurde und die Stromaufnahme des Analog-Digital-Wandlers ebenfalls konstant ist, hat sich herausgestellt, dass der Zugriff auf die SD-Karte zur Speicherung des Audiostreams für die regelmäßigen Störgeräusche verantwortlich ist. Zur Visualisierung des Rauschens innerhalb eines 300 ms langen Teilstücks wurde die Software Audacity herangezogen.

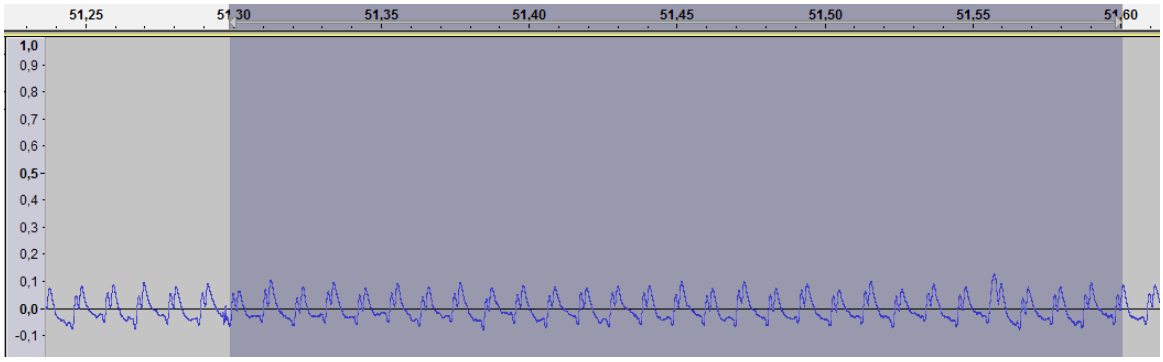


Abbildung 4.5: Regelmäßige Störgeräusche in der Aufnahme, bedingt durch SD-Karten Zugriffe, geschrieben werden 1024 Byte Blöcke.

In Abbildung 4.5 ist hierzu eine regelmäßige Frequenz zu beobachten: ca. 28,3 Peaks in 300ms ergeben eine Frequenz von 94,33 Hz. Bei dieser Aufnahme wurden 1024 Byte in einem Schreibvorgang auf die SD-Karte übertragen. Dies ergibt ca. 96.597 Byte pro Sekunde. Gehen wir davon aus, dass 96.000 Byte pro Sekunde geschrieben werden müssen $(48\,kHz \cdot 16\,Bit)$ ist dies, abgesehen von durch das Ablesen bedingte Rundungsfehler, miteinander identisch.

Um nun gegenzutesten, wurde die zu schreibende Blockgröße auf 512 Byte verringert. Da ebenfalls die Mindestgröße eines Speicherblocks 512 Byte ist, wird erst dann ein Schreibvorgang ausgelöst, wenn diese Datenmenge erreicht ist. Wir erwarten nun eine doppelt so hohe Frequenz, da sich die Anzahl der einzelnen Schreibvorgänge verdoppelt.

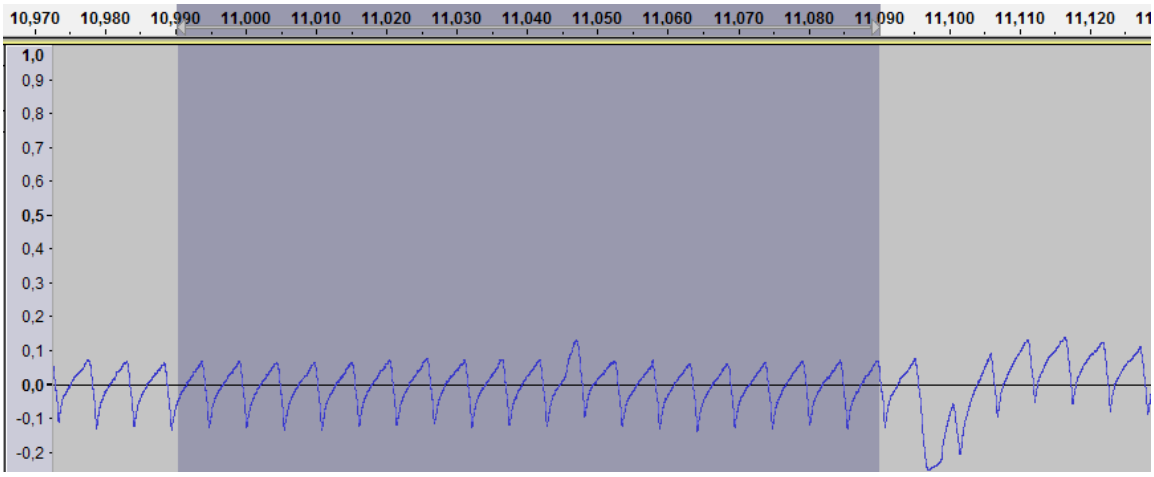


Abbildung 4.6: Werden 512-Byte große Blöcke geschrieben, verdoppelt sich die Frequenz des Störsignals.

Wie in Abbildung 4.6 zu erkennen, hat die Spannung nun gegenüber dem vorigen Fall keine Zeit mehr sich zu stabilisieren und geht in ein Sägezahnmuster über. Wie erwartet hat das Signal eine doppelt so hohe Frequenz von etwa 188 Hz und lässt auf einen Datenstrom von ca. 96.256 Byte pro Sekunde schließen.

Der Ursprung des Problems liegt im verwendeten Spannungswandler LM1117 der Firma National Semiconductor. Dessen eigentliche Aufgabe ist die über ein externes Netzteil oder den Pin VIN hinzu geführte Spannung auf 5V zu stabilisieren und sie den einzelnen Verbrauchern zur Verfügung zu stellen. Je nach Lastwechsel, wie er zum Beispiel beim Einschalten von Peripherie zu Stande kommt, sinkt die Spannung allerdings kurzzeitig unter 5V. Sinkt die Last wieder, hat dies den gegenteiligen Effekt und die Spannung steigt kurzzeitig über 5V, bevor erneut versucht wird diese zu stabilisieren. Genau dieses Verhalten ist auch dem Datenblatt des Spannungswandlers zu entnehmen: Entfernt man die Stabilisierungsphase zwischen dem Spannungseinbruch und der Lastsenkung in Abbildung 4.7, so erhalten wir ein Sägezahnmuster, welches dem in Abbildung 4.6 signifikant ähnelt.

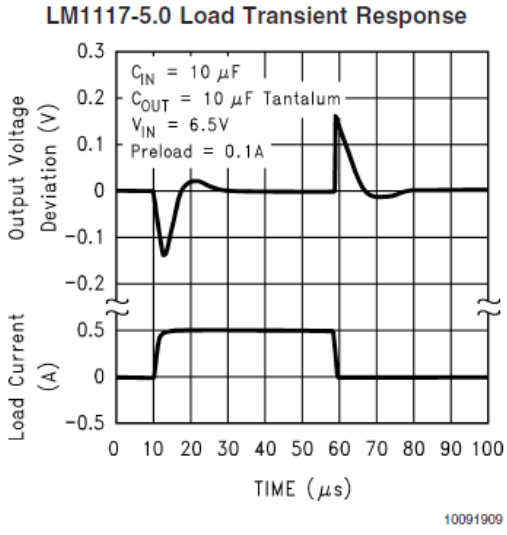


Abbildung 4.7: Spannungseinbruch am LM1117 bei Lastwechsel [LM06]

Nicht genau bestimmt werden konnte, warum das Zeitverhalten des LM1117 nicht mit dem Datenblatt übereinstimmt. So ist aus Abbildung 4.7 zu entnehmen, dass beide Vorgänge zusammen maximal 20µs dauern sollten. Das heißt, dass bei einer Schreibfrequenz von 94 Hz oder 188 Hz, maximal jeder 510., bzw. 255. Messwert betroffen sein sollte. Vermutet werden allerdings Ladevorgänge innerhalb der Schaltung der Olimex-Platine.

Da die generierten 5V des LM1117 für die Versorgungsspannung des Elektretmikrofons, die Vorverstärkerschaltung und den Operationsverstärker verwendet wird, zeigen sich be-

reits kleine Schwankungen in der Tonaufnahme als Rauschen. Gehen wir von einer Verstärkung von etwa 125 bis 500 bei einer erwarteten Spannungsdifferenz von maximal 2,5V zur Ruhelage am AD-Wandler Eingang aus, so reicht eine Spannungsänderung von 1mV aus, um bereits 5% bei 125facher Verstärkung auszusteuern und 20% bei 500facher Verstärkung.

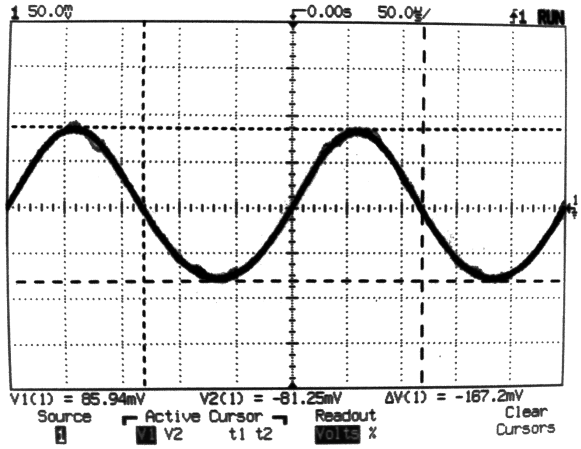
Aus diesem Grund wurde die Verstärkung auf den Faktor drei gesenkt und eine batteriebetriebene Vorverstärkerschaltung – der Mikrofonkompressor – davorgeschaltet, um kurzfristig eine tragfähige Lösung für eine gute Aufnahmequalität zu schaffen. Das beschriebene Rauschverhalten konnte hierdurch unterdrückt werden.

4.4. Tonqualität

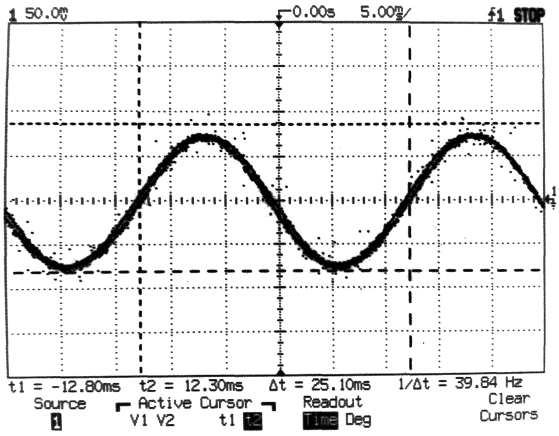
Die derzeitige Ausrichtung des Systems auf die Aufnahme von Gesang von Singvögeln, erfordert eine ausführliche Analyse der erzielten Qualität, um gegebenenfalls Schlussfolgerungen für nachfolgende Revisionen ziehen zu können. Diese wird unterteilt in die Analyse der Grenzfrequenzen, sowohl innerhalb der Vorverstärkerschaltung, als auch in der fertiggestellten Aufnahme.

Zunächst wurden die Grenzfrequenzen des ersten analogen Teilstücks, dem nachträglich hinzugefügten Mikrofonkompressor betrachtet. Hierzu wurde anstatt eines Mikrofons, eine Sinusspannung mit einem Frequenzgenerator erzeugt. Die Amplitude beträgt 20mV, was die minimal wählbare Einstellung darstellt. Dies entspricht in etwa dem Bereich, welcher ein sehr lauter Ton in einem Elektret-Mikrofon erzeugen kann. Für die Messung sind allerdings die Frequenzen interessant, welche das Ursprungssignal gegenüber dem optimalen Wirkungsbereich des Mikrofonverstärkers verändern. In Tests konnte mit einem HP 54600A Oszilloskop herausgefunden werden, dass die untere Grenzfrequenz bei etwa 40Hz liegt. Hier verringert sich die positive maximale Amplitude unter den Wert von 859,4mV und deutliches Rauschen ist zu erkennen (Abbildung 4.8). Bei der oberen Grenzfrequenz wurde lediglich der für einen Menschen relevante Bereich hörbarer Frequenzen bis 20kHz betrachtet und die Bestimmung daher bei dieser Grenze abgebrochen. Wie zu erkennen ist, wird das Signal unverändert wiedergegeben (Abbildung 4.8 unten), der Mikrofonkompressor ist somit für Frequenzen bis 20 kHz geeignet.

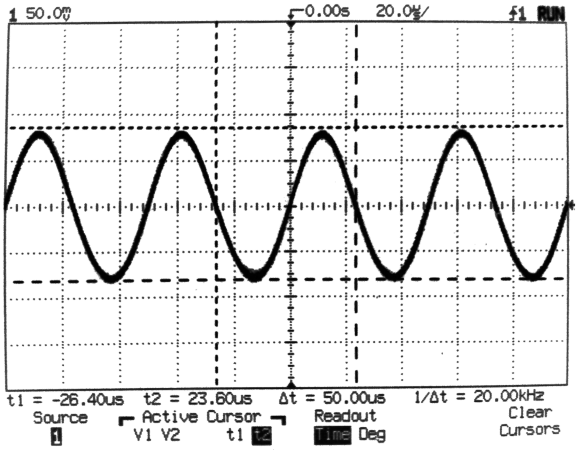
Ein ähnliches Bild ergibt sich, wenn nun die vollständige zweistufige Vorverstärkerschaltung und somit das Signal am Eingang des AD-Wandlers betrachtet wird. Da das Signal mit einer Referenzspannung verglichen wird, ist der Nullpunkt um die Größe dieser Span-



Ausgangssignal des Mikrofonkompressors
859,4mV bis -812,5mV (Line-Out Pegel)



untere Grenzfrequenz ~40Hz
(erkennbar: einbrechende Amplitude)

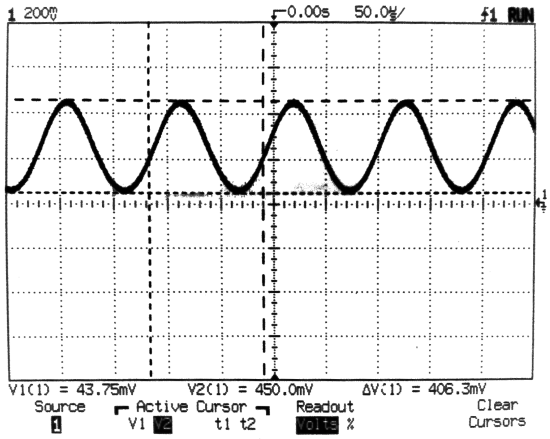


obere relevante Grenzfrequenz ~20kHz
(keine einbrechende Amplitude erkennbar)

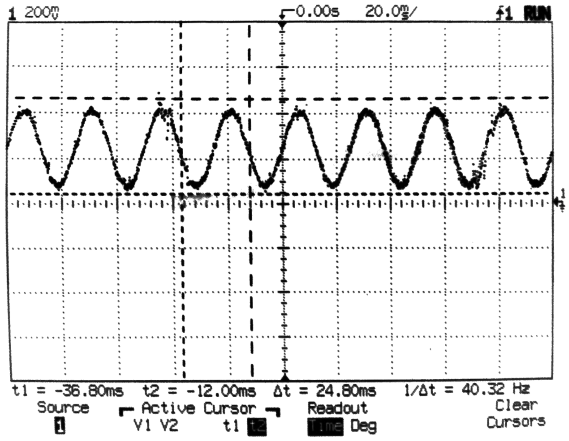
Abbildung 4.8: Grenzfrequenzen des Mikrofonkompressors (Messkopf Teiler 1:10)

nung in den positiven Bereich verschoben. Hiervon abgesehen, ergeben sich keinerlei Unterschiede zur vorigen Messung. Da sich die untere Grenzfrequenz selbstverständlich nicht verbessern, also unter 40 kHz sinken kann (begrenzt durch die untere Grenzfrequenz des Mikrofonkompressors aus unserer vorigen Messung), so bleibt sie doch zumindest bei 40Hz. Eine Einschränkung zeigt sich auch hier in der oberen Grenzfrequenz nicht und so kann davon ausgegangen werden, dass Frequenzen bis 20kHz ohne Verzerrung den AD-Wandler Eingang erreichen können.

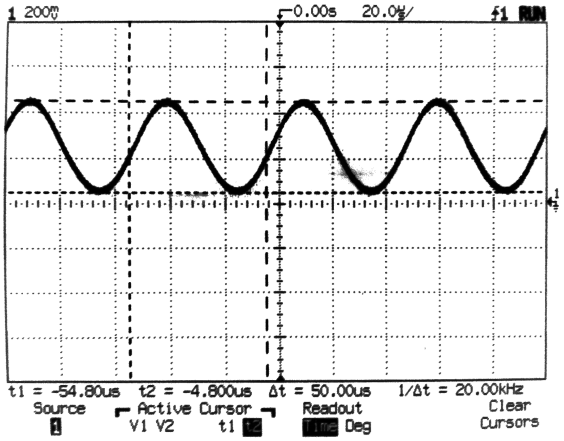
Ein anderes Bild zeigte sich allerdings bei der Analyse mit der Software SASLab Pro von Avisoft, wie sie von der AG Verhaltensbiologie verwendet wird. Obwohl die Aufnahmen der Mikrofonstation gegenüber denen der Basisstation noch das beste Bild zeigten weil sie mehr Informationen enthielten (Abbildung 4.10), waren hohe Frequenzen verloren gegangen (Abbildung 4.11). Die Aufnahme wirkt in diesen Bereichen leicht dumpf, fehlende Höhen sind somit bei der Gegenüberstellung mit dem Original leicht erkennbar. Der Grund



Signal am Eingang des AD-Wandler
(4,5V bis 437,5mV, Differenz 4,06V)



untere Grenzfrequenz ~40Hz
(deutlicher Einbruch der Amplitude und Rauschen erkennbar)



obere relevante Grenzfrequenz ~20kHz
(keine einbrechende Amplitude erkennbar)

Abbildung 4.9: Grenzfrequenzen am Eingang des AD-Wandlers (Messkopf Teiler 1:10)

hierfür ist das verwendete Elektret-Mikrofon, welches nicht mit dem Mikrofon übereinstimmt, mit dem die Originalaufnahme erstellt wurde. Dieses besitzt im Vergleich eine niedrigere obere Grenzfrequenz, da es vermutlich vorrangig für die Aufzeichnung von Sprache konzipiert wurde. Vom geplanten Einsatz dieser Mikrofone in der Forschung muss somit abgeraten werden.

Viel schlechter ist allerdings die Prognose für die verwendete EEE PC Plattform der Basisstationen. Diese besitzen zusätzlich in der internen Soundkarte einen weiteren Tiefpassfilter, der die zusätzlich fehlenden Informationen gegenüber der Aufnahme der Mikrofonstation erklärt. Sollte dies eine gängige Praxis sein, z.B. auch bei externen USB-Soundkarten, so könnten externe Soundkarten mit Line-In Eingang verwendet werden. Das Signal der Mikrofone könnte mit dem nahezu verlustfreien Mikrofonkompressor entsprechend dem benötigten Line-In Pegel verstärkt werden.

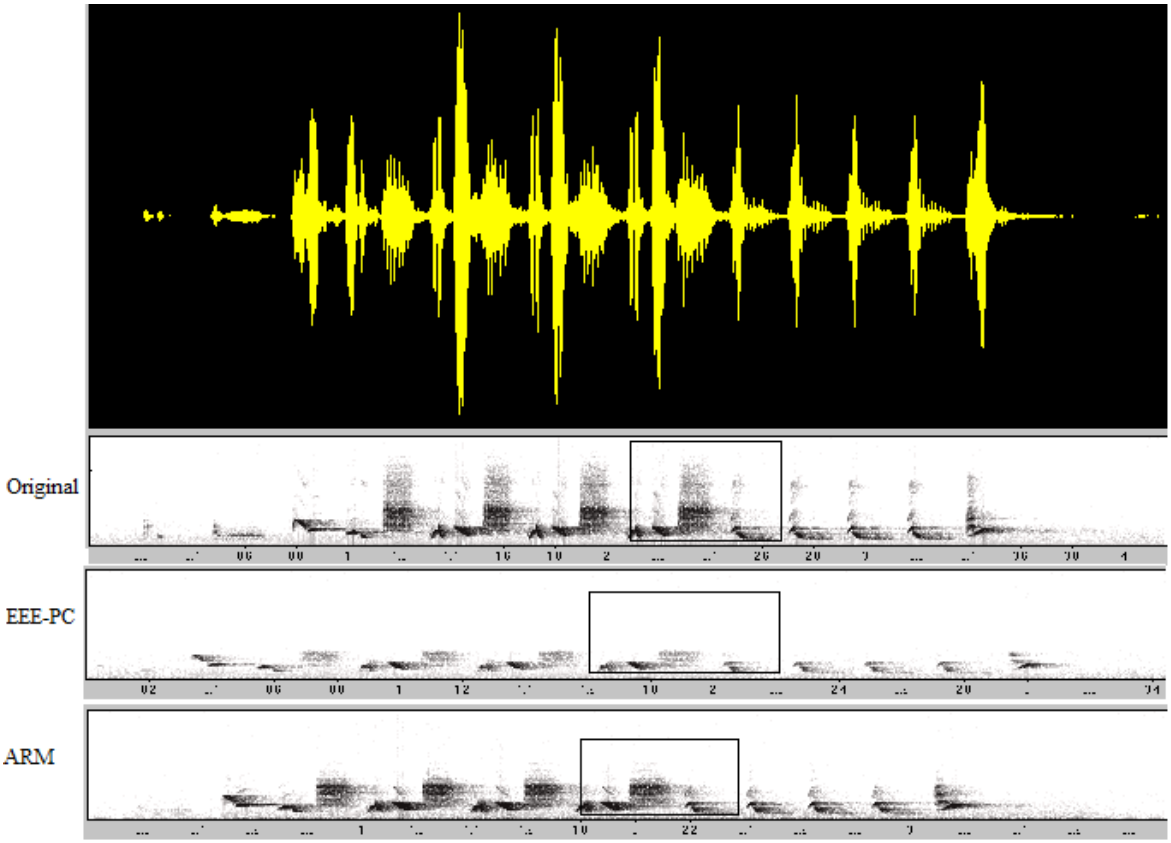


Abbildung 4.10: Vergleich der Qualität einer Strophe mit der Analysesoftware SASLab Pro. Im oberen Bereich sind die Amplituden der Frequenzen (gelb) der Originalaufnahme zu sehen. Darunter werden die vorhandenen Frequenzen der unterschiedlichen Aufnahmeplattformen gegenübergestellt. (Die Zeitachse ist leicht verschoben, deswegen wurde ein fester Ausschnitt ausgewählt.)

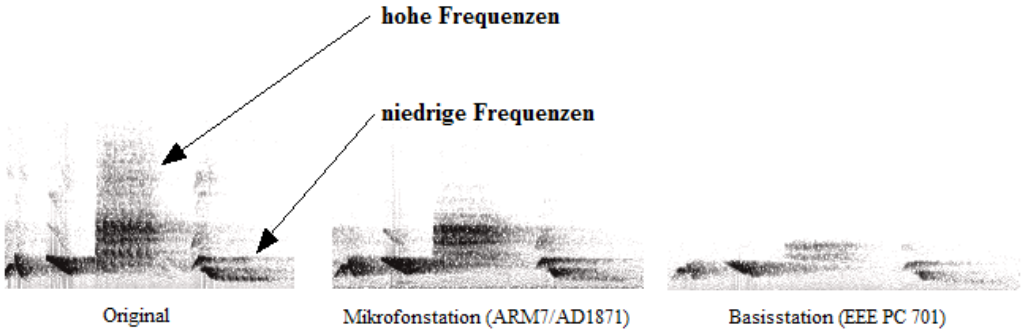


Abbildung 4.11: Im direkten Vergleich enthält die Aufnahme einer Mikrofonstation mehr Informationen für die spätere Analyse und Zuordnung der Strophe als die einer Basisstation und zeigt sogar bis zum Erreichen der Grenzfrequenz des verwendeten Mikrofons ein nahezu identischen Bild gegenüber die Originalaufnahme

4.5. Energieverbrauch

Zum Abschluss der Auswertung soll kurz auf den Energieverbrauch der Mikrofonstation in ihren verschiedenen Zuständen eingegangen werden. Hierzu wurde an zwei Messpunkten die Stromstärke ermittelt: Als erstes zwischen der 12V Batterie und dem externen Span-

nungswandler (MA1) und andererseits nach der Wandlung auf 5V über einen offenen Jumper (MA2, Abbildung 4.12).

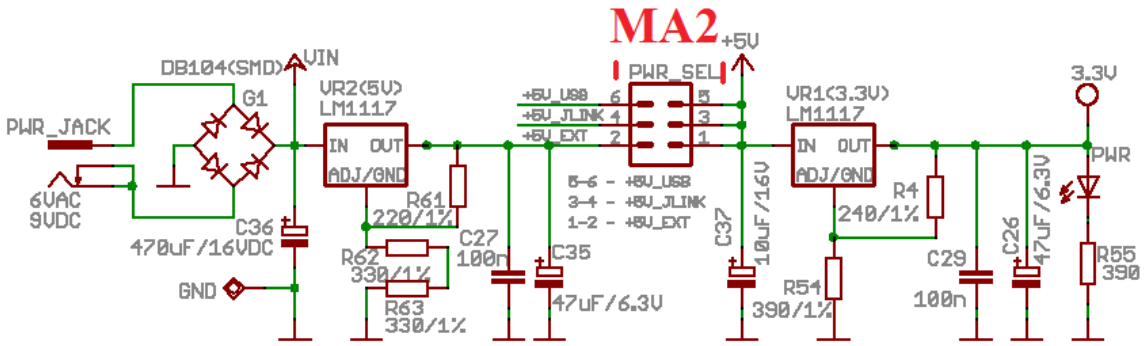


Abbildung 4.12: Spannungswandler DB104, LM1117 und Messpunkt MA2

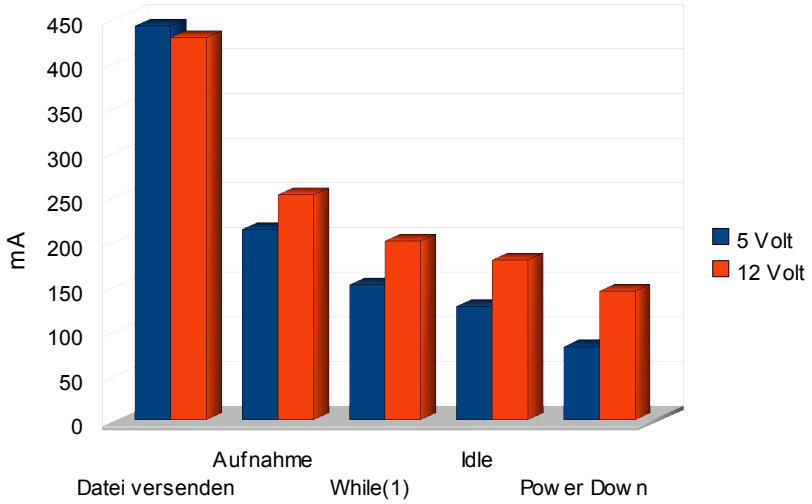


Abbildung 4.13: mittlere Stromstärke, zurückgeführt auf die verschiedenen Zustände. Blau: Messpunkt MA2, Rot: Messpunkt MA1

Die Resultate können aus Abbildung 4.13 und 4.14 entnommen werden. Wie erwartet erfordert das Versenden einer Datei/Zeitplansynchronisation durch die Funkkomponente den höchsten Energiebedarf, gefolgt von einer Aufnahme (aktiver AD-Wandler, Zugriff auf SD-Karte). Testweise wurden – obwohl nicht vollständig implementiert – auch sämtliche Energiesparmodi des Prozessors mit einbezogen, um einen Ausblick auf die spätere Eignung der Plattform zu geben. Überraschend ist hier allerdings selbst im Power Down Modus bei abgeschalteten externen Komponenten eine Stromaufnahme von 82 mA zu verzeichnen. Diese resultiert allein auf der Vielzahl verwendeter Bausteine auf der Evaluationsplatine von Olimex und stellt keinen Mangel der Implementierung der Erweiterungsplatine oder der Softwarearchitektur dar. Zu Gunsten einer hervorragenden Debugging-Fähigkeit und

eines niedrigen Preises hat der Hersteller auf Komponenten mit höherer Energieeffizienz verzichtet. Hierdurch kann die Frage nach einer tragfähigen und in der Zukunft mit der MSB-A2 Plattform wiederverwendbaren Energiequelle für die Stromversorgung des Systems nicht geklärt werden. Eine Neuevaluation sollte nach Umstieg auf die vorgesehene MSB-A2 Plattform stattfinden.

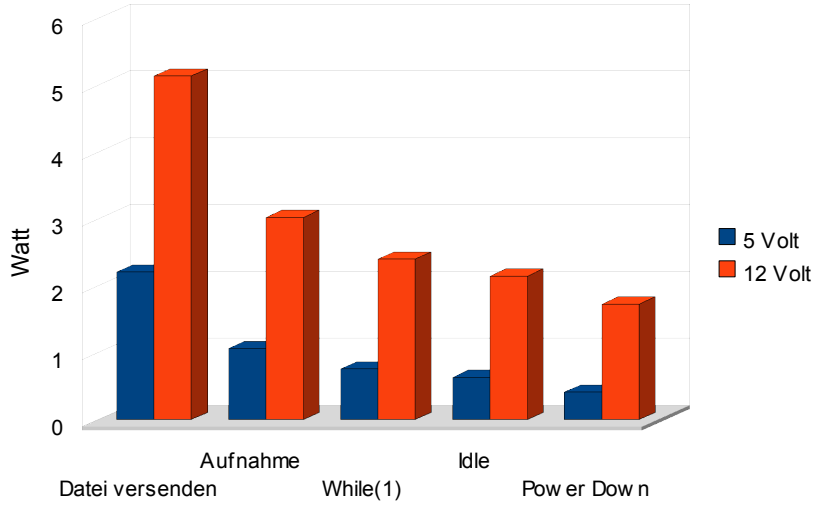


Abbildung 4.15: mittlere Energieaufnahme, zurückgeführt auf die verschiedenen Zustände. Blau: Messpunkt MA2, Rot: Messpunkt MA1

Spannung	5V	12V	5V	12V
Messpunkt	M2	M1	M2	M1
Datei versenden	441 mA	428 mA	2,21 W	5,14 W
Aufnahme	213 mA	252 mA	1,07 W	3,02 W
While(1)	151 mA	200 mA	0,76 W	2,40 W
Idle	127 mA	179 mA	0,64 W	2,15 W
Power Down	82 mA	144 mA	0,41 W	1,73 W

Tabelle 4.2: In Abbildung 4.14 und 4.15 dargestellte Messwerte

5. Zusammenfassung und Ausblick

Die vorliegende Arbeit verbindet den Entwurf und die Implementierung einer eingebetteten Hardwareplattform mit einer mehrschichtigen Softwarearchitektur, welche ihre Funktion in einem interdisziplinären Kontext für die Gesangsaufnahme der Nachtigall unter Beweis stellt. Dabei integrieren sich die hieraus entstandenen Mikrofonstationen nahtlos in eine bereits bestehende Infrastruktur von Basisstationen (x86/Windows basierende Netbooks), welche in der Diplomarbeit von Frank Beier [FB09] entworfen wurde. Das Ziel eine autonom arbeitende Möglichkeit zur Gewinnung hochwertiger Daten zu schaffen, die die weitergehende Analyse komplexer Lernstrategien von Singvögeln vereinfacht, hat eine praktikable und erweiterbare Plattform hervorgebracht, die die Gesamtheit ihrer Komplexität vollständig vor dem eigentlichen Anwender verbirgt. Obwohl die umfangreiche Softwarearchitektur – welche vollständig auf die Verarbeitung eines hohen Datendurchsatzes optimiert wurde – als auch die darauf aufbauende Anwendung eine nahezu vollständige Eigenentwicklung darstellen, hat sich die Implementierung gemäß den Anforderungen als robust erwiesen.

Für den exemplarisch erstellten Prototyp einer Mikrofonstation wurde als Basis ein Mikrocontroller ausgewählt, welcher in ähnlicher Form am Fachbereich bisher hauptsächlich in Low-Power Umgebungen, wie sie zum Beispiel Sensornetze darstellen, Anwendung fand. (vgl. [HW08]) Um den Funktionsumfang zu erweitern, entstand im Rahmen der Diplomarbeit eine Platine, welche die on-Chip Peripherie – unter Nutzung der vorhandenen Schnittstellen des Mikrocontrollers – mit zusätzlichen Komponenten ergänzt. Hierzu zählen ein WLAN-Modul für eingebettete Systeme, ein 16-Bit Sigma-Delta AD-Wandler und eine hieran angepasste Vorverstärkerschaltung zum Anschluss handelsüblicher Mikrofone. Durch die Wahl eines WLAN-Moduls kann nicht nur von einer, gegenüber den sonst üblicherweise verwendeten CC1020/CC1100 [CC11], deutlich höheren Datenrate profitiert werden. Auch wird hierdurch vermieden, dass Modifikationen an der Hardware der Basisstation nötig sind, um eine direkte drahtlose Kommunikation zwischen ihnen und den neuen Mikrofonstationen zu ermöglichen. Generell ist das Kriterium eines hohen Datendurchsatzes ein Teilziel, welches den gesamten Entwurf signifikant geprägt hat. Die gestellte Qualitätsanforderung an verlustfreie Aufnahmen in 16-Bit Auflösung bei 48kHz Abtastrate zieht nach sich, dass das System fähig sein muss 96kByte pro Sekunde an PCM-Audiodaten zu verarbeiten. In der vorliegenden Implementierung konnte sichergestellt werden, dass

hierdurch bedingte Engpässe bei der Zusammenarbeit von AD-Wandler, Datenbus und Speicherkarte während einer Aufnahme ausgeschlossen werden können.

Die in Module unterteilte, mehrschichtige Softwarearchitektur ermöglicht durch ihre klare Struktur eine einfache Wartung – wie sie z.B. beim Austausch von Hardwarekomponenten notwendig werden kann – ohne Seiteneffekte auf fremde Module zu bewirken. Sie enthalten jeweils die Low-Level Ansteuerung der Schnittstelle, eine zusätzliche Kommunikationsschicht welche die Syntax der Hardware umsetzt und darauf aufbauend eine dokumentierte Benutzer-API. Diese fasst häufig benötigte Anwendungsfälle logisch zusammen, integriert eine Fehlerbehandlung und stellt diese als Paket nach außen zur Verfügung. Auf diesen kompakten Befehlssatz kann nun die eigentliche Anwendung zurückgreifen. Diese basiert im Kern auf einem Scheduler, welcher anstehende Aufgaben in einer Prioritätswarteschlange verwaltet und diese zeitgesteuert aufruft. Durch eine Rangliste können höher priorisierte Aufgaben, niedrig priorisierte unterbrechen. Hierdurch konnte erreicht werden, dass der eigentlich serielle Programmablauf verborgen wird und eine vorrangige Ausführung von Aufnahmen garantiert werden kann. Zeitpläne, welche die Start-, Endzeiten und die eindeutige ID einer jeden Aufnahme enthalten, werden in regelmäßigen Abständen mit einer erreichbaren Basisstation abgeglichen. Durch die verwendete Abstraktion ist es möglich geworden, dem eigentlichen Nutzer die zu Grunde liegende und vollständig unterschiedliche Implementierung von Hardware und Software beider Systeme zu verbergen.

5.1. Ausblick

Durch die in großen Teilen sehr hardwarenahe Umsetzung konnten umfangreiche Kenntnisse über das Verhalten und die Grenzen des gezeigten Systems erlangt werden.

Bereits im Entwurf wurde festgelegt, dass die gewählte Evaluationsplatine nur ein Platzhalter für das endgültige Layout des an der Freien Universität Berlin entwickelten *Modular Sensor Board MSB-A2* von Dr. Dipl.-Ing Achim Liers darstellen soll. Portierungsbedingte Änderungen konnten hierdurch bereits im Vorfeld berücksichtigt und minimiert werden. Unabdingbar ist allerdings eine Änderung des Layouts der entstandenen Erweiterungsplatine, da die Anordnung der nach außen geführten Schnittstellen nicht übernommen werden kann.

Langfristig sollte es allerdings vorrangig das Ziel sein, den bisherigen Funktionsumfang der Software stetig zu erweitern. Die Integration von WavPack – eine verlustfreie Kompression, die durch ihre Berechnungen auf Integer-Basis besonders für den Embedded Be-

reich geeignet ist – könnte dazu genutzt werden das Datenaufkommen zu halbieren und idealerweise bereits während der Aufnahme ansetzen, entgegen der bisher verwendeten nachträglichen Kompression.

Der für die Erzielung eines Lerneffekts hervorragend geeignete serielle, aber priorisierte Programmablauf sollte durch ein Betriebssystem ersetzt werden, das dazu in der Lage ist mehrere Prozesse gleichzeitig zu verwalten. So könnten anstatt einer zeitversetzten Versendung, fertiggestellte Dateien bereits parallel zu einer aktuellen Aufnahme übertragen werden, wodurch auch positiv auf die Laufzeit der Basisstationen eingewirkt werden kann. Diese könnten Übertragungen nur in den Zeiträumen erlauben, in denen auch sie aktiv sein müssen (beispielsweise bei eigenen Aufnahmen) und ihnen hierdurch das Recht einräumen, Verbindungen zu Gunsten der Optimierung ihres Energiebedarfs zu beenden. Generell könnten für die Auswahl einer Basisstation als langfristigen Kommunikationspartner zusätzliche Gütekritierien eingeführt werden, welche diese nach ihrer Last, freiem Speicherplatz oder Entfernung (Entscheidung nach Empfangsqualität) einordnet und auswählt.

In Anknüpfung an das bestehende Konzept, welches mit dem Abruf der Aufnahmen durch den Nutzer und der Nachbearbeitung mit Hilfe der Analysesoftware SASLab Pro von Avisoft endet, sollte eine Erkennungssoftware entwickelt werden, welche die Kategorisierung der Strophen der Nachtigall übernimmt und derzeit aufwendig von Hand vorgenommen wird. Aufgrund des zeitlichen Anspruchs an eine solche Aufgabe, sollte diese in einer weiterführenden Arbeit entworfen und implementiert werden.

Literaturverzeichnis

[HIST0]: Jivan S. Parab, Santosh A. Shinde, Vinod G. Shelake, Rajanish K. Kamat, Gourish M. Naik, Practical Aspects of Embedded System Design Using Microcontrollers, Springer-Verlag, 2008

[VB09]: Freie Universität Berlin, Institut für Biologie, Verhaltensbiologie, 2009, http://www.biologie.fu-berlin.de/verhaltensbiologie/forschung/nachtigallenforschung/index.html

[BKLS07]: Michael Baar, Enrico Köppe, Achim Liers, Jochen Schiller, Poster and Abstract: The ScatterWeb MSB-430 Platform for Wireless Sensor Networks, März 2007, SICS Contiki Hands-On Workshop. Kista, Sweden

[BWB08]: Michael Baar, Heiko Will, Bastian Blywis, Thomas Hillebrandt, Achim Liers, Georg Wittenburg, Jochen Schiller, The ScatterWeb MSB-A2 Platform for Wireless Sensor Networks, September 2008, Computer Systems and Telematics, Freie Universität Berlin

[AMVH]: Microsoft Research, Autonomous Monitoring of Vulnerable Habitats, ,

[FB09]: Frank Beier, Autonomous Wildlife Monitoring - Design, implementation, and evaluation of a wireless solution for nightingale song recording, 2009, Freie Universität Berlin

[EMB05]: David J. Katz and Rick Gentile, Embedded Media Processing, Elsevier Newnes, 2005

[UZ05]: Udo Zölzer, Digitale Audiosignalverarbeitung, Teubner Verlag, 2005

[JM09]: Johannes Merx, Psychoakustische Prinzipien, 2009,

[MF08]: Marco Ziegert und Frank Beier, MSB-430H Implementierung eines Audiolinks, 2008,

[MSR05]: Klaus-Dieter Walter, Messen, Steuern und Regeln mit ARM-Mikrocontrollern, Franzis Verlag, 2005

[SOC02]: Steve Furber, ARM Rechnerarchitekturen für System-on-Chip-Design, mitp-Verlag, 2002

[CM306]: Shyam Sadasivan, An Introduction to the Cortex-M3 Processor, October 2006,

[TOS09]: TOSHIBA Electronics Europe GmbH, Embedded ARM® cores from Toshiba, 2009, http://www.toshiba-components.com/ASIC/EmbeddedARMCores.html

[RAI09]: Raisonance, STM32 PRIMER2 Schematic and User manual, 2009,

[ST09]: STMicroelectronics, STM32F103xE Datasheet, 2009,

[LPC87]: NXP, LPC2387 Product data sheet, 2008,

[LPC78]: NXP, LPC2377/78 Product data sheet, 2008,

[NOS08]: Nordic Semiconductor, nRF24LU1 Product Specification, 2008,

[ZIGB07]: Maxstream, Product Manual v1.xAx - 802.15.4 ProtocolFor OEM RF Module Part Numbers: XB24-...-001, XBP24-...-001IEEE® 802.15.4 OEM RF Modules, 2007,

[ZB]: ZigBee Alliance, ZigBee and Wireless Radio Frequency Coexistence, 2007,

[HAUER09]: Hauer, hauer_ewsn20091.pdf, 2009,

[AVI09]: Avisaro AG, Wiki Online Documentation, 2009, http://www.avisaro.com/tl/index.php/docu-home.html

[CC11]: Chipcon TI, CC1100 datasheet, 2009,

[SBBC]: Sangkyu Baek und Bong Dae Choi, PERFORMANCE ANALYSIS OF POWER SAVE MODE INIEEE 802.11 INFRASTRUCTURE WLAN, ,

[FLY08]: Avisaro, WLAN Modul Flyer, 2008,

[MSP02]: Texas Instruments, MSP430x1xx Family User's Guide, 2002, slau049.pdf

[LPC09]: NXP, UM10211 LPC23xx User manual, 2009, user.manual.lpc23xx.pdf

[ST08]: STMicroelectronics, STw5094A datasheet, 2009,

[AD02]: Analog Devices, AD1871 datasheet, 2002,

[ELE41]: Alessandra Wene, Andreas Deml, Arwed Starke, Matthias Wegner, Patrick Schweitzer, Samir Nazzal, Sebastian Feese, Achim Volmer, Projekt Elektronik ELE41 Abschlussbericht, 2007, TU Berlin

[ELV09]: ELV Electronik AG, SMD-Mikrofonvorverstärker Bau- und Bedienungsanleitung, 2009,

[AD09]: Analog Devices, SSM2167 data sheet, 2009,

[E2V09]: e2v Semiconductor, Design considerations for Mixed-Signal: How to Design a PCB-Layout, 2009,

[MT09]: Martin Thomas, ChaN's FAT-Module and LPC23xx/24xx MCI, 2009, http://www.siwawi.arubi.uni-kl.de/avr_projects/arm_projects/arm_memcards/index.html

[ChaN09]: ChaN, FAT File System Module, 2009, http://elm-chan.org/fsw/ff/00index_e.html

[NXPERR09]: NXP, LPC2378 Errata Sheet V1.8, 2009,

[HW08]: Heiko Will, Entwurf und Implementierung einer Multi-Plattform Sensornetz-Architektur, 2008, Freie Universität Berlin

[LB04]: Lutz Bierl, Das große MSP430 Praxisbuch, Franzis Verlag, 2004

[LM06]: National Semiconductor, LM1117 800mA Low-Dropout Linear Regulator, 2006,

[OLI08]: Olimex, LPC-P2378 jumper description , 2008, http://www.olimex.com/dev/lpc-p2378.html

[AVF09]: Avisaro AG, Firmware, 2009, http://www.avisaro.com/tl/releases.html

Begriffserklärung

ADC	Analog-Digital-Wandler
AGC	Automatic Gain Control
ARM	Advanced RISC Machines Synonym verwendet für alle Mikrocontroller mit Prozessorkern der ARM Limited
APB	Advanced Peripheral Bus
CISC	Complex Instruction Set Computer
DMA	Direct-Memory-Access Die direkte Übertragung von Daten zwischen Speicher oder Peripherie ohne die Nutzung von Rechenleistung der Mikrocontrollers
I²S	Spezielle Schnittstelle ausschließlich zur Übertragung von Tondaten
MCU	Microcontroller Unit
MISO	Master-In-Slave-Out Datenleitung bei SPI vom Sender (Slave) zum Empfänger (Master)
MOSI	Master-Out-Slave-In Datenleitung bei SPI vom Sender (Master) zum Empfänger (Slave)
PLL	Phase-Locked Loop
RISC	Reduced Instruction Set Computer
RTC	Real-Time Clock
SCK	Serial Clock Taktsignal, u.a. bei SPI
SSEL	Slave Select Aktiv Low, wählt es den Kommunikationspartner des Masters aus
SPI	Serial Peripheral Interface
SPS	Samples per Second Erfasste Messpunkte pro Sekunde, meist synonym verwendet zu Hz
WSN	Wireless Sensor Network

Anhang

Anwenderhandbuch

Entwicklerplatine – Erstkonfiguration

Diese Schritte beziehen sich auf die Olimex LPC-P2378 Entwicklerplatine:

Die Jumper ISP_E und RST_E (direkt neben der seriellen Schnittstelle) müssen geschlossen sein, um die Firmware später mit dem Tool Flash Magic übertragen zu können. Bei Benutzung einer externen Stromversorgung muss PWR_SEL entsprechend gesetzt sein, dieser befindet sich in der Nähe des Stromanschlusses. Weitere Informationen unter [OLI08].

WLAN-Modul – Erstkonfiguration

Beim Aufstecken des WLAN-Moduls muss unbedingt die Orientierung beachtet werden. Für die Erstkonfiguration muss auf der LPC-P2378 Entwicklerplatine die „Rettungsmodus.hex“ (siehe WLAN-Modul – Rettungsmodus) geflasht sein. Nach diesem Vorgang sollte kurzzeitig die Stromversorgung getrennt werden. Erst danach kann das Webinterface aufgerufen werden: Nach Einwahl in das sichtbare und unverschlüsselte WLAN-Netzwerk „avisaro“, muss in einem beliebigen Browser die Standard-IP Adresse mit der jedes WLAN-Modul ausgeliefert wird aufgerufen werden: „192.168.0.74“. Kann keine Verbindung hergestellt werden, sollte darauf geachtet werden, dass sich die IP der WLAN-Karte des Laptops im selben Subnetz befindet (also im Bereich 192.168.0.1 bis 192.168.0.254, bei Subnetz 255.255.255.0). Die Zugangsdaten sind „admin“ und „1234“. Der erste Schritt sollte nun die Einstellung des „Data Interface“ auf SPI und des „SPI Mode“ auf 3 (siehe Tabelle 1) sein. Bereits hier müssen aus Sicherheitsgründen die Einstellungen über „Submit“ gespeichert werden. Das Modul sollte nun selbstständig neu starten, falls nicht ist noch ein Klick auf „Reboot Device“ nötig. Als nächstes muss unbedingt über den Punkt Firmware mindestens die Version 4.43 von der Webseite [AVF09] des Herstellers eingespielt werden. Erst danach sollten alle anderen Einstellungen aus Tabelle 1 übernommen werden. Über die bisher verwendeten IP-Adressen sollte Buch geführt werden, um keine mehrfach zu vergeben. Außerdem erfolgt die Zuordnung der Mikrofonstationen in der Verwaltungssoftware allein über diese IP-Adresse. War die Konfiguration erfolgreich, kann nun die normale „Nightingale.hex“ auf die Entwicklerplatine geflasht werden. Die Mikrofonstation ist nun einsatzbereit, erkennbar an einer blinkenden LED. Sie sollte sich nun

selbstständig an einer Basisstation anmelden und kann dann nach einem Klick auf „Refresh“ über die Verwaltungssoftware programmiert werden.

Einstellung	Wert
General	
Data Interface	SPI
Network Interface	WLAN
Recovery Mode	Disabled
Scheduling Frequency	0
WLAN	
SSID	voicelink
Mode	Ad-hoc
Channel	1
Encryption	WEP104
WEP Key	766f6963656c696e6b31323334
IP	
WLAN:Local	192.168.1.X
WLAN:Subnet	255.255.255.0
WLAN:Use DHCP	Disabled
TCP Keep Alives	10
SPI	
Mode	3: CPOL=1, CPHA=1
Webserver	
User	admin (Standard)
Password	1234 (Standard)

Tabelle 1: Einstellungen im Webinterface des Avisaro WLAN Modul 2.0, Firmware 4.32

WLAN-Modul – Rettungsmodus

Sollten bei der Eingabe der Einstellungen im Webinterface Fehler unterlaufen und das Modul nun nicht mehr erreichbar sein, z.B. durch Eingabe eines falschen WEP-Schlüssels, wurde zu Wartungszwecken ein Kommandozeilentool implementiert. Über diesen können falsch konfigurierte WLAN-Module wiederhergestellt werden, in sofern wie empfohlen die SPI-Schnittstelle korrekt konfiguriert wurde. Dazu muss die Entwicklerplatine mit dem seriellen Anschluss am USB-JTAG verbunden werden. Über das Tool „Flash Magic“ von NXP wird die „Rettungsmodus.hex“ übertragen, die Einstellungen sind der Abbildung 1 links zu entnehmen. Der COM-Port muss der Angabe im Gerätemanger angepasst werden.

Ist das WLAN-Modul nicht aktiv (keine rote LED) muss nach dem Flashvorgang die gesamte Entwicklerplatine kurz von der Stromversorgung getrennt werden, erst dann können die fehlerhaften Einstellungen über den Terminal von Flash Magic mittels Textkommandos korrigiert werden (Terminal Settings im Screenshot beachten). Die einzelnen Befehle, um das WLAN-Modul wieder erreichen zu können, sind:

IP LOCAL 192.168.0.74

IP MASK 255.255.255.0

IP DHCP OFF

NET WLAN NOBR

WLAN SSID avisaro

WLAN MODE ADHOC

WLAN CHANNEL 1

WLAN SECURITY NONE

RESTART

Das Modul sollte nun neu starten und wieder im Auslieferungszustand sein. Weitere Befehle sind dem Wiki des Herstellers zu entnehmen, zu finden unter [AVI09].

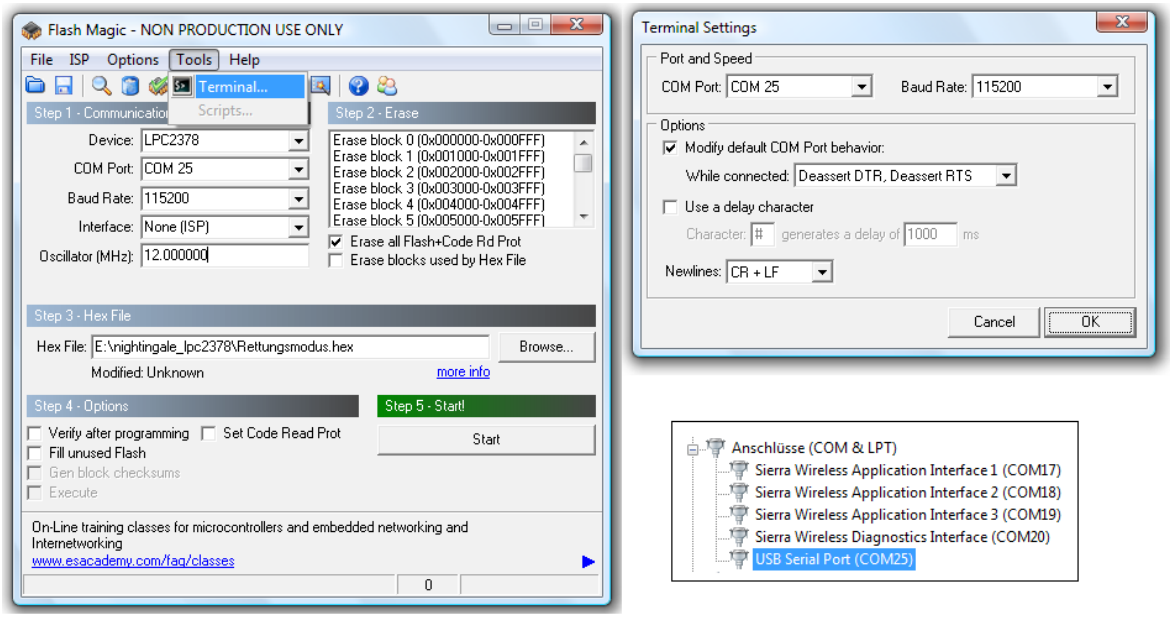


Abbildung 1: Links ist die nötige Konfiguration zur Ansteuerung des LPC2378 zu sehen, der COM-Port kann über den Gerätemanager in Erfahrung gebracht werden (unten rechts). Nach dem Flashvorgang kann im Rettungsmodus über den Terminal das WLAN-Modul programmiert werden, im normalen Betrieb können Debug-Ausgaben eingesehen werden.

Basisstation Konfiguration

Aufgrund eines Fehlers bei der Bestätigung von TCP-Paketen, kann möglicherweise nur alle 200ms ein Paket gesendet werden. Um dieses Problem zu beheben muss in der Windows Registrierung der Wert für TCP-Acknowledge für das WLAN-Interface auf 1 gesetzt werden. Der Standardwert ist hier 2.

Auf der CD ist der TCP-Server für die Basisstation sowohl als Windows-Forms als auch als Windows-Service enthalten. Der Service kann über installutil ARM_Server.exe dauerhaft installiert werden.

Basisstationen müssen sich im Adressbereich 192.168.1.1 bis 192.168.1.10 befinden, ansonsten können diese nicht von den Mikrofonstationen gefunden werden.

Hinweise für Entwickler

Partlist

Exported from Erweiterungsplatine.sch

EAGLE Version 5.4.0 Copyright (c) 1988-2009 CadSoft

Part	Value	Device	Package	Library
AD1871	AD1871	AD1871	SSOP28	analog-devices
C1	100µF	CPOL-EU153CLV-0605	153CLV-0605	rcl
C2	1µF	CPOL-EU153CLV-0405	153CLV-0405	rcl
C6	100nF	C-EUC0805	C0805	rcl
C15	rad 100/25	CPOL-EUE2.5-6	E2,5-6	rcl
C37	rad 10/35	CPOL-EUE2-5	E2-5	rcl
C38	rad 10/35	CPOL-EUE2-5	E2-5	rcl
C39	rad 10/35	CPOL-EUE2-5	E2-5	rcl
C40	rad 10/35	CPOL-EUE2-5	E2-5	rcl
C41	0805 100pF	C-EUC0805	C0805	rcl
C42	0805 100p	C-EUC0805	C0805	rcl
C43	0805 1nF	C-EUC0805	C0805	rcl
C44	0805 1nF	C-EUC0805	C0805	rcl
C45	0805 100pF	C-EUC0805	C0805	rcl

C46	0805 100p	C-EUC0805	C0805	rcl
C47	0805 100n	C-EUC0805	C0805	rcl
C48	rad 10/35	CPOL-EUE2-5	E2-5	rcl
C49	0805 100n	C-EUC0805	C0805	rcl
C50	0805 100n	C-EUC0805	C0805	rcl
C51	rad 100/25	CPOL-EUE2.5-6	E2,5-6	rcl
C52	rad 100/25	CPOL-EUE2.5-6	E2,5-6	rcl
EXT2		PINHD-2X20	2X20	PINHEAD_old
L1	FERRITPERLE	FERRITPERLE	FERRITPERLE	inductors
L2	LQH3C 330µH	L-EUL3230M	L3230M	rcl
LMC6494	LMC6494	LMC6494	SO14	linear
MICBOARD		PINHD-1X4	1X04	pinhead
QG12.288MHZ	FEC 9712518	QG5860	DIL14S	crystal
R1	10k	R-EU_M1206	M1206	rcl
R2	47k	R-EU_M1206	M1206	rcl
R3	47k	R-EU_M1206	M1206	rcl
R4	470k	R-EU_M1206	M1206	rcl
R5	570	R-EU_M1206	M1206	rcl
R10	5k oder 200k	R-TRIMM4G/J	RTRIM4G/J	rcl
R11	1k2	R-EU_M1206	M1206	rcl
R12	1k2	R-EU_M1206	M1206	rcl
R14	0805 100	R-EU_R0805	R0805	rcl
UEXT		PINHD-2X5	2X05	PINHEAD_old
WLANMODU-LE2.0	AVISARO_2.0 APACKAGE1	AVISARO_2.0A-PACKAGE1	AVISARO_2.0	Avisaro2-0