HTL anichstraße
Biomedizintechnik · Elektronik · Elektrotechnik · Maschinenbau · Wirtschaftsingenieure

htl bildung mit zukunft

Diplomarbeit

Bidirektionale Videosprechanlage

Erweiterung einer Raspberry Pi basierten Videogegensprechanlage

Höhere Technische Bundeslehr- und Versuchsanstalt Anichstraße

Abteilung Elektrotechnik

Ausgeführt im Schuljahr 2019/20 von:

Andreas Grain 5AHET
Matthias Mair 5AHET

Betreuer:

DI(FH) Mario Prantl

Innsbruck, am 2020-03-16

Abgabevermerk:

Betreuer:

Datum:

Erklärungen

Eidesstattliche Erklärung

Wir erklären an Eides statt, dass wir die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht haben.

Ort, Datum	Andreas Grain
Ort, Datum	Matthias Mair

Gender-Erklärung

Aus Gründen der besseren Lesbarkeit wird in dieser Diplomarbeit die Sprachform des generischen Maskulinums angewendet. Es wird an dieser Stelle darauf hingewiesen, dass die ausschließliche Verwendung der männlichen Form geschlechtsunabhängig zu verstehen ist.

Inhaltsverzeichnis

Kurzfassung/Abstract

Deutsch

Die vorliegende Diplomarbeit beschäftigt sich mit der Erweiterung einer RPi-basierten Videosprechanlage um eine Smartphone-Applikation, sowie der grundlegenden Überarbeitung der Stations-Hardware. Zusätzlich soll ein Linux-basierter, aus dem Internet erreichbarer Server zum Verteilen der Video-Streams eingerichtet werden.

Dieses Projekt basiert auf einer früheren Diplomarbeit von Sebastian Wagner und Tobias Pfluger an der HTL Anichstraße aus dem Jahr 2017/18, welche ein voll funktionsfähiges Videosprechanlagensystem hervorbrachte. Die verwendete Hardware bietet Verbesserungspotential und wird im Zuge dieser DIplomarbeit weiterentwickelt.

Die Hardware-Erweiterung besteht grundsätzlich aus zwei Teilen: zum einen werden die vielen Elektronik-Bausteine in einer zentralen Platine vereint, was eine einfachere und kostengünstigere Fertigung erlaubt. Zusätzlich wird die aktuelle Hardware der Station um einen Watchdog-Timer erweitert, welcher im Fehlerfall die Anlage zurücksetzt.

Softwareseitig wird die Anlage um eine Smartphone-App ausgebaut, mit welcher der Fernzugriff auf das System ermöglicht werden soll.

English

This diploma thesis deals with the extension of a RPi-based video intercom system by a smartphone application, as well as the fundamental revision of the station hardware. In addition, a Linux-based server, accessible from the Internet, will be set up to distribute the video streams.

This project is based on an earlier diploma thesis by Sebastian Wagner and Tobias Pfluger at the HTL Anichstraße from the year 2017/18, which produced a fully functional video intercom system. The hardware used offers potential for improvement and will be further developed in the course of this diploma thesis.

The hardware extension basically consists of two parts: on the one hand, the many electronic components are combined in a central circuit board, which allows easier and more cost-effective production. In addition, the current hardware of the station is extended by a watchdog timer, which resets the system in case of an error.

On the software side, the system will be extended by a smartphone app, which will enable remote access to the system.

Projektergebnis

Im Zuge dieser Diplomarbeit wurden folgende Software-Teile programmiert:

- Oberfläche der Applikation (PiBell/MainPage.xaml)
- Klassen zur Verwaltung der Dienste (PiBell/Classes/MediaClasses.cs)
- Berechtigungsmanagement (PiBell.Android/MainActivity.cs)
- Audio-Aufnahme-Service in Android (PiBell.Android/AudioRecordingService.cs)
- Audio-Aufnahme-Service in iOS (PiBell.iOS/AudioRecordingService.cs)

Abbildung 1: App-Ansichten

Aus diesen Software-Teilen wurde eine fertige Android-Applikation erstellt, mit der der Fernzugriff auf die fest installierte Videosprechanlage jederzeit möglich ist. Die Kommunikation wurde bidirektional für die Ton- und unidirektional für die Video-Übertragung realisiert. Die Applikation wurde auf verschiedenen Systemen simuliert und in geringem

Ausmaß getestet. Eine iOS-Version wurde aufgrund fehlender Endgeräte nicht fertiggestellt bzw. getestet.

Die programmierte Applikation ist in diesem Dokument ausführlich beschrieben und wesentliche Funktionen im Detail erklärt. Die Erläuterung des Programm-Codes ist kein normrelevantes Regelwerk, sondern ein Kommentar wie es der Autor Andreas Grain versteht! Im Zweifelsfall wird darauf hingewiesen, dass die offiziellen Angaben der Hersteller zu verwenden sind. Auf diese wird an relevanten Stellen verwiesen.

Verwendete Technologien und Software:

- RTSP zur Übertragung der Mediendaten über das Netzwerk
- LibVLC Sharp zum Abrufen und Darstellen des Video-Streams
- GStreamer zur Weiterverarbeitung der Mediendaten am Server
- Live555 Proxy zum Verteilen des RTSP-Streams
- Push Benachrichtigung zur Kontaktaufnahme mit dem Benutzer der App
- Visual Studio 2019 als Entwicklungsumgebung
- Xamarin-Rahmenwerk für die Entwicklung der Multiplattform-App
- XeLaTeX zur Erstellung der Dokumentation
- OneNote zur Projektplanung

1 Einleitung

Jedes moderne Wohngebäude ist inzwischen mit einer Gegensprechanlage ausgestattet. Eine solche Anlage erlaubt es dem Wohnungsbesitzer mit jemandem vor der Haustür zu reden, bevor dieser hereingelassen wird. Manche Wohnungen besitzen bereits eine Videosprechanlage, die dem Bewohner nicht nur den Ton, sondern zusätzlich ein Video von der Außenstelle bereitstellt. Dem Gast wird jedoch immer noch nur der Ton von innen übertragen. Diese Applikation ist ortsgebunden, d.h. es müssen sowohl der Gast, als auch der Bewohner vor der entsprechenden Station stehen.

Im Zuge einer früheren Diplomarbeit des Schuljahres 2017/18 an der HTL Anichstraße entwickelten Sebastian Wagner und Tobias Pfluger eine Videogegensprechanlage, welche mehrere Innenstationen und die bidirektionale Video- und Tonübertragung unterstützt. Die Idee ist, dass in einem Wohnungskomplex pro Partei eine Innenstation verbaut wird, sowie eine Außenstation am Hauseingang. Die einzelnen Innenstationen, also Wohnungsparteien, können dann nicht nur mit der Außenstation, sondern auch mit anderen Innenstationen per Video und Ton kommunizieren. Die Stationen wurden mit Hilfe eines RPi realisiert, um die Kosten gering zu halten.

Am Anfang des 5. Schuljahres im Herbst 2019 wurde uns von unserem FI-Professor DI(FH) Mario Prantl angeboten, dieses Projekt durch eine Smartphone-App und eine Hardware-Überarbeitung zu verbessern. Für die Entwicklung der App sollte das Xamarin-Rahmenwerk verwendet werden, damit die Applikation auf sowohl Android-, als auch iOS-Geräten lauffähig ist. Der Benutzer soll über die App das Video der entsprechenden Station abrufen und mit dem Gesprächspartner sprechen können. Hierfür wird ein aus dem Internet erreichbarer Medien-Verwaltungsserver benötigt, der die Medien-Streams der Einzelstationen empfängt und an die Station des Gesprächspartners überträgt. Die Hardware-Überarbeitung hat mehrere Ziele; zum einen soll dadurch die Komplexität des internen Aufbaus einer Station verringert werden. Zum anderen soll mit Hilfe eines Watchdog-Timers das Langzeit-Betriebsverhalten verbessert werden.

1.1 Funktionalität

Ortsunabhängiger Anlagenzugriff: Um dem Anwender mehr Komfort bieten zu können, beschränkt sich die Anlage und dessen Funktion nicht mehr auf die fest installierte

Station, sondern ist auch über ein mobiles Gerät bedienbar. Dies bietet den Vorteil, dass auch wenn man sich gerade nicht in der Nähe der Innenstation befindet, mit Besuchern kommuniziert werden kann.

Paketdienst: Die Paketübergabe erfolgt üblicherweise persönlich. Wenn der Empfänger nicht zuhause ist, nimmt der Paketzusteller die Ware wieder mit, deponiert sie bei einer Paketabholstelle und hinterlässt dem Empfänger lediglich eine Benachrichtigung. Diese Vorgangsweise ist besonders ärgerlich, wenn man den Paketzusteller um wenige Minuten versäumt hat oder nicht schnell genug zur Innenstelle gelangen konnte. Beispielsweise befindet sich der Bewohner gerade auf dem Dachboden oder im Keller. Mit einer Smartphone-App ist eine sofortige Kontaktaufnahme mit dem Paketzusteller möglich und verhindert eine unnötige zusätzliche Bearbeitung des Paketversandes.

Sicherheit: Die neu gewonnene Ortsunabhängigkeit der Anlage bietet eine höhere Einbruchssicherheit, da man jederzeit Überblick über erwünschte bzw. unerwünschte Besucher hat. Über die Smartphone-Applikation ist das Öffnen des Türschlosses aus mehreren Gründen nicht möglich:

- Wenn der Benutzer nicht zuhause ist, ist eine Türöffnung in den allermeisten Fällen nicht erwünscht.
- Wenn der Benutzer zuhause ist, muss er sowieso zur Eingangstür gehen, um den Besuch in Empfang zu nehmen. Die im Eingangsbereich installierte Station dient hier zur Öffnung der Haustür.
- Falls unbefugter Zugriff auf die Applikation erfolgen sollte, besteht kein besonderes Sicherheitsrisiko hinsichtlich Einbruchs in die Wohnung.

1.2 Aufgabenteilung

Andreas Grain ist der Projekt-Hauptverantwortliche und zuständig für die App-Programmierung. Dies beinhaltet die Einarbeitung in und Verwendung des Xamarin-Rahmenwerks zur Erstellung einer Multiplattform-Smartphone-Applikation. Diese soll den Video-Stream der gewählten Station abrufen und anzeigen, sowie den Mikrofon-Ton des Mobilgerätes zur Anlage zurückschicken. Weiters ist er für die Einarbeitung in das GStreamer-Rahmenwerk zur zentralen Verarbeitung der Video- und Audio-Daten der Stationen und Mobilgeräte über einen Linux-basierten Server zuständig. Darüber

hinaus organisiert Andreas Grain sämtliche Treffen mit dem Betreuer, achtet auf die Einhaltung des Zeitplans und ist zuständig für das Erstellen des XELATEX-Dokuments

Matthias Mair ist verantwortlich für die Hardware-Überarbeitung der Station. Diese beinhaltet die Einarbeitung in das PCB-Entwicklungsprogramm EAGLE von der Firma Autodesk mit der Version 9.5.2, sowie das Recherchieren und Auswählen möglicher Verstärker-Schaltungen und -ICs für die Mikrofon- und Lautsprecher-Verstärkung. Im Zuge der Hardware-Überarbeitung sollen die vielen Einzelmodule in einer übersichtlichen Platine vereint werden. Für die Leiterplatte soll die weit verbreitete SMD-Technologie verwendet werden, da diese heutzutage Marktstandard für integrierte Geräte ist und im Vergleich zu traditionellen THT-Platinen viel platzsparender ist. Zusätzlich wird die Station um einen Watchdog-Timer erweitert, den Matthias Mair entwerfen und die dazugehörige Software schreibt.

Teil I

Software - Andreas Grain

2 Rahmenwerk

2.1 Anforderungen

Das Rahmenwerk dient der Vereinfachung des Entwicklungsprozesses. Je nach Anwendung sind von diesem verschiedene Anforderungen zu erfüllen, daher ist die Wahl des richtigen Rahmenwerks besonders wichtig. Dieser Abschnitt beschäftigt sich mit dem Vergleich der Möglichkeiten, sowie einer endgültigen Auswahl eines der Rahmenwerke zur Verwendung im Diplomprojekt. Die geforderten Funktionen für dieses Projekt beinhalten:

- eine einfache Entwicklung der Multiplattform-App. Wenn möglich soll nur eine App geschrieben werden, die dann auf allen Zielplattformen (Android, iOS) lauffähig ist.
- Die Programmierung sollte in einer bereits bekannten Programmiersprache möglich sein. Für die Fachrichtung Elektrotechnik der HTL Anichstraße werden momentan die Sprachen C, C++ und C#, in untergeordnetem Ausmaß auch JavaScript für Webseiten-Entwicklung unterrichtet.
- Das Rahmenwerk und die dazugehörige Entwicklungsumgebung sollten für nichtkommerzielle Anwendungen wie diese Diplomarbeit frei zur Verfügung stehen.
- Die Entwicklungswerkzeuge sollten das Windows-, optional das Linux-Betriebssystem unterstützen.

2.2 Vergleich

Für die App-Programmierung stehen viele verschiedene Rahmenwerke bereits zur Verfügung.

2.2.1 Microsoft Xamarin

Hierbei handelt es sich um ein Rahmenwerk, mit dem man Multiplattform-Applikationen für unter anderem Android, iOS, UWP und noch viele weitere Plattformen erstellen kann. Es basiert auf Microsofts .NET und dem Mono-Projekt, welches sich als Ziel gesetzt

hat, das .NET auf andere Plattformen zu portieren. Xamarin bietet mehrere Projekttypen an, darunter Xamarin.Android und Xamarin.iOS für native App-Entwicklung und Xamarin.Forms für großteils plattformunabhängige Entwicklung. Eine Xamarin.Forms-Solution besteht daher aus mehreren Teilen:

- Portable/.NET-Standard Projekt, das den plattformunabhängigen Code beinhaltet
- natives Xamarin-Projekt für jede Zielplattform

Der Vorteil Xamarins ist, dass der Großteil des geschriebenen Codes im plattformunabhängigen Projekt bleibt und nur für wenige Funktionen auf native Programmierung zurückgegriffen wird, wie zum Beispiel für hardwarenahe Audio-Aufnahme. Außerdem ist das Xamarin-Projekt Open-Source, das bedeutet jeder kann den Quellcode betrachten und unter Umständen Verbesserungen vorbringen. Zusätzlich ist es für diese Anwendung kostenfrei.

2.2.2 Webtechnologiebasierte Rahmenwerke

In dieser Kategorie existieren sehr viele verschiedene Rahmenwerke, darunter Apache Cordova, Adobe PhoneGap, sowie Facebook React Native. Am Beispiel von Apache Cordova werden hier die Vor- und Nachteile dieses Ansatzes erläutert.

Cordova ist ein App-Entwicklungs-Rahmenwerk welches ursprünglich von der Firma Nitobi unter dem Name PhoneGap entwickelt wurde. Im Jahr 2011 wurde Nitobi von Adobe aufgekauft. Später wurde neben der Closed-Source-Version auch eine quelloffene Version unter dem Namen Cordova veröffentlicht. [vgl. 1]

Apps werden mittels gängiger Webtechnologie, wie zum Beispiel JavaScript, CSS und ähnlichen, entwickelt und realisiert, weshalb das Rahmenwerk alle gängigen Plattformen unterstützt. Der Vorteil von Cordova und ähnlichen Rahmenwerken liegt im enormen Support dieser Webtechnologien, jedoch sind Sprachen wie JavaScript für Back-End-Programmierung weniger geeignet. Noch dazu kommt, dass an der HTL Anichstraße großteils die Programmierung nur in C und C# gelehrt wird, weshalb die Entwicklung mit JavaScript im vereinbarten Zeitrahmen der Diplomarbeit nicht realistisch umsetzbar ist.

Aufgrund der oben erläuterten Vor- und Nachteile der möglichen Ansätze wurde die Verwendung von Xamarin für diese Diplomarbeit festgelegt.

Function	Community	Professional	Enterprise
Supported Usage Scenarios			
Enterprise		Yes	Yes
IDE			
Live Dependency Validation			Yes
Architectural Layer Diagrams			Yes
Architecture Validation			Yes
Code Clone			Yes
CodeLens	Partial	Yes	Yes
Advanced Debugging And Diagnostics			
IntelliTrace			Yes
Code Map Debugger Integration			Yes
.NET Memory Dump Analysis			Yes
Snapshot Debugger			Yes
Time Travel Debugging (Preview)			Yes
Testing Tools			
Live Unit Testing			Yes
IntelliTest			Yes
Microsoft Fakes (Unit Test Isolation)			Yes
Code Coverage			Yes
Cross-platform Development			
Embedded Assemblies			Yes
Xamarin Inspector			Yes
Xamarin Profiler			Yes

Tabelle 2.1: Unterschiede Visual-Studio-Editionen [vgl. 2]

2.3 Build-Umgebung

2.3.1 Visual Studio 2019

Aufgrund obiger Aufstellung wurde Visual Studio 2019 der Firma Microsoft als Entwicklungsumgebung gewählt. Visual Studio ist das offizielle Werkzeug für die Programmierung mit dem Xamarin-Rahmenwerk und die wahrscheinlich bestbekannte Entwicklungsumgebung überhaupt. Die Community-Edition für Schüler und Private steht gratis zum Download zur Verfügung. Diese beinhält alle wichtigen Entwicklungswerkzeuge wie zum Beispiel die automatische Code-Vervollständigung. Für die Nutzung wird nach den ersten 30 Tagen ein Microsoft-Konto benötigt.

Neben der Community-Edition existieren auch noch die Professional- und die Enterprise-Edition. Diese unterstützen zusätzliche Funktionen an:

Der Programmcode wird von Visual Studio 2019 in sogenannten Solutions organisiert. Am Beispiel einer Xamarin.Forms-Applikation lässt sich das sehr gut erläutern; die Solution enthält auf oberster Ebene drei Projekte: das portable Projekt, das Android- und das iOS-Projekt. Ein Projekt kann bereits für sich lauffähig oder nur Teil eines größeren Programms sein.

Visual Studio 2019 unterstützt viele verschiedene Einsatzbereiche, die bei der Installation ausgewählt werden müssen. Eine volle Installation mit allen Funktionen und Projektarten ist zwar möglich, benötigt allerdings bis zu 210GB Speicherplatz des Systems. Eine typische Xamarin-Installation benötigt etwa 6.25GB Festplattenspeicher. Für eine solche Installation muss im Visual Studio Installer das „Mobile development with .NET“-Paket ausgewählt und installiert werden.

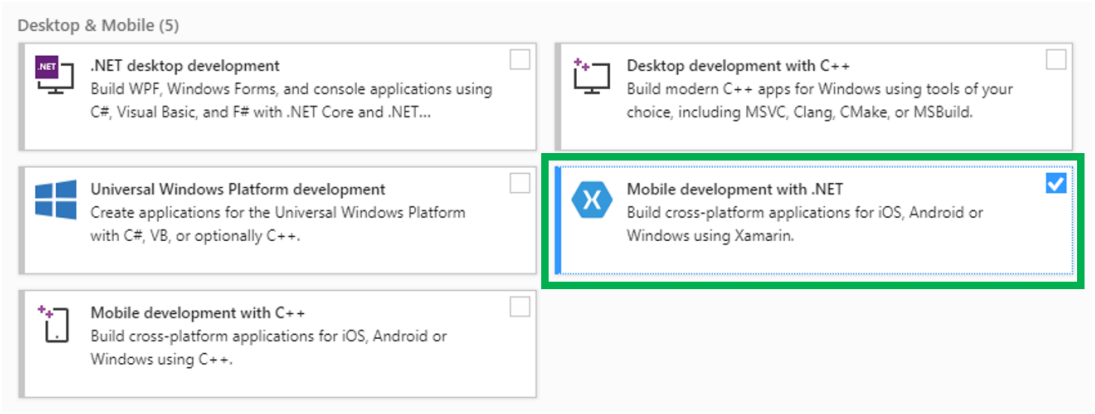


Abbildung 2.1: Auswahl des Xamarin-Rahmenwerks bei der Installation

Es empfiehlt sich, zusätzlich das Paket für „.NET desktop development“ zu installieren, damit neue, noch nicht bekannte .NET-Technologien in einer einfacheren Konsolen-Anwendung ausprobiert werden können. Das Paket benötigt ungefähr weitere 0.8GB.

Auf nähere Details der Installation von Visual Studio wird hier nicht eingegangen, diese können aber auf der Microsoft-Dokumentationsseite [vgl. 3] nachgeschlagen werden.

2.3.2 Android SDK

Um Xamarin-Applikationen für Android-Geräte entwickeln zu können wird das von Google bereitgestellte Android SDK benötigt, um das Programm in ein APK umzuwandeln. Die einzelnen Komponenten können im Android-SDK-Manager des Visual Studio

installiert werden. Dieser ist im Menüband unter „Tools\Android\Android SDK Manager“ zu finden.

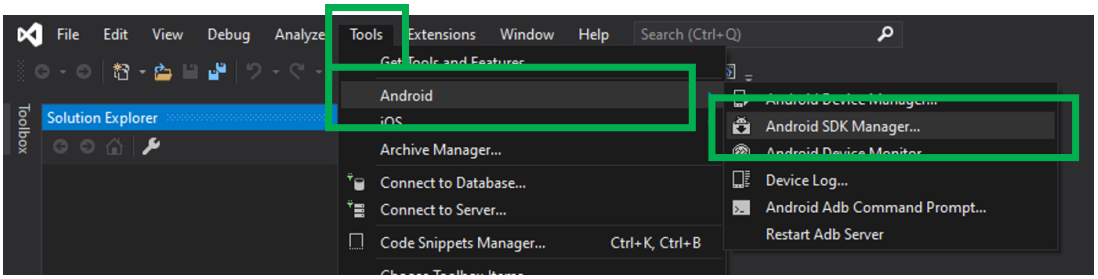


Abbildung 2.2: Android SDK Manager

Der SDK-Manager ist in zwei Seiten unterteilt - „Platforms“ und „Tools“.

Auf der „Platforms“-Seite können die benötigten SDK-Teile für jede Android-Version installiert werden. Benötigt wird nur die Version, mit der das Projekt später kompiliert werden soll. Außerdem kann man auf dieser Seite des SDK-Managers System-Abbilder für den Android Emulator installieren.

Über die „Tools“-Seite können verschiedenste Entwicklungswerkzeuge installiert werden. Grundsätzlich erforderlich sind „Android SDK Tools“, „Android SDK Platform Tools“ und „Android SDK Build Tools“, um Android-Apps mit Xamarin entwickeln zu können. Für dieses Projekt werden Push-Benachrichtigungen via Google Cloud Messaging verwendet, daher werden noch zusätzlich die Pakete für die „Google Play Services“ sowie die „Google Cloud Messaging for Android Library“ installiert. Wie Push-Benachrichtigungen genauer funktionieren, wird später im Kapitel *4 Push-Benachrichtigung* auf Seite 23 behandelt.

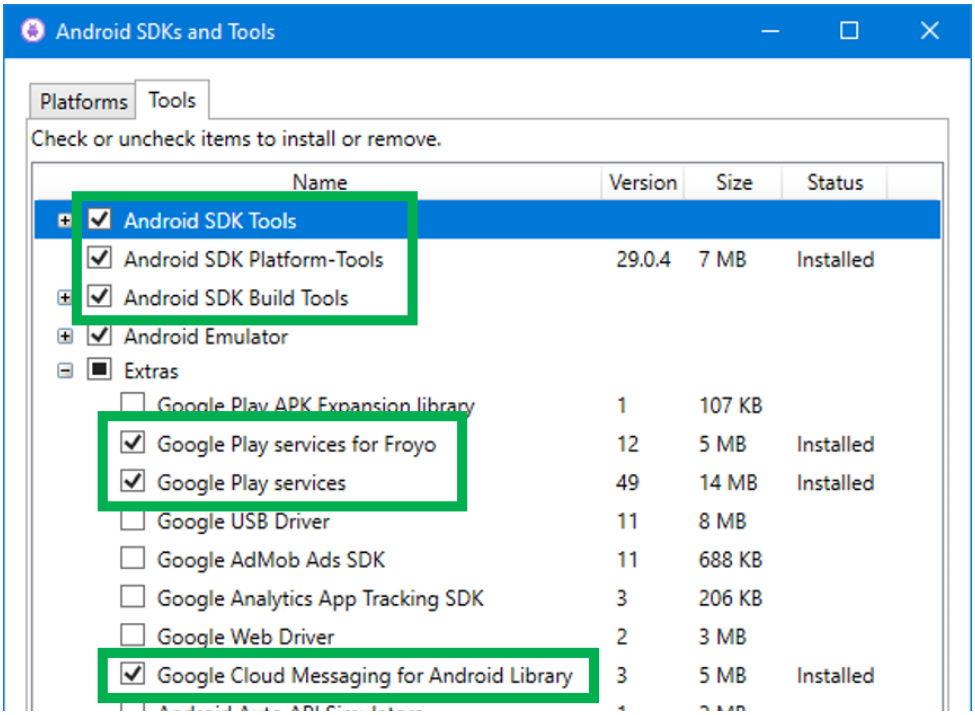


Abbildung 2.3: Notwendige Teile des Android SDK

2.3.3 Build-Vorgang

Um den geschriebenen Code in eine lauffähige Applikation umzuwandeln, wird ein sogenannter Compiler benötigt. Die Aufgabe des Compilers ist es, die geschriebene Hochsprache entweder direkt in CPU-Maschinensprache bzw. eine Art Zwischencode zu kompilieren, d.h. umzuwandeln. Im Fall einer einfachen C#-Konsolenanwendung ist der Build-Vorgang noch relativ simpel:

1. Alle einzelnen Code-Dateien werden nacheinander vom Compiler in CIL, eine Zwischensprache des .NET, umgewandelt.

2. Der Linker fügt die einzelnen Teile zu einem Gesamten zusammen. Das Resultat ist eine Datei, die wie eine ausführbare .EXE-Datei erscheint.

3. Wenn das zusammengefügte Programm gestartet wird, kompiliert die CLR den Zwischencode in einen direkt ausführbaren Maschinen-Code. Die CPU des Systems kann diesen dann ausführen.

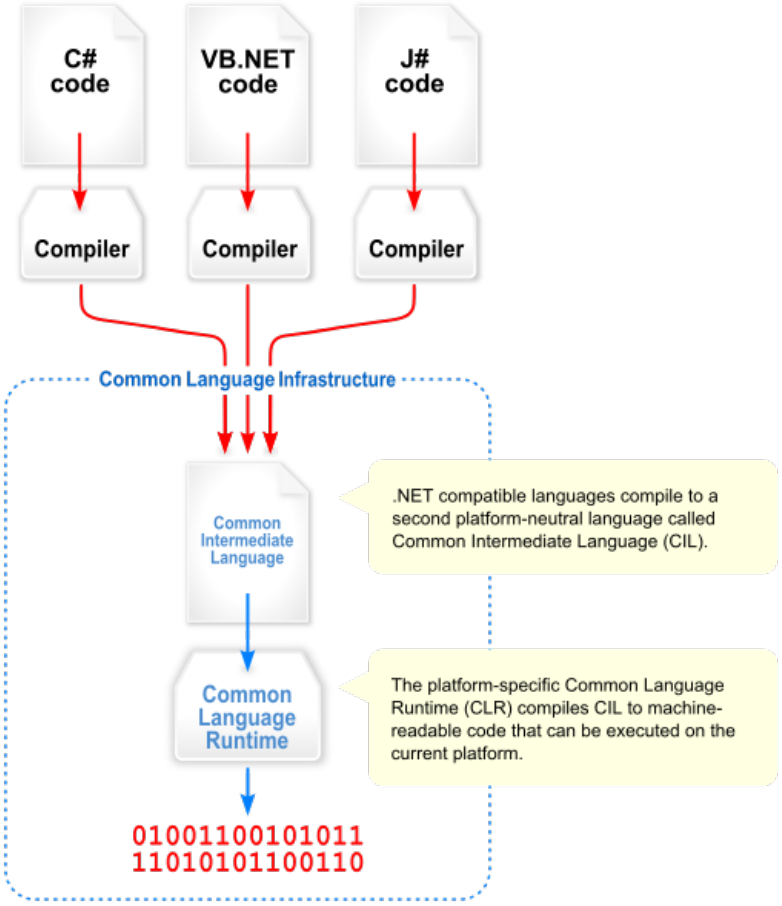


Abbildung 2.4: Build-Vorgang einer .NET-Applikation [4]

Ähnlich funktioniert dieser Vorgang auch bei einer Xamarin.Forms-App, allerdings wesentlich komplexer. Eine Xamarin.Forms-Solution besteht aus drei Einzel-Projekten (PiBell, PiBell.Android, PiBell.iOS), wobei das .NET-Standard-Projekt als Teil der anderen zwei verbaut werden muss. Dadurch ergibt sich ein in Abbildung *2.5 Build-Vorgang einer Xamarin.Forms-Solution* dargestellter Build-Vorgang.

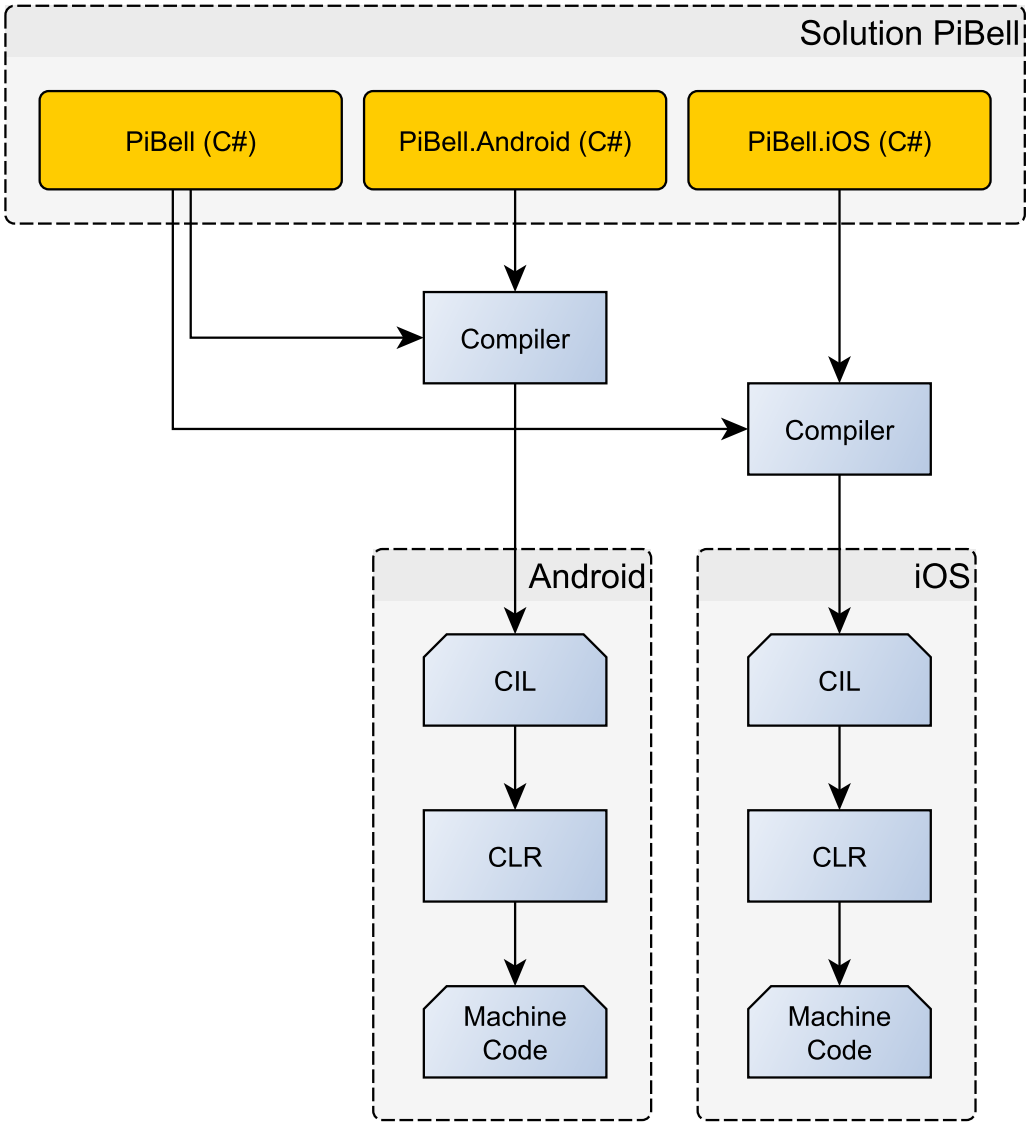


Abbildung 2.5: Build-Vorgang einer Xamarin.Forms-Solution

3 Verwendete Software-Module

3.1 Real Time Streaming Protocol

RTSP ist ein Netzwerkprotokoll zum Aufbauen und Verwalten von Netzwerkverbindungen zur Übertragung kontinuierlicher Medien-Daten (Streams). Dieses Netzwerkprotokoll ist im IETF-Dokument *RFC 2326 - Real Time Streaming Protocol (RTSP)* festgelegt. Es wurde gemeinsam von H. Schulzrinne (Columbia University), A. Rao (Netscape Communications) und R. Lanphier (RealNetworks) entwickelt. [vgl. 5]

3.1.1 RTSP Version 2

Im Zuge dieser Diplomarbeit wird die erste Version des RTSP-Protokolls verwendet. Neben der verwendeten Version 1 existiert auch eine neue Version 2, welche im IETF-Dokument *RFC 7826 - Real-Time Streaming Protocol Version 2.0* definiert ist. Die neue Version wurde von H. Schulzrinne (Columbia University), A. Rao (Cisco Networks), R Lanphier u. a. erarbeitet. [vgl. 6]

Es wurde dennoch die erste Version verwendet, da die Stationen dieses bereits verwenden und die neue Version nicht rückwärtskompatibel ist.

RTSP wird in der Industrie vielseitig verwendet, um Live-Übertragungen von Überwachungskameras zu verwalten. Eine weitere Anwendung des RTSP Protokolls ist das Streamen von Medien-Daten zu einem Server, der diese dann beispielsweise transcodiert oder aufnimmt.

3.1.2 Anfragen-Aufbau

Das RTSP-Protokoll definiert mehrere Anfragen zur Verwaltung der Netzwerkverbindung. Diese Anfragen werden, ähnlich wie beim HTTP, in unverschlüsseltem Plain-Text verschickt. In Tabelle 3.1 sind sämtliche Anfragen des RTSP-Protokolls und deren Übertragungsrichtung aufgelistet. Die in Tabelle 3.1 verwendeten Kurzbezeichnungen haben folgende Bedeutung:

Richtung: C…Client, S…Server

Objekt: P…Präsentation, S…Stream

Methode	Richtung	Objekt	Verbindlichkeit
DESCRIBE	C→S	P,S	empfohlen
ANNOUNCE	C→S, S→C	P,S	optional
GET_PARAMETER	C→S, S→C	P,S	optional
OPTIONS	C→S, S→C	P,S	erfordert, (S→C: optional)
PAUSE	C→S	P,S	empfohlen
PLAY	C→S	P,S	erfordert
RECORD	C→S	P,S	optional
REDIRECT	S→C	P,S	optional
SETUP	C→S	S	erfordert
SET_PARAMETER	C→S, S→C	P,S	optional
TEARDOWN	C→S	P,S	erfordert

Tabelle 3.1: Anfrage-Arten des RTSP-Protokolls

Das Aussehen einer solchen Anfrage wird anhand des Beispiels der DESCRIBE-Methode veranschaulicht. Der Client beginnt die Anfrage mit dem URL des Medien-Streams, den er abrufen möchte und teilt dem Server mit, welche Formate er versteht.

```
1 C->S: DESCRIBE rtsp://server.example.com/fizzle/foo RTSP/1.0
2     CSeq: 312
3     Accept: application/sdp, application/rtsl, application/mheg
```

Der Server antwortet auf diese Anfrage mit dem Session Descriptor, der alle wichtigen Informationen des Medien-Streams beinhaltet, wie zum Beispiel Video- und Audio-Format, Übertragungsmethode, sowie allgemeine Informationen wie Titel und Beschreibung.

```
1  S->C: RTSP/1.0 200 OK
2      CSeq: 312
3      Date: 23 Jan 1997 15:35:06 GMT
4      Content-Type: application/sdp
5      Content-Length: 376
6
7      v=0
8      o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
9      s=SDP Seminar
10     i=A Seminar on the session description protocol
```

```
11      u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
12      e=mjh@isi.edu (Mark Handley)
13      c=IN IP4 224.2.17.12/127
14      t=2873397496 2873404696
15      a=recvonly
16      m=audio 3456 RTP/AVP 0
17      m=video 2232 RTP/AVP 31
18      m=whiteboard 32416 UDP WB
19      a=orient:portrait
```

3.2 LibVLC Sharp

LibVLC ist ein Multimedia-Rahmenwerk der Non-Profit-Organisation VideoLAN, welche durch den VLC Media Player und dessen Funktionsumfang bekannt wurde. Der VLC Media Player funktioniert sozusagen als grafische Oberfläche zur LibVLC-Programmbibliothek. Die Bibliothek beinhält viele nützliche Funktionen, darunter die folgenden:

- Videos nahezu beliebigen Formates decodieren und darstellen
- Video und Audio aufnehmen und transcodieren für die weitere Verwendung
- Live-Streaming von Video-Daten, Webcam-Bild und Mikrofon-Ton.

LibVLC ist eine Bibliothek für die C-Sprache. Um diese in einem C#-Projekt verwenden zu können sind Bindings notwendig, welche Zugriff auf die C-Bibliothek geben. LibVLC wurde von der Non-Profit-Organisation VideoLAN entwickelt und erstmals am 1. Februar 2001 veröffentlicht. [vgl. 7]

Im Xamarin.Forms-Projekt wird die zum Zeitpunkt der App-Entwicklung aktuelle Version 3.4.2 der LibVLC Sharp Bindings verwendet. Bindings erlauben es dem Programmierer, auf Programmbibliotheken zuzugreifen, welche für eine andere Sprache geschrieben wurden. Man kann sich Bindings wie Kleber vorstellen, der die Funktionen der einen Programmiersprache mit denen der anderen Sprache verbindet. Die Version 3.4.2 ist inzwischen nicht mehr die neueste Version, jedoch ist sie in Hinsicht auf Funktionalität deckungsgleich mit der neuesten Version.

3.2.1 Erwähnenswerte Funktionen

Die LibVLC-Sharp-Bindings ermöglichen die Verwendung des MVVM-Modells, welches vom WPF- und Silverlight-Architekten John Gossman im Jahr 2005 auf seinem Blog vorgestellt wurde. [vlg. 8, The Evolution of Model-View-ViewModel] Die Philosophie hinter diesem Modell ist, die Oberfläche (View) bis auf wenige Schnittstellen komplett vom eigentlichen Code zu trennen. Dies ermöglicht es ohne größeren Aufwand, die Oberfläche zu ändern oder gar auszutauschen. Folgende Terminologie wird in diesem Zusammenhang verwendet:

- Models … beinhalten die Programmdaten. Meist werden einfache Klassen oder Strukturen hierfür verwendet.
- Views … Die grafische Oberfläche und deren Elemente wie Buttons, TextBoxen, …
- ViewModels … wandeln die Daten der Models so um, dass sie von der Oberfläche dargestellt werden können. ViewModels sind meist als Klasse ausgeführt.

Die Daten der Oberfläche und die Daten des Models bleiben zueinander konsistent. Sobald sich ein Wert auf einer Seite ändert, wird er sofort auf der anderen aktualisiert.

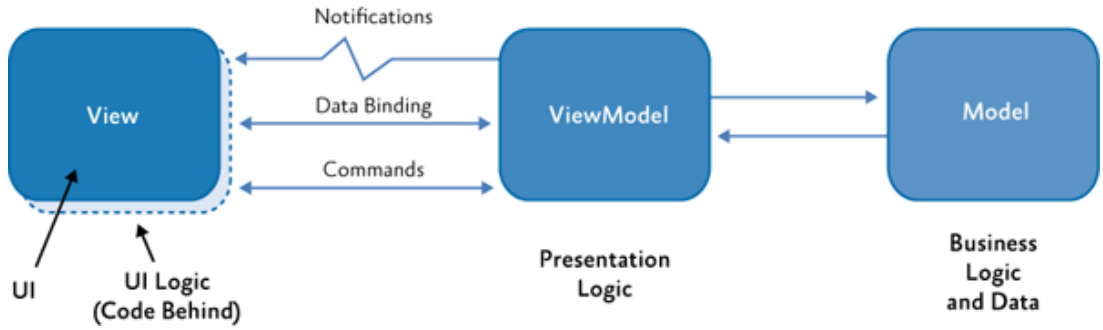


Abbildung 3.1: Datenfluss MVVM

Das MVVM-Modell bietet sich auch an, wenn die Daten des Programms oft in eine andere Form gebracht werden müssen, bevor sie dargestellt werden können. Zum Beispiel wird vom Model die aktuelle Zeit als Ganzzahl abgespeichert, welche nur die vergangenen Millisekunden seit einem bestimmten Zeitpunkt zählt; um die Zeit allerdings darstellen zu können, muss diese Ganzzahl zuerst in ein lesbares Datumsformat gebracht werden. Diese Aufgabe würde das ViewModel übernehmen.

3.2.2 Nachteile

Sowohl die ursprüngliche LibVLC-Bibliothek, als auch die entsprechenden C#-Bindings sind auf der offiziellen Dokumentationsseite nur spärlich dokumentiert. [vgl. 9] Beispielsweise verwenden alle Beispielprojekte der LibVLC Sharp Bindings das MVVM-Modell, daher ist nicht bekannt, ob eine Implementierung ohne dieses Modell überhaupt möglich ist. Weiters sind die einzelnen Funktionen zwar auf der Dokumentationsseite aufgelistet, allerdings nicht sehr ausführlich beschrieben. [vgl. 9]

3.2.3 Alternativen

Während der Recherche wurden wenige Alternativen gefunden. Diese Alternativen sind jedoch seit langem nicht mehr in Entwicklung und sind daher nicht für die Verwendung empfohlen. Die LibVLC-Bibliothek ist momentan die einzige kostenfreie Variante, RTSP-Streams mit Xamarin auf dem Mobilgerät abspielen zu können. Es gibt Bibliotheken von VASTreaming, jedoch sind diese kostenpflichtig zu erwerben, was für diese Diplomarbeit keine Option ist. [vgl. 10, Pricing]

3.3 GStreamer

3.3.1 Modularität

GStreamer ist ein extrem leistungsfähiges und vielseitiges Framework zur Erstellung von Medien-Streaming-Anwendungen. Viele der Vorzüge des GStreamer-Frameworks liegen in seiner Modularität: GStreamer kann neue Plugin-Module nahtlos integrieren. Aber da Modularität und Leistungsfähigkeit oft mit einem Preis für eine größere Komplexität einhergehen, ist das Schreiben neuer Anwendungen nicht immer einfach. [11, aus dem Englischen übersetzt]

GStreamer ist Pipeline-basiert, d.h. die Funktion wird mit einer Vielzahl von hintereinander geschalteten Filtern bestimmt. Eine Pipeline hat immer einen Eingang (Source) und einen oder mehrere Ausgänge. Zwischen diesen können sich beliebig viele Filter befinden, die unterschiedlichste Funktionen haben können:

- Signale de- und encodieren (Codec, z.B. x264enc)
- Größe und Position von Videos verändern (Caps, z.B. video/x-raw)
- die Pipeline in zwei Teile auftrennen, z.B. in Video und Audio (Demux)

- Daten zu einem Netzwerk-Paket zusammenbündeln (z.B. rtph264pay)

- etc.

Ein Filter hat bestimmte Ein- und Ausgangsformate, die er unterstützt. Als Beispiel nehmen wir den H264-Encoder: dieser unterstützt für das Eingangssignal RAW-Video, also decodierte Binärdaten. Am Ausgang kommt logischerweise ein H264-codierter Video-Stream heraus.

3.3.2 Version

Das GStreamer-Rahmenwerk wird aktuell vom GStreamer-Team fortlaufend upgedatet und verbessert. Die erste Version des GStreamer-Rahmenwerks wurde im Jahr 2001 mit der Versionsnummer 0.1.0 veröffentlicht. Im Jahr 2012 kam eine große Umstellung, bei der grundlegende Teile des Rahmenwerks ausgetauscht wurden, wodurch einige Teile nicht mehr kompatibel waren. Ab diesem Zeitpunkt begann die Versionsnummer mit 1.x.x, um die neuen Versionen deutlich von den Vorgängern abzutrennen. Die aktuelle Version des GStreamer-Rahmenwerks zum Verfassungszeitpunkt dieser Arbeit ist die Version 1.16.2.

Aktuell wird die Version 1.16.2 verwendet. Da aber alle 1.x.x-Versionen mehr oder weniger miteinander kompatibel sind, kann sich das ändern, sobald ein neues Update veröffentlicht wird. Es gibt keinen besonderen Grund auf genau dieser Version zu bleiben.

3.3.3 Alternativen

Es wurden während der Recherche einige mögliche Alternativen für das GStreamer-Rahenwerk gefunden:

- FFmpeg, ein Multimedia-Rahmenwerk, das vor allem im Bereich Transcodierung Verwendung findet. Es bietet alle Funktionen des GStreamer-Rahmenwerks an.

- LibVLC, die Grundlage des VLC Media Players, welche vor allem zur Wiedergabe verwendet wird. Diese Bibliothek kann zur Transcodierung und Live-Übertragung genutzt werden, ist allerdings nicht dafür optimiert.

3.3.4 Vorteile

GStreamer wird in diesem Projekt verwendet, da bei Live-Video-Übertragung möglichst wenig Latenz vorkommen soll. Das GStreamer-Rahmenwerk ist in dieser Hinsicht den anderen Multimedia-Rahmenwerken weit voraus. Zusätzlich gibt es von GStreamer keine einschränkenden Vorgaben, was Ein- und Ausgangsformat betrifft. Es kann jede beliebige Quelle mit jedem beliebigen Output verknüpft werden, solange die richtige Pipeline verwendet wird. Weiters ist das Projekt quelloffen, d.h. man ist in Bezug auf die Installationsdateien nicht auf den Hersteller angewiesen, sondern kann das Programm selbst für die jeweilige Plattform kompilieren, wenn der Hersteller das noch nicht getan hat. Außerdem kann jeder Entwickler Vorschläge für Änderungen am Code sowie Bugfixes vorbringen.

3.3.5 Funktionsweise

Die detaillierte Funktionsweise des GStreamer-Rahmenwerks lässt sich am besten anhand eines realistischen Beispiels erklären. In diesem Beispiel wird ein in Echtzeit generiertes Testvideosignal zuerst auf 1280x720 Pixel vergrößert und danach als H264-encodierter Stream via RTP verschickt. Mit dem Kommandozeilen-Programm `gst-launch-1.0.exe` kann ohne großen Aufwand eine Pipeline aufgebaut werden:

```
1  gst-launch-1.0 videotestsrc !                                                  1
   ↪ video/x-raw,width=1260,height=720,framerate=20/1 !
   ↪ autovideoconvert ! queue ! x264enc tune=zerolatency bitrate=4096
   ↪ speed-preset=superfast ! queue ! rtph264pay config-interval=1
   ↪ mtu=1300 ! udpsink host=127.0.0.1 port=5000
```

Der Filter „queue“ wird hier verwendet, um vor und nach dem Encoding-Vorgang ein kurzes Video-Segment in einem Zwischenspeicher zu behalten. Dies ist notwendig, da der H264-Codec mehrere Video-Frames auf einmal komprimiert.

3.3.6 Zusatzmodul

gst-rtsp-server ist ein Zusatzmodul für das GStreamer- Rahmenwerk, welches die Verwendung des RTSP-Protokolls für Server-Applikationen bereitstellt. Mit diesem Modul können Medien-Streams, welche mit GStreamer verarbeitet wurden, mittels dem RTSPProtokoll den Netzwerk bereitzustellen.

3.4 Live555 Proxy

Ein RTSP-Stream ist grundsätzlich eine Unicast-Verbindung zwischen Server und Client. Falls mehrere Clients den gleichen Medien-Stream abrufen möchten, muss der Server diesen für jeden Client einzeln transcodieren und verschicken. Dies stellt eine große Belastung für den Server und dessen Internetanbindung dar. Mit Hilfe eines Proxy-Servers muss der Medien-Stream nur einmal verarbeitet werden, bevor er dann an die einzelnen Clients verteilt wird.

Der Live555 Proxy Server ist eine quelloffene Applikation, welche für genau diese Aufgabe von Live Networks Inc. entwickelt wurde. Mit Hilfe des Live555 Proxy Servers wird ein RTSP-Stream einmalig eingelesen und an alle verbundenen Clients verteilt. Dadurch bleibt die CPU-Belastung des Servers unabhängig von der Anzahl der verbundenen Clients, wodurch er unter Umständen mehrere verschiedene Streams bereitstellen könnte. Für jeden Medien-Stream, den der Server anbietet, wird eine weitere Live555-Proxy-Server-Instanz benötigt.

Das ursprüngliche Veröffentlichungsdatum des Live555 Proxy Servers konnte nicht gefunden werden. Bei Analyse der Google-Such-Trends zum Begriff Live555 wurde festgestellt, dass Suchanfragen mit Oktober 2005 gestartet haben. Daher lässt sich vermuten, dass das Programm kurz vor diesem Zeitpunkt veröffentlicht wurde. [vgl. 12, Interest over time]

Neben dem Live555 Proxy Server entwickelt das Unternehmen mehrere Projekte:

- Live555 Media Server, mit dem lokale Medien-Dateien per RTSP dem Netzwerk bereitgestellt werden können.
- liveCaster, zur Multicast-Übertragung von MP3-Dateien
- diverse Kommandozeilenprogramme zum Senden und Empfangen von Echtzeit-Streams

Die von Live Networks Inc. entwickelten Programme und Bibliotheken finden in vielen bekannten Applikationen Anwendung, darunter auch der VLC Media Player mit der dazugehörigen LibVLC. In diesem Programm werden die Live555-Bibliotheken zum Empfangen und Entschlüsseln von RTSP-Streams genutzt.

4 Push-Benachrichtigung

4.1 Push Notification Service

Eine Notification dient dazu den Benutzer über etwas zu informieren. Dies kann ein Kalender-Eintrag, eine eingetroffene SMS oder ein anderes Ereignis sein. Diese Art der Benachrichtigung wird meist über eine Local Notification erzeugt. Eine Local Notification wird aufgrund eines lokal am Gerät auftretenden Ereignisses ausgelöst.

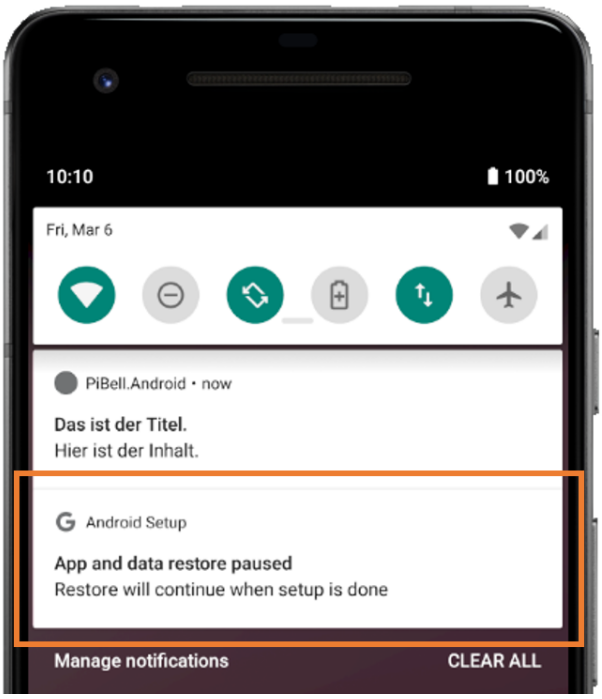


Abbildung 4.1: Lokale Benachrichtigung

Im Gegensatz dazu dient eine Push Notification dazu, den Benutzer über ein externes Ereignis auf einem anderen Gerät aufmerksam zu machen. Diese Art der Benachrichtigung findet meist Anwendung in der Katastrophenwarnung, um den Benutzer vor einer herannahenden Gefahr zu warnen. Um den Benutzer also darüber zu informieren, dass jemand an der Haustür klingelt, wird eine Push Benachrichtigung verwendet, da das Anläut-Ereignis bei der Außenstation erfolgt.

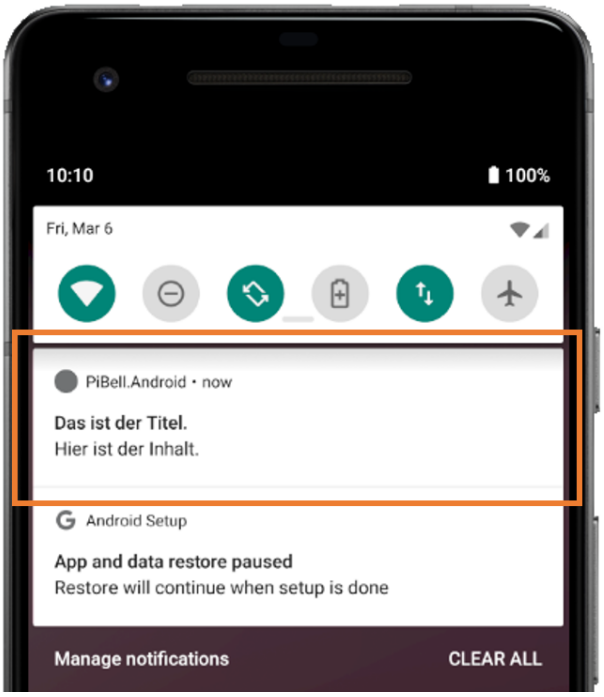


Abbildung 4.2: Push-Benachrichtigung

4.1.1 Funktionsweise

Eine Push Notification wird immer von einem sogenannten Push Notification Service an alle Zielgeräte geschickt. Dieser Service wird vom jeweiligen Betriebssystem-Hersteller bereitgestellt. Im Fall der hier entwickelten App sind das Google und Apple, mit ihren Push Notification Services GCM und APNs.

Der Service ist dafür verantwortlich, die Benachrichtigung an alle registrierten Zielgeräte zu schicken. Wenn das Zustellen nicht möglich ist, bricht der Service die Zustellung nach mehreren Versuchen für dieses Gerät ab. Alle anderen Geräte erhalten die Benachrichtigung. Daher ist es nicht sichergestellt, dass eine Push-Benachrichtigung bei jedem Gerät ankommt.

Zusätzlich zum Nachrichtentext können bei einer Push Notification zusätzliche Daten mitgeschickt werden. Diese können zum Beispiel die App über die genaue Benachrichtigungsursache informieren.

4.2 AppCenter Push

AppCenter Push hat in erster Linie die Funktion, den Zugriff auf die verschiedenen Push Notification Services zu vereinheitlichen und damit einfacher zu gestalten. Microsoft

AppCenter verwaltet alle Server-API-Schlüssel, Zielgerätelisten und ähnliche Einstellungen, um den Prozess des Benachrichtigung-Sendens zu vereinfachen. AppCenter Push ist entweder über das Web-Interface unter `https://appcenter.ms/` oder über die AppCenter API erreichbar und verwendbar.

Die AppCenter Push SDK abstrahiert den ganzen Prozess der Geräte-Registration, des Registrieren des Eingangs-Events und versteckt alles hinter einer einzelnen simplen Funktion `AppCenter.Start(„App Secret", typeof(Push));`. Das App Secret ist der von AppCenter zugewiesene Schlüssel mit dem sichergestellt wird, dass das Gerät für das richtige AppCenter Projekt registriert wird. Ohne diese SDK müsste in der App viel mehr konfiguriert werden, was einen höheren Programmieraufwand und größere Fehlerrate darstellt. Außerdem würde das Rad neu erfunden werden.

In der entwickelten App wird die AppCenter-SDK-Version 2.6.4 verwendet. Zu beachten ist, dass AppCenter Push nicht mehr weiterentwickelt und irgendwann dieses Jahr der Service abgeschaltet wird, wie es John Wargo in seinem Blog-Post am 3. Februar 2020 schrieb. Als Alternative wird von Microsoft der Azure-Notification-Hubs-Dienst angeboten. Bevor AppCenter Push endgültig terminiert wird, will Microsoft detaillierte Anleitungen zum Umstieg auf Azure bereitstellen.

4.2.1 Konfiguration des SDK

Um AppCenter Push in der Applikation verwenden zu können, müssen mehrere NuGet-Pakete installiert werden. Das AppCenter Paket beinhaltet alle Kernfunktionen wie AppCenter.Start(), das AppCenter-Push-Paket erweitert das Grundpaket um Push-bezogene Funktionen und Datentypen.

NuGet-Paket	PiBell	PiBell.Android	PiBell.iOS
AppCenter	Ja	Ja	Ja
AppCenter Push	Ja	Ja	Ja

Tabelle 4.1: Installation NuGet-Pakete

5 Programm-Dokumentation

5.1 Übersicht

Das im Zuge dieser Diplomarbeit entwickelte Software-Projekt basiert auf einer der von Microsoft bereitgestellten Xamarin.Forms-Vorlagen, welche als Teil von Visual Studio bereitgestellt werden. Es gibt mehrere Auswahlmöglichkeiten, wobei alle Vorlagen bis auf „Blank“ Beispielcode beinhalten. Da der Beispielcode der anderen Vorlagen für dieses Projekt nicht relevant ist und ohnehin gelöscht werden müsste, wird die „Blank“-Vorlage verwendet. Diese Vorlage beinhält nur eine Haupt-Oberflächenseite ohne jeglichen Beispielcode.

Der immer wieder vorkommende Name PiBell ist eine freigeistliche Erfindung. PiBell ist der Projektname, der sich aus Raspberry Pi und „Bell“, dem englischen Wort für Klingel oder Glocke, zusammensetzt.

Visual Studio organisiert alle Programmteile in einer sogenannten Solution, welche diesen Projektnamen trägt. In diesem Fall beinhält die Solution „PiBell“ drei Programmteile oder „Projects“:

PiBell ist das portable Projekt, auch .NET-Standard-Projekt genannt. Dieses beinhaltet den ganzen plattformunabhängigen Code, sowie die Definition der UI-Oberfläche mittels XAML-Dateien. In diesem Projekt befindet sich der größte entwickelte Code-Anteil, da die meisten Funktionen, wie das Abspielen von Videos mit Hilfe der LibVLC-Bindings, plattformunabhängig funktionieren und deswegen geteilt werden können.

PiBell.Android beinhaltet allen Code, der plattformspezifisch für Android geschrieben wurde. Unter anderem ist hier die Android-Implementation des Mikrofon-Aufnahme-Services, sowie das Berechtigungs-Management enthalten. Die Mikrofon-Aufnahme erfolgt auf einer sehr hardwarenahen Ebene, was die Plattform-Abhängigkeit erklärt.

PiBell.iOS beinhaltet wie PiBell.Android den plattformspezifischen Code für die iOS-Plattform. Aufgrund mangelnder Entwicklungswerkzeuge wird dieser Teil nicht großartig behandelt.

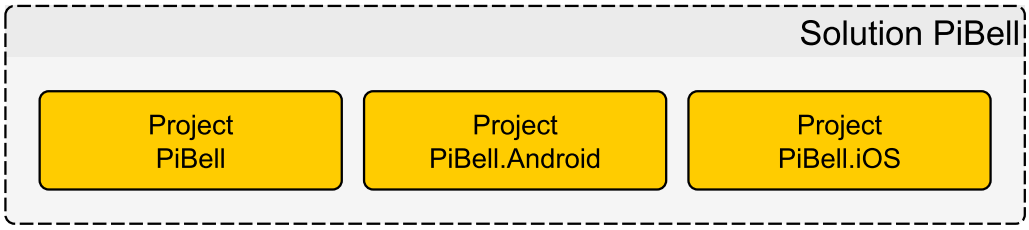


Abbildung 5.1: Aufbau der Solution

5.2 Portable Project

Im folgenden Teil werden die wesentlichen Teile des portablen Projektes PiBell beschrieben. In den einzelnen Programmteilen gibt es immer wiederkehrende Abläufe, wie Initialisierung, die nur einmal beschrieben werden. Die von Microsoft vorgefertigten Teile werden nicht behandelt. Der Fokus liegt damit auf den selbst erstellten Programmteilen. Die Programmteile werden generell in derselben Reihenfolge behandelt, wie sie vom Programm aufgerufen werden.

Textsegmente mit vorgestellter Zeilennummer sind direkt vom Programm eingefügt. Der beschreibende Text bezieht sich auf den Programmteil mit Hilfe der Zeilennummern.

5.2.1 App.xaml.cs

Im diesem Programmteil wird die Applikation grundlegend initialisiert. Die Initialisierung beinhaltet unter anderem das Laden aller benötigten Programm-Bibliotheken und NuGet-Pakete.

```
17 public App()                                                          17
18 {                                                                     18
19     InitializeComponent();                                            19
20     Core.Initialize();                                                20
21     MainPage = new MainPage();                                        21
22                                                                       22
23                                                                       23
24     AppCenter.LogLevel = Microsoft.AppCenter.LogLevel.Verbose;        24
25     AppCenter.Start("android={Your Android App secret here};" +        25
26                     "uwp={Your UWP App secret here};" +               26
```

```
27                      "ios={Your iOS App secret here}",                    27
28          typeof(Analytics), typeof(Crashes), typeof(Push));               28
29  }                                                                        29
```

19-20: Die Funktionen InitializeComponent() und Core.Initialize sind vorgefertigte Funktionen; diese sind zu verwenden, um die Applikation und alle benötigten Bibliotheken zu initialisieren.

21: Die Anweisung MainPage = new MainPage() erzeugt eine neue Oberfläche, mit der der Benutzer mit der App interagiert. Diese Oberfläche erscheint nach dem App-Start am Bildschirm und passt sich automatisch der physikalischen Bildschirmgröße an.

24-28: Mit AppCenter.LogLevel kann festgelegt werden, welche Log-Nachrichten angezeigt werden. In diesem Fall werden Alle Nachrichten aktiviert. Anschließend wird das AppCenter SDK mit AppCenter.Start(..) gestartet. Am Ende dieser Anweisung werden alle Module (Analytics, Crashes und Push), die dazu gestartet werden sollen, angegeben.

5.2.2 MainPage.xaml

In einer XAML-Datei wird das Aussehen der Oberfläche nach dem WYSIWYM-Prinzip beschrieben. Das Format ähnelt sehr einer XML-Datei, in der Hinsicht, dass ein Element mit einer spitzen Klammer (<) beginnt und mit einem Schrägstrich und einer spitzen Klammer (/>) endet. Einige Elemente, wie zum Beispiel Listen oder ein Grid, können sogenannte Kind-Elemente beinhalten.

```
3   <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"                3
4                xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"       4
5                xmlns:local="clr-namespace:PiBell"                           5
6                xmlns:piBell="clr-namespace:PiBell;assembly=PiBell"          6
7                xmlns:shared="clr-namespace:LibVLCSharp.Forms.Shared;        7
                 ↪  assembly=LibVLCSharp.Forms"
8                x:Class="PiBell.MainPage"                                    8
9                Title="Main Page"                                            9
10               BackgroundColor="#ff1b1b1b">                                 10
```

3-8: Am Anfang der XAML-Datei wird der allgemeine Seiten-Typ, gemeinsam mit einigen Quell-Abkürzungen (Namenspräfix für Xamarin-fremde Elemente) definiert. In diesem Fall handelt es sich um eine simple ContentPage mit den Standard-Abkürzungen. Zur Verwendung der LibVLC wurde in Zeile 7 eine zusätzliche Quelle definiert. Die neu hinzugefügte Quelle erlaubt den Zugriff auf Elemente wie die VideoView, welche ein Video als Teil der Oberfläche darstellt.

9: Mit x:Class wird die Programmklasse angegeben, die den Code der Oberfläche beinhält. In dieser Klasse finden sich alle Event-Handler für jegliche Oberflächenereignisse, wie das Drücken eines Buttons.

10: Das Festlegen der Title-Variable ist in diesem Fall optional. Falls eine Master-Detail-Page-Struktur verwendet werden würde, wäre der Titel oben in der Kopfzeile der App zu sehen. Da es sich hier aber um ein Sigle-Page-Layout handelt, ist der Titel nicht zwingend notwendig.

11: Mit der Variable BackgroundColor wird die Hintergrundfarbe der Oberfläche definiert. Die gewünschte Farbe wird als HEX-Zahl mit dem Format Alpha-Rot-Grün-Blau angegeben. Der Alpha-Wert legt die Durchsichtigkeit der Farbe fest. Es wird ein dunkles Design verwendet, da dieses in einer wenig beleuchteten Umgebung die Augen besser schont.

```
12 <ContentPage.Content>                                                    12
13     <Grid Margin="10" x:Name="MainGrid">                                 13
14         <Grid.RowDefinitions>                                            14
15             <RowDefinition Height=".7*" />                               15
16             <RowDefinition Height="3*" />                                16
17             <RowDefinition Height="2*" />                                17
18         </Grid.RowDefinitions>                                           18
19         <Grid.ColumnDefinitions>                                         19
20             <ColumnDefinition Width="*" />                               20
21         </Grid.ColumnDefinitions>                                        21
```

12: Mit dieser Zeile wird das Element ContentPage.Content geöffnet. Content steht hier für den gesamten Oberflächeninhalt, also alle Schaltflächen und Anzeigen.

13-21: Als erstes und einziges Kind-Element von ContentPage.Content wird die Komponente Grid gewählt. Diese ermöglicht es, mehrere Kind-Elemente in einer schach-

brettartigen Anordnung zu organisieren. Hierfür werden mit Hilfe der Row- und ColumnDefinitions die Höhen und Breiten der einzelnen Zeilen und Spalten festgelegt. Der Stern (*) in der Längenangabe bedeutet relatives Maß bezogen auf die verfügbare Breite bzw. Höhe.

```
23 <Entry x:Name="EditMrl" Text="{Binding Mrl, Mode=TwoWay}" Grid.Row="0"
   ↪ Grid.Column="0" TextColor="White" />
24
25 <shared:VideoView x:Name="VideoView" Grid.Column="0" Grid.Row="1"
   ↪ MediaPlayer="{Binding MediaPlayer}"/>
```

23: Hier wird das erste eigenständige Oberflächen-Element erzeugt. Es handelt sich um ein Text-Eingabe-Feld (Entry), mit dem zu Testzwecken die Serveradresse eingegeben werden kann. Dies ist notwendig, da die finale Serveradresse noch nicht bekannt ist und sich diese im Zuge der Entwicklung ständig ändert. Für den späteren Betrieb könnte dieses Problem umgangen werde, indem die aktuelle Server-Adresse per Push-Benachrichtigung mitgeschickt wird. Hier ist auch eine Anwendung des MVVM-Modells erkennbar: es wird mit Hilfe des Binding-Schlüsselwortes der Speicherort des angezeigten Textes festgelegt. Mode=TwoWay beschreibt die Richtung der Synchronisierung der Daten. Egal, ob sich der Wert im Speicher oder der Wert in der Anzeige ändert, wird die jeweils andere Seite aktualisiert. Mit Grid.Row und Grid.Column wird die Position im vorhin definierten Grid gesetzt.

25: Das zweite Oberflächen-Element, eine VideoView der LibVLC, wird hier erzeugt. Da die Komponente VideoView nicht standardmäßig in Xamarin enthalten ist, wird das am Anfang der Datei definierte Namenspräfix „shared:“ benötigt.

```
36 <ImageButton x:Name="BtConnect" Source="call.png"
   ↪ BackgroundColor="LimeGreen" Grid.Row="0"
37              Grid.Column="0" Margin="10"
38              Clicked="BtConnect_Clicked"/>
```

36: Ein ImageButton kann im Vergleich zu einem normalen Button als Inhalt ein Bild anzeigen. Die Bildquelle wird mit der „Source“-Property gesetzt. Es gibt mehrere mögliche Bildformate, wobei hier PNG aufgrund von Transparenz-Unterstützung gewählt wurde. Ohne Definition der „BackgroundColor“ würde der ImageButton mit der Default-Farbe grau gefüllt werden.

38: Mit dem EventHandler „Clicked“ kann die Funktion, die aufgerufen wird, wenn der ImageButton gedrückt wird, spezifiziert werden.

```
43 <ImageButton x:Name="BtToggleSpeaker" Source="SpeakerMute.png"
   ↪ BackgroundColor="Transparent"
44              Grid.Row="1" Grid.Column="0" Margin="10"
45              Clicked="BtToggleSpeaker_Clicked">
46     <VisualStateManager.VisualStateGroups>
47         <VisualStateGroup x:Name="SpeakerStates">
48             <VisualState Name="Mute">
49                 <VisualState.Setters>
50                     <Setter Property="Source"
51                             Value="SpeakerMute.png" />
52                 </VisualState.Setters>
53             </VisualState>
54             <VisualState Name="Unmute">
55                 <VisualState.Setters>
56                     <Setter Property="Source"
57                             Value="SpeakerUnmute.png" />
58                 </VisualState.Setters>
59             </VisualState>
60         </VisualStateGroup>
61     </VisualStateManager.VisualStateGroups>
62 </ImageButton>
```

47-62: Mit dem VisualStateManager können für die meisten Oberflächen-Elemente bestimmte Zustände definiert werden (z.B. wenn sie gedrückt sind). Hier werden zwei Zustände definiert, die das angezeigte Bild dem Lautsprecher-Zustand anpassen.

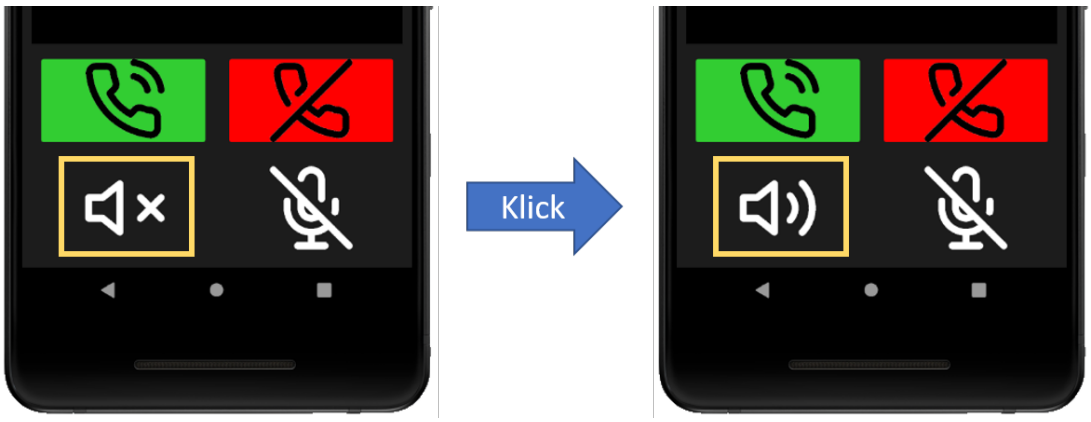


Abbildung 5.2: VisualStateManager

Die hier beschriebenen Konzepte werden mehrmals verwendet, um die Oberfläche aufzubauen. Die Datei als Gesamtheit ergibt die in Bild *5.3 App-Oberfläche* gezeigte Oberfläche.

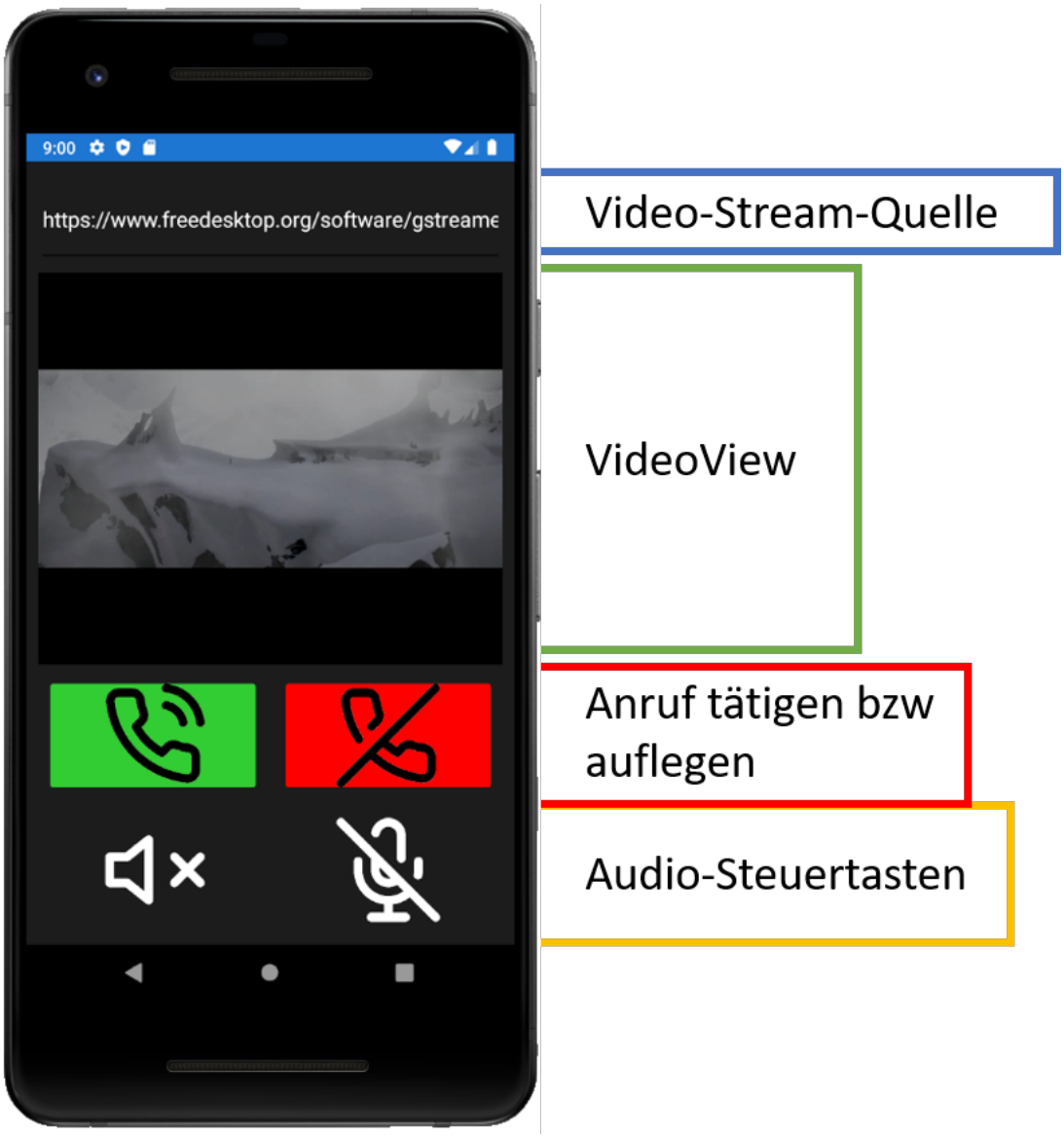


Abbildung 5.3: App-Oberfläche

5.2.3 MainPage.xaml.cs

Dieser Teil beinhaltet den C#-Code, der für die Steuerung und Verwaltung der Oberfläche notwendig ist. Unter anderem wird hier das Verhalten beim Wechseln zwischen Hoch- und Querformat festgelegt.

```
36      if (!AppCenter.Configured)                                    36
37          Push.PushNotificationReceived += async (sender, e) =>     37
38          {                                                         38
39  #if DEBUG                                                         39
```

```
40              Console.WriteLine("DEBUG - Push-Notification recieved");
41  #endif
42              foreach (string key in e.CustomData.Keys)
43              {
44  #if DEBUG
45                  Console.WriteLine($"DEBUG - Custom Data:
                    ↪ {key}:{e.CustomData[key]}");
46  #endif
47                  if (key == "mrl")
48                      _player.Mrl = e.CustomData[key].ToString();
49              }
50
51              if (await DisplayAlert("Incoming Call", "Connect now?", "Yes",
                ↪ "No"))
52              {
53                  _player.StartCall();
54                  _streamer.StartCall();
55              }
56          };
```

Hier wird das Verhalten bei eingehender Push Benachrichtigung festgelegt.

39,41,44,46: Die mit „#“ beginnenden Anweisungen sind sogenannte Preprocessor-Statements, mit denen dem Compiler spezielle Anweisungen gegeben werden können. Hier soll der Programmteil, der sich zwischen `#if DEBUG` und `#endif` befindet, nur im DEBUG-Modus verarbeitet werden. Wenn ein anderer Compiler-Modus ausgewählt ist, werden diese Teile ignoriert.

42-49: Mit einer Schleife werden alle Zusatzdaten der Benachrichtigung überprüft. Wenn der Datensatz `mrl` enthalten ist, wird dessen Wert in `_player.Mrl` gespeichert. Der Wert stellt die Server-Adresse dar, von der der Video-Stream abgerufen werden soll.

51-55: Der Benutzer wird per Popup gefragt, ob der Anruf entgegengenommen werden soll (Abb. *5.4 Abfrage Anrufannahme*).

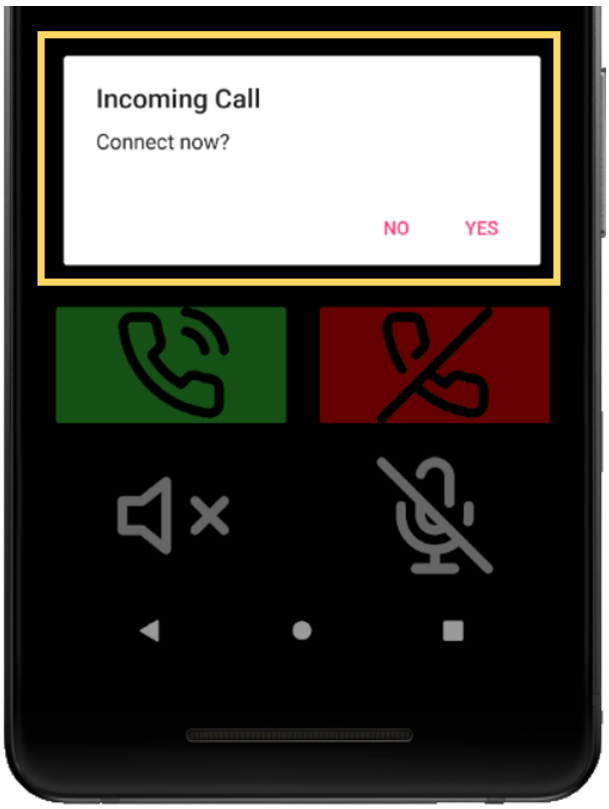


Abbildung 5.4: Abfrage Anrufannahme

Wenn die Applikation in den Hintergrund geht, verliert die LibVLC die Referenz auf die VideoView und kann den Video-Stream nicht mehr darstellen. Daher muss die Wiedergabe gestoppt werden, sobald die App in den Hintergrund geht und neu gestartet werden, wenn die App wieder im Vordergrund ist. Da die Benachrichtigung über den Vorder- bzw. Hintergrundstatus der Applikation nur über das native Projekt funktioniert, muss diese Information vom nativen Projekt zum portablen Projekt übertragen werden. Dies geschieht mit der MessagingCenter-Komponente.

```
57 MessagingCenter.Subscribe<string>(this, "OnPause", app =>
58 {
59     _wasConnected = _player.MediaPlayer.IsPlaying;
60     _player.MediaPlayer.Pause();
61     _prevPosition = _player.MediaPlayer.Position;
62     _player.MediaPlayer.Stop();
63
64     MainGrid.Children.Remove(VideoView);
65 });
```

```
66
67  MessagingCenter.Subscribe<string>(this, "OnRestart", app =>
68  {
69      RegenerateVideoView();
70      VideoView.MediaPlayer = _player.MediaPlayer;
71      if (_wasConnected)
72      {
73          _player.MediaPlayer.Play();
74          _player.MediaPlayer.Position = _prevPosition;
75      }
76
77      _prevPosition = 0;
78  });
```

57-65: Dieses Ereignis wird aufgerufen, wenn die Applikation in den Hintergrund wechselt. Bevor dies geschieht, wird die aktuelle Wiedergabe-Position in einer Variablen gespeichert und die Wiedergabe gestoppt. Anschließend wird die VideoView von der Oberfläche entfernt, damit sie später neu hinzugefügt werden kann.

67-78: Das `OnRestart`-Event wird aufgerufen, wenn die App vom Hintergrund wieder in den Vordergrund geht. Hier geschieht derselbe Vorgang wie davor, jedoch in sinngemäß umgekehrter Reihenfolge. Als erstes wird die VideoView der Oberfläche wieder hinzugefügt, damit die LibVLC eine Referenz dazu bilden kann. Falls davor ein Video gespielt wurde, wird dieses fortgesetzt und die Variable für die vorherige Wiedergabeposition wird gelöscht.

```
84  BindingContext = _player;
```

84: Der BindingContext gibt für das MVVM-Modell an, wo die Variablen, die mit der Oberfläche verknüpft sind, sich befinden. In diesem Fall sind alle Variablen in der Player-Instanz `_player` zu finden.

An folgender Funktion wird das Verhältnis von Oberflächenereignissen mit dem entsprechenden Code-Behind erklärt.

```
109 private void BtConnect_Clicked(object sender, EventArgs e)
110 {
111     _player.StartCall();
```

```
112     _streamer.StartCall("192.168.43.78");
113 }
```

109: Eine Funktion für ein Button-Event hat grundsätzlich als Rückgabe-Typ `void`, also nichts. Auf die Funktion kann nur aus der selben Klasse zugegriffen werden, was mit dem `private`-Schlüsselwort bekanntgegeben wird. Als Parameter werden Informationen zum Event-Auslöser und zum Event selbst mitgegeben.

111-112: Wenn der grüne „Anrufen“-Button gedrückt wird, beginnen `_player` und `_streamer` damit, die Daten abzurufen bzw. bereitzustellen. Diese Funktionen werden im Abschnitt *5.2.4 MediaClasses.cs* genauer beschrieben.

Im letzten Abschnitt dieser Datei wird das Verhalten beim Wechseln zwischen Hoch- und Querformat festgelegt.

```
121 protected override void OnSizeAllocated(double width, double height)
122 {
123     base.OnSizeAllocated(width, height);
124     if (_width != width || _height != height)
125     {
126         _width = width;
127         _height = height;
128         if (width < height) //Portrait
129         {
130             MainGrid.ColumnDefinitions.Clear();
131             MainGrid.ColumnDefinitions.Add(new ColumnDefinition
132                 {Width = new GridLength(1, GridUnitType.Star)});
133
134             MainGrid.RowDefinitions.Clear();
135             MainGrid.RowDefinitions.Add(new RowDefinition
136                 {Height = new GridLength(0.7, GridUnitType.Star)});
137             MainGrid.RowDefinitions.Add(new RowDefinition
138                 {Height = new GridLength(3, GridUnitType.Star)});
139             MainGrid.RowDefinitions.Add(new RowDefinition
140                 {Height = new GridLength(2, GridUnitType.Star)});
141
142             Grid.SetColumnSpan(EditMrl, 1);
143             Grid.SetColumn(ButtonGrid, 0);
```

```
144                 Grid.SetRow(ButtonGrid, 2);                                    144
145             }                                                                  145
```

[...]

```
164         }                                                                      164
165     }                                                                          165
```

121: Die `OnSizeAllocated()`-Funktion ist eine vorgefertigte Funktion, die aufgerufen wird, wenn sich die Abmessungen der Oberfläche geändert haben. Die neuen Abmessungen sind über die Parameter `width` und `height` verfügbar.

124-164: Wenn die Abmessungen anders sind, als die Abmessungen beim letzten Aufruf wird das Oberflächen-Layout entsprechend angepasst. Falls die Höhe größer als die Breite ist, greift der erste Teil der If-Else-Struktur, ansonsten der zweite.

130-144: Zuerst werden die Reihen- und Spalten-Definitionen gelöscht, um sie neu zuweisen zu können. Für das Hochformat werden eine Spalte und drei Reihen benötigt. Diese werden mit den richtigen relativen Maßen hinzugefügt. Wenn dies geschehen ist, werden die Oberflächenelemente den richtigen Plätzen im Grid zugewiesen.

146-163: Das Verändern der Oberfläche für Querformat wird hier ausgelassen, da es sinngemäß äquivalent zum Hochformat ist.

5.2.4 MediaClasses.cs

In dieser Source-Datei sind sämtliche Hilfsklassen zum Abrufen und Bereitstellen von Medieninformationen definiert. Grundsätzlich finden sich hier zwei Klassen:

- Player, die für das Abrufen des Medien-Streams vom Server und Darstellen von diesem zuständig ist
- Streamer, mit der das Zurückschicken der Mikrofon-Rohdaten erfolgen soll

5.2.4.1 Player

Die Player-Klasse implementiert das MVVM-Modell, um die MediaPlayer-Komponente mit der Oberfläche zu verknüpfen. Um die Anzeige über eine Änderung der Variablen zu

informieren, muss der `PropertyChanged`-EventHandler mit dem Variablennamen als Argument aufgerufen werden. Um die Lesbarkeit den Codes zu vereinfachen, wird dieser Aufruf meist in einem Unterprogramm versteckt, welches von Visual Studio automatisch generiert werden kann. Daher wird auf den genauen Aufbau dieses Unterprogrammes nicht genauer eingegangen. Die Initialisierung beinhaltet keine besonderen Funktionen und wird daher ebenfalls nicht behandelt.

```
36 public bool ToggleSpeaker()                                          36
37 {                                                                    37
38     MediaPlayer.Volume = _speakerActive ? 0 : 100;                   38
39     return _speakerActive = !_speakerActive;                         39
40 }                                                                    40
```

38-39: Über die Volume-Eigenschaft des MediaPlayer-Elements kann die Audio-Ausgabe in der Lautstärke verändert oder ganz abgeschaltet werden. Mit der Funktion `ToggleSpeaker()` wird der aktuelle Zustand der Ausgabe invertiert. Das heißt, wenn die Ausgabe aktiviert ist, wird sie durch einen Aufruf des Unterprogrammes deaktiviert und umgekehrt. Der neue Zustand wird als Wahrheitswert an die aufrufende Funktion zurückgegeben, wobei `true` bedeutet, dass die Audio-Ausgabe aktiv ist. Als Abkürzung für ein mehrzeiliges If-Else-Statement wird der Ternary-Operator verwendet. Diese Anweisung nimmt abhängig von einem Wahrheitswert zwei verschiedene Werte an. Die Syntax dieser Abkürzung sieht so aus:

```
1 variable = <true/false> ? <Wert wenn true> : <Wert wenn false>;      1
```

Die `StartCall()`-Funktion wird aufgerufen, wenn der Benutzer auf den grünen Knopf drückt.

```
42 public void StartCall()                                              42
43 {                                                                    43
44     MediaPlayer.Media = new Media(LibVlc, Mrl, FromType.FromLocation);  44
45     MediaPlayer.Play();                                              45
46     MediaPlayer.Volume = _speakerActive ? 100 : 0;                   46
47 }                                                                    47
```

44: Bevor die MediaPlayer-Komponente ein Video abrufen kann, muss erst definiert werden, wo dieses Video zum Abruf bereitsteht. Über die Variable `Media` kann die Medi-

eninformation des Ziel-Videos festgelegt werden. Die Variable `Mrl` stellt hier die aktuelle Adresse des Video-Streams dar.

45-46: Mit der `Play()`-Funktion kann anschließend, wie der Name vermuten lässt, die Wiedergabe gestartet werden. Jedoch wird im gleichen Zug die Wiedergabe-Lautstärke zurückgesetzt, weshalb sie nach dem Aufruf der `Play()`-Funktion wieder dem aktuell gewünschten Zustand der Audio-Wiedergabe angepasst werden muss.

```
49 public void EndCall()
50 {
51     MediaPlayer.Stop();
52 }
```

49: Die Wiedergabe des Video-Streams kann am Ende des Gesprächs mit der `Stop()`-Funktion beendet werden.

Die C#-Sprache bietet aufgrund der strengen Objektorientierung das Sprachkonstrukt „Property“ an. Eine Property kann mit einer gewöhnlichen Variable verglichen werden, allerdings bietet sie zusätzlich `get`- und `set`-Funktionen an. Diese Zusatzfunktionen werden aufgerufen, wenn auf den Wert der Property zugegriffen bzw. dieser verändert wird. Dies wird anhand eines Beispiels erläutert:

```
54 private string _mrl;
55 public string Mrl
56 {
57     get => _mrl;
58     set
59     {
60         _mrl = value;
61         OnPropertyChanged();
62     }
63 }
```

54: Eine Property mit benutzerdefinierten `get`- und `set`-Funktionen benötigt eine zusätzliche Variable, die den eigentlichen Wert der Property speichert. Diese kann auch dann `private` und somit nach außen versteckt sein, wenn die Property selbst `public`, also von außen zugänglich ist.

57: Die `get`-Funktion ist bei den meisten Properties so definiert, dass der Wert der internen Variablen einfach weitergegeben wird, da beim Auslesen in den meisten Fällen keine zusätzliche Aktion erforderlich ist.

58-62: In der `set`-Funktion wird hier nicht nur der Wert der internen Variable mit dem neuen Wert überschrieben, sondern auch das System über eine Änderung informiert. Dies ist für die Funktion des MVVM-Modells essenziell.

5.2.4.2 Streamer

Die Streamer-Klasse ist deutlich kürzer, als die Player-Klasse, da die Streamer-Klasse selbst keine wirkliche Funktionalität beinhaltet. Vielmehr dient diese Klasse als Interface zu den plattformspezifischen Softwaremodulen, die für die Audioaufnahme zuständig sind. Dies wird mit Hilfe eines DependencyService erreicht.

```
90  private IAudioRecordingService Service { get; set; }
91
92  public Streamer()
93  {
94      Service = DependencyService.Get<IAudioRecordingService>();
95  }
```

94: Der DependencyService dient als „Fernsteuerung“, mit dem vom portable Projekt ausgehend Ereignisse und Funktionen im nativen Code aufgerufen werden können.

```
112 public interface IAudioRecordingService
113 {
114     void Start(IPEndPoint target);
115     void Stop();
116     bool ToggleMic();
117 }
```

112-117: Mit einem `interface` wird festgelegt, welche Funktionen eine Klasse, die von diesem abgeleitet wird, haben muss. In diesem Fall muss der plattformspezifische Teil der Audioaufnahme die Funktionen `Start()`, `Stop()` und `ToggleMic()` definieren, um mit dem Interface kompatibel zu sein. Über ein Interface können nicht nur Funktionen, sondern auch Variablen und Properties vorgegeben werden.

5.3 PiBell.Android

Dieses Projekt beinhaltet den gesamten nativen Code für die Android-Plattform. Dazu gehören das Berechtigungs-Management, sowie die Audio-Aufnahme.

5.3.1 MainActivity.cs

Ähnlich wie bei *5.2.1 App.xaml.cs* wird hier das native Android-Projekt initialisiert. Dazu gehört das Laden aller Runtime-Ressourcen und des benötigten Programm-Codes, sowie Überprüfen bzw. Anfragen der Berechtigungen.

```
33 if (CheckSelfPermission(Manifest.Permission.RecordAudio) !=
   ↪ Permission.Granted)
34 {
35     RequestPermissions(new[] { Manifest.Permission.RecordAudio }, 1);
36 }
```

33-36: Es wird überprüft, ob die Berechtigung für die Audio-Aufnahme bereits erteilt ist. Wenn das nicht der Fall ist, wird diese beim System angefragt. Der Benutzer erhält dann ein PopUp, mit dem er gefragt wird, ob die Berechtigung erteilt werden soll.

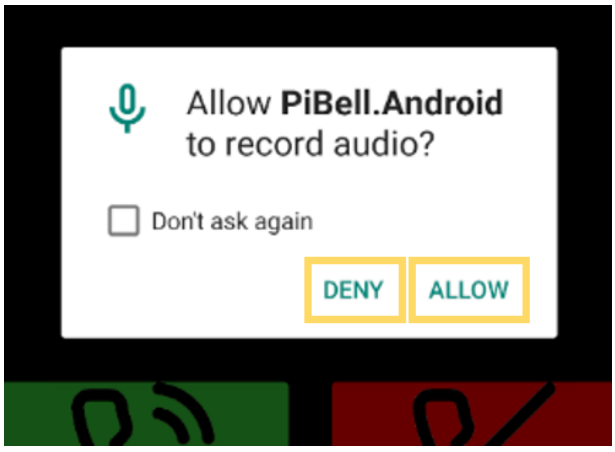


Abbildung 5.5: Anfrage für Berechtigung

```
39 protected override void OnPause()
40 {
41     base.OnPause();
42     MessagingCenter.Send("app", "OnPause");
43 }
```

```
44                                                                    44
45  protected override void OnRestart()                              45
46  {                                                                 46
47      base.OnRestart();                                             47
48      MessagingCenter.Send("app", "OnRestart");                     48
49  }                                                                 49
```

39-49: Das in *5.2.3 MainPage.xaml.cs* beschriebene Verhalten beim Wechsel der App von und in den Hintergrund benötigt ein vom nativen Projekt ausgelöstes Ereignis. Über das MessagingCenter wird eine Nachricht an das portable Projekt geschickt, damit dieses den Zustandswechsel mitbekommt und darauf reagieren kann.

6 Software-Ausblick

Nachfolgender Teil ist noch in Bearbeitung durch Andreas Grain

6.1 Gesicherte Verbindung

6.2 Weitere mobile Plattformen

Neben dem ausprogrammierten Android-System gibt es noch das weit verbreitete iOS der Apple-Geräte. Mit Xamarin.Forms lässt sich dieses verglichen mit anderen Entwicklungsmethoden einfach unterstützen. Es muss lediglich der plattformspezifische Teil auf die neue Plattform portiert werden, während das portable Projekt unverändert bleibt. Dies bedeutet, dass analog zum PiBell.Android ein Projekt PiBell.iOS geschrieben werden muss, um die Plattform zu unterstützen. In diesem Projekt müssen alle Funktionen des Android-Teils repliziert werden. Wichtig für die Entwicklung ist auch, dass die Endgeräte zum Übertragen und Testen verfügbar sind. Im Fall der iOS-Plattform wird ein Apple Mac für die Übertragung und ein Apple iPhone zum Testen benötigt.

Für Testzwecke kann die Applikation auf bis zu fünf iPhones installiert und getestet werden. Wenn die App schlussendlich veröffentlicht werden soll, ist eine kostenpflichtige Mitgliedschaft beim Apple Developer Program notwendig. Diese beläuft sich auf etwa USD 99 pro Jahr [vgl. 13, Integrated Development Environment (IDE) Availability]

Teil II

Hardware - Matthias Mair

7 Technische Grundlagen

Die im Dokument verwendeten Spannungsangaben sind immer Gleichspannung (DC) und werden nicht jedes Mal ausdrücklich als solche gekennzeichnet. Sollten abweichende Spannungsangaben zur Anwendung kommen, so wird dies explizit angeführt.
Nachfolgender Teil ist noch in Bearbeitung durch Matthias Mair

7.1 Power over Ethernet

7.2 Spannungsregler

7.3 Verstärker

7.3.1 Lautsprecher

7.3.2 Mikrofon

7.4 SMT

Vorteile:

Nachteile:

7.4.1 Fehler

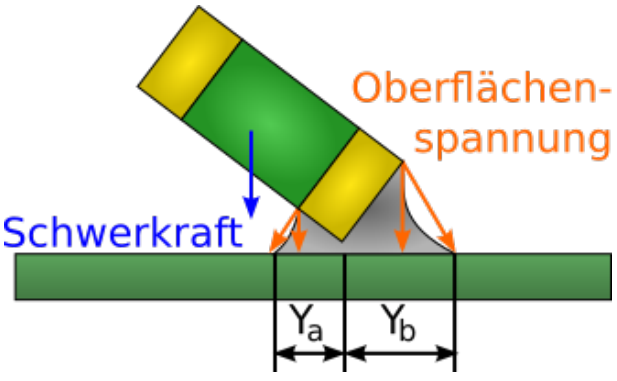


Abbildung 7.1: Grabstein-Effekt

Grabstein-Effekt:

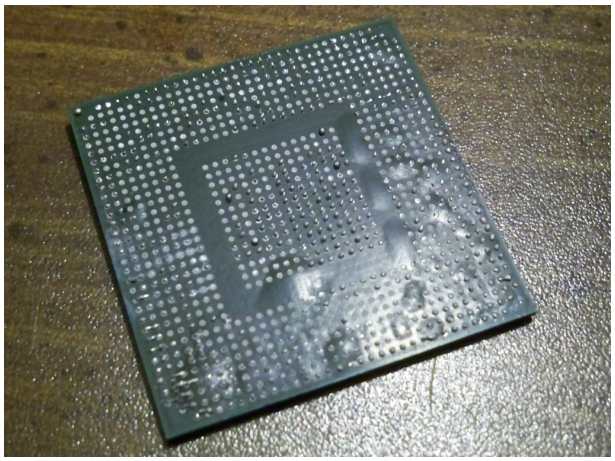


Abbildung 7.2: Popcorn-Effekt

Popcorn-Effekt:

7.5 UART

7.6 SPI

8 Ausgangssituation

8.1 Hardware-Konfiguration

Die aktuelle Hardware der Station setzt sich aus mehreren Einzel-Platinen mit unterschiedlichen Betriebsspannungen zusammen:

- Mikrofon-Vorverstärker-Platine, die das Signal des Kondensator-Mikrofons verstärkt, bevor es zum RPi weitergeleitet wird. Diese Platine wird mit einer Spannung von 5VDC betrieben.
- Lautsprecher-Verstärker-Platine, die für die Verstärkung der wiederzugebenden Audiosignale des RPi zuständig ist. Die verstärkten Signale werden direkt an den Lautsprecher der Station übergeben. Diese Platine benötigt eine Versorgungsspannung von 24VDC.

Die elektrische Versorgung der Station erfolgt mittels PoE bei einer Speisespannung von etwa 30VDC. Alle Platinen sind mit Heißkleber in der Station montiert.

8.2 Problemdefinition

Systemstabilität: In der aktuellen Konfiguration kommt es immer wieder zu unvorhersehbaren und unkontrollierten Systemabstürzen. Die betroffene Station reagiert weder auf eingehende Signale, noch erfolgt irgendeine Form der Informationsausgabe. Dieser Zustand lässt sich nur durch ein Zurücksetzen des Prozessors beenden. Die Abstürze sind bei der am weitesten von der Spannungsversorgung entfernten Station am häufigsten zu beobachten. Dieser Fehler tritt in unregelmäßigen Abständen von bis zu mehreren Monaten auf. Die betroffene Station weist keinen Hardware-Unterschied zu den anderen Stationen auf. Daher lässt sich vermuten, dass die Länge und Art der Verkabelung mit diesem Phänomen in Zusammenhang steht.

Komplexer Aufbau: Die aktuelle Hardware-Konfiguration verwendet einige Einzelplatinen, die miteinander über eine fliegende Verdrahtung verbunden sind. Dies reduziert die Übersichtlichkeit des Stationsinneren und birgt die Gefahr von fehlerhafter Ver-

drahtung in sich. Durch die fliegende Verdrahtung ist kein einheitliches Erscheinungsbild des Stationsinneren gewährleistet.

Brumm-Schleife: Die aktuelle Verdrahtung führt zu einer Masseschleife. Diese erzeugt ein konstantes Störgeräusch. Über die Verstärkerschaltung führt dies zu einem unerträglichen Brumm-Ton. In der aktuellen Hardware-Konfiguration wird das Problem mit einer Massetrennung nach dem Ausgang des RPi umgangen. Dieser zusätzliche Baustein wurde fliegend verdrahtet und mit Heißkleber befestigt.

Folgend ein Bild der aktuellen Station und deren Verdrahtung.

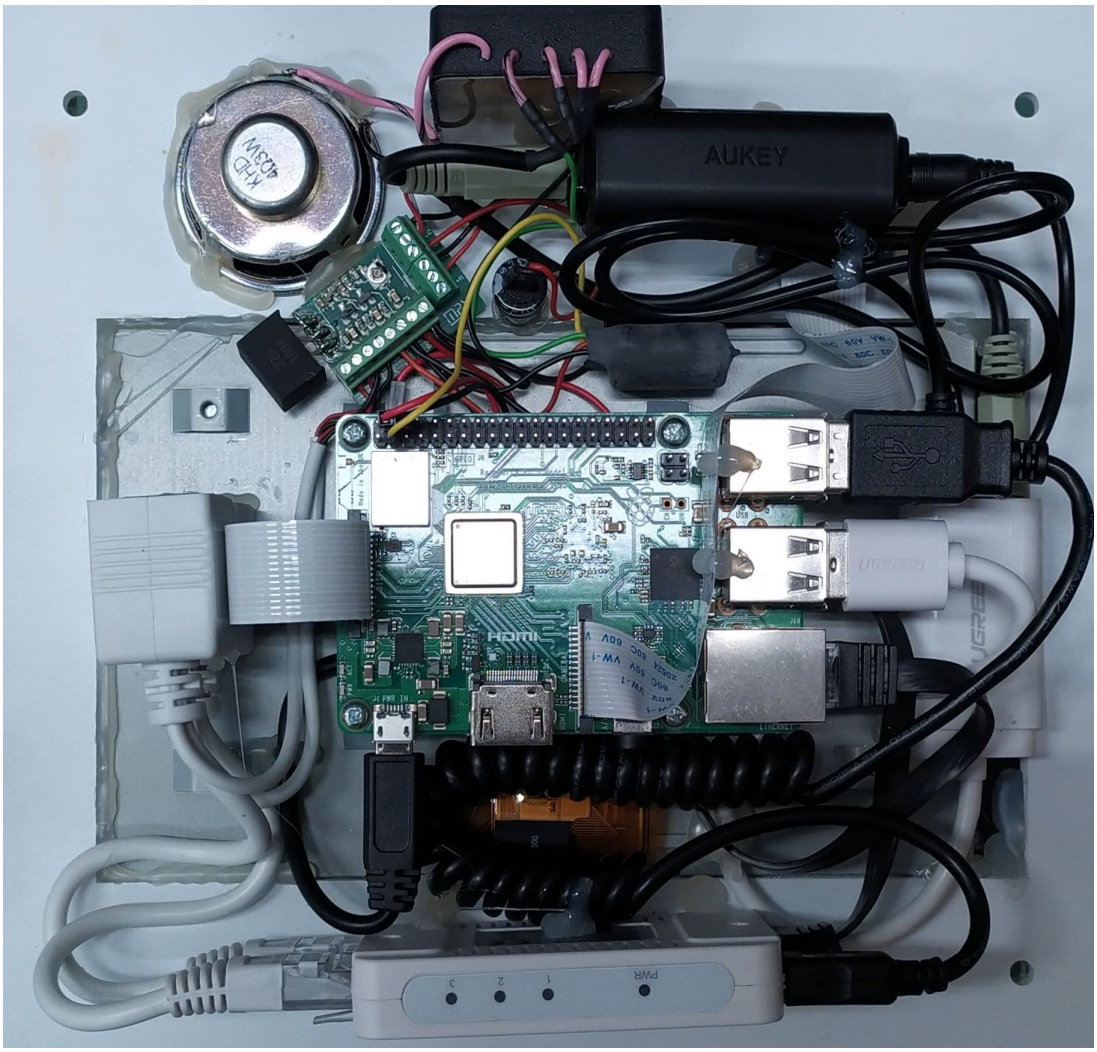


Abbildung 8.1: Unübersichtliche Verdrahtung

9 Berechnung

Nachfolgender Teil ist noch in Bearbeitung durch Matthias Mair

9.1 Spannungsabfall

Herkömmliche CAT5-Ethernetkabel wurden ursprünglich nicht für die Übertragung großer elektrischer Leistungen entwickelt und besitzen daher Leitungen mit einem sehr geringen Ader-Durchmesser. Aufgrund dieser Gegebenheit kann es bei weiten Entfernungen oder großen übertragenen Leistungen zu abnormal hohen Spannungsabfällen kommen. Um die Anlage in dieser Hinsicht zu überprüfen werden sämtliche Spannungsabfälle und die damit verbundenen Verlustleistungen bei maximaler Belastung durch die Stationen berechnet. Zur Berechnung der Spannungsabfälle sind folgende Angaben verwendet worden.

- Es werden vier Stationen über PoE versorgt. Die Stationen sind nach Abbildung *9.1 Verdrahtungsschema* miteinander verdrahtet.
- Jede Station hat einen maximalen Leistungsverbrauch von 12 W.
- Das Ethernetkabel zwischen den Stationen hat jeweils eine Länge von 10 m.
- Der Querschnitt einer Ader des Ethernet-Kabels beträgt 0.128 mm^2 [vgl. 14].
- Am PSE wird eine Spannung von 30 V in das Netz eingespeist.

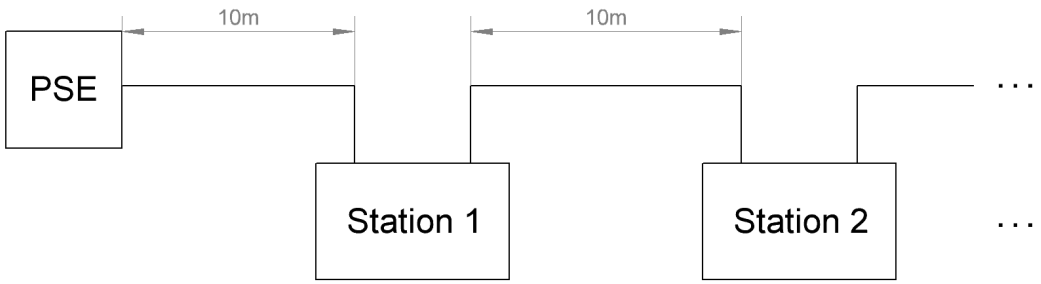


Abbildung 9.1: Verdrahtungsschema

Der Leitungswiderstand eines Adernpaares des Ethernetkabels beträgt nach der allgemeinen Drahtwiderstandsformel $R = \frac{l}{\gamma \cdot A} = 0.686\Omega$. Je nach verwendeter PoE-Type werden mehrere Adernpaare parallel geschaltet, um den Leitungswiderstand zu verringern. Anhand der oben beschriebenen Verdrahtung ergeben sich folgende Ersatzschaltungen:

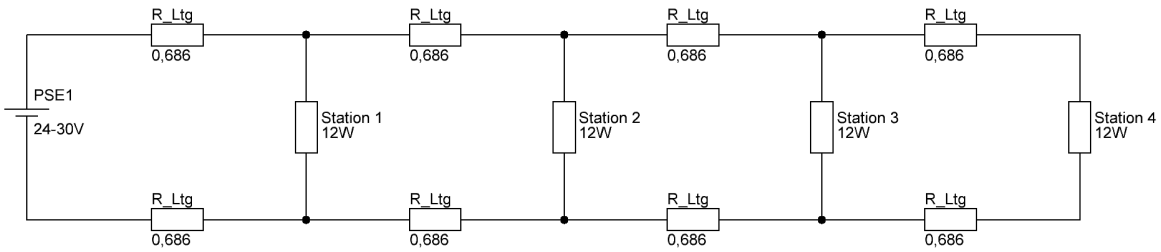


Abbildung 9.2: Ersatzschaltung PoE Type 1 und 2

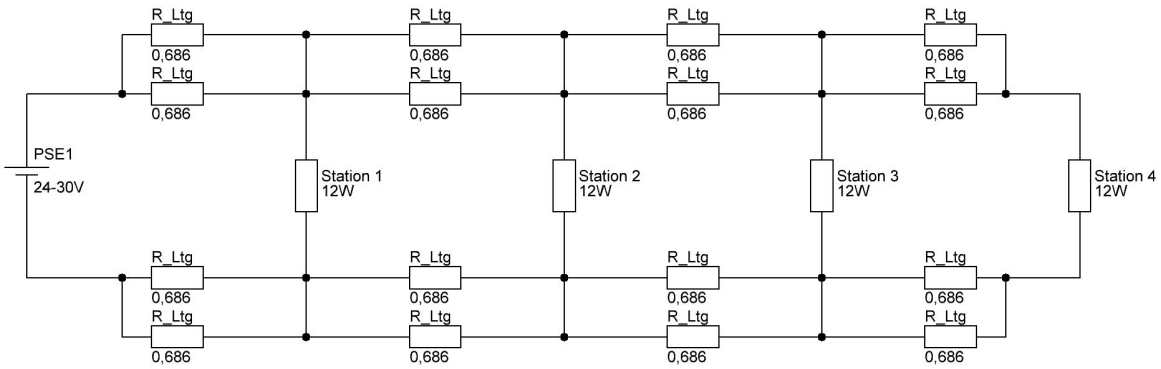


Abbildung 9.3: Ersatzschaltung PoE Type 3 und 4

U_n…Spannung an Station n, I_n…Strom durch Station n, U_0…Quellenspannung;

Anhand dieser Ersatzschaltung ergeben sich folgende Berechnungsformeln:

$$U_1 = U_0 - 2 \cdot R_{Ltg} \cdot (I_1 + I_2 + I_3 + I_4) \tag{9.1}$$
$$U_2 = U_1 - 2 \cdot R_{Ltg} \cdot (I_2 + I_3 + I_4) \tag{9.2}$$
$$U_3 = U_2 - 2 \cdot R_{Ltg} \cdot (I_3 + I_4) \tag{9.3}$$
$$U_4 = U_3 - 2 \cdot R_{Ltg} \cdot I_4 \tag{9.4}$$
$$12W = U_1 \cdot I_1 \tag{9.5}$$
$$12W = U_2 \cdot I_2 \tag{9.6}$$
$$12W = U_3 \cdot I_3 \tag{9.7}$$
$$12W = U_4 \cdot I_4 \tag{9.8}$$

Zur Lösung dieses Gleichungssystems wurde das CAS-System Maxima verwendet. Die Eingabe sieht wie folgt aus:

```
(%i16) u_0: 30$
       r_ltg: 0.686$
       p: 12$
       eqn1: u_1=u_0-2·r_ltg·(i_1+i_2+i_3+i_4)$
       eqn2: u_2=u_1-2·r_ltg·(i_2+i_3+i_4)$
       eqn3: u_3=u_2-2·r_ltg·(i_3+i_4)$
       eqn4: u_4=u_3-2·r_ltg·i_4$
       eqn5: p=u_1·i_1$
       eqn6: p=u_2·i_2$
       eqn7: p=u_3·i_3$
       eqn8: p=u_4·i_4$
       algsys([eqn1,eqn2,eqn3,eqn4,eqn5,eqn6,eqn7,eqn8],[u_1,u_2,u_3,u_4,i_1,i_2,i_3,i_4]);
```

Abbildung 9.4: Eingabe wxMaxima Type 1 & 2

Die Lösungen für das Gleichungssystem werden vom Programm in Textform zurückgegeben. Das Programm findet neben der relevanten Lösung noch einige komplexe Lösungen für das Gleichungssystem. Diese werden hier wegen Platzgründen und niedriger Relevanz weggelassen. Man sieht: Der Spannungsabfall ist wie erwartet bei Type 3 und 4 etwa halb so groß wie bei Type 1 und 2.

	Type 1 & 2		Type 3 & 4	
	U	I	U	I
Station 1	27.35 V	438 mA	28.81 V	416 mA
Station 2	25.30 V	474 mA	27.91 V	430 mA
Station 3	23.90 V	502 mA	27.31 V	439 mA
Station 4	23.19 V	517 mA	27.00 V	444 mA

Tabelle 9.1: Maxima-Ergebnisse Spannungsabfall

10 Verbesserte Hardwarelösung

Nachfolgender Teil ist noch in Bearbeitung durch Matthias Mair

10.1 Konzept

Die neue Lösung sollte den Aufbau einer Station vereinfachen und die Ausfallrate einer Station auf gänzlich null reduzieren. Um dieses Ziel zu erreichen werden folgende Ansätze definiert:

Zusammenfassen der Platinen zu einer einzigen: Die Reduktion der Anzahl der Platinen auf eine Platine ermöglicht die Zusammenfassung von verschiedenen Modulen. Damit werden die Produktionskosten gesenkt und die Montage in den Stationen vereinfacht.

Reduzierung der Bauteile: Die Zusammenfassung der Module auf eine einzelne Platine ermöglicht die Reduktion von Bauteilen. Insbesondere werden Bauteile für die Verbindungstechnik eingespart. Dies führt zu einer Kostenreduktion bei gleichzeitiger Senkung der Fehlermöglichkeiten.

Entfall der Verdrahtung zwischen den Einzelplatinen: Der Entfall der Verdrahtung führt zu einer Steigerung der Übersichtlichkeit und reduziert zusätzlich den Platzbedarf. Aufwände für die händische Verdrahtung und die Prüfung der Verbindungen der einzelnen Platinen entfällt. Fehlverdrahtung wird konstruktiv durch die Einzelplatine verhindert.

Verbesserung der Spannungsversorgung: Leistungsbeschränkungen sowie die Verwendung eines neuen verbesserten Spannungsreglers verbessern die Spannungsqualität. Die verstärkte Spannungsversorgung ermöglicht einen späteren update auf den leistungsstärkeren RPi Model 4.

Einführung WatchDogTimer: Zusätzlich wird ein Watchdog mittels eines Mikroprozessors verbaut. Dieser wird benötigt, um eventuelle Ausfälle des RPi und somit der gesamten Station zu erkennen. Falls ein Fehler auftritt setzt der Mikroprozessor den RPi zurück und die Station startet neu. Nachdem die Station neugestartet ist, wird der

Normalbetrieb wieder aufgenommen und der Watchdog nimmt den Zustand des Überwachens wieder ein.

10.2 Spannungsversorgung

10.2.1 Anforderungen

Um einen möglichst reibungslosen Betrieb zu garantieren, werden an die Spannungsversorgung mehrere Anforderungen gestellt:

- Der Eingangsspannungsbereich muss mindestens 24 bis 35 V betragen.
- Die gewünschte Ausgangsspannung beträgt 5 V.
- Der Oberwellenanteil (ripple) der Ausgangsspannung soll unter 1% liegen.
- Die Strombelastbarkeit muss mindestens 2.5 A betragen.

Die Versorgungsspannung wird über PoE (Power over Ethernet) zur Verfügung gestellt. Der Spannungsregler muss diese Art der Versorgungsspannung unterstützen. Der Spannungsregler sollte die 24V auf 5V möglichst verlustarm reduzieren. Alle versorgten Komponenten auf der Platine arbeiten mit einer Spannung von 5V. Diese Komponenten sind z.B. der RPi, der Mikrocontroller und die Verstärkerschaltung für Mikrofon und Lautsprecher.

Die Spannungsversorgung soll bei Spannungsschwankungen kurzzeitig die Versorgung aufrechterhalten, um dem Versagen einer Station vorzubeugen. Zur Stützung der Spannungsversorgung werden Kondensatoren auf der Eingangsspannungsseite und auf der Ausgangsspannungsseite benötigt.

10.2.2 Auswahl Spannungsregler

Aufgrund der oben angeführten Anforderungen wird der Spannungsregler LM33630 ausgewählt. Dieser Spannungsregler kann im Bereich von 3.8 V bis 36 V betrieben werden. Die Ausgangsspannung ist immer kleiner als die Eingangsspannung und liegt im Bereich von 1 V bis 24 V. Der LM33630 ist ein Step-down Spannungsregler, das bedeutet er kann nur von einer höheren auf eine niedrigere Spannung regeln.

Um eine Schwingung, die wenig Aufwand zur Glättung benötigt, zu erzeugen verwendet der Spannungsregler eine hohe Schaltfrequenz von 2.1 MHz. Der LM33630 hat

standardmäßig einen maximalen Ausgangsstrom von 3 A, diese wird auch nahezu vollständig benötigt für den RPi, der bezieht allein bereits ca. 1.5 A und der Lautsprecher mit der Vorschaltung benötigt auch über 0.5 A.

Vom LM33630 sind mehrere Varianten verfügbar, es gibt die DDA und RNX Variante. Die DDA Variante hat 8 Pins und ein eingebauter Kühlkörper, der mit einen großen Massefeld verbunden werden kann, damit der IC nicht überhitzt. Die RNX Variante hat 12 Pins und anstatt eines größeren Kühlfläche mehrere Ground Pins, die ebenfalls zur Kühlung des ICs dienen. Zusätzlich gibt es eine weitere Unterteilung bei den zwei Varianten, es gibt verschiedene Schaltfrequenzen und diese sind 400, 1400 und 2100 kHz.

Die technischen Daten dieses Spannungsreglers sind in vollem Umfang im Datenblatt ersichtlich. [vgl. 15]

10.2.3 Funktion des Spannungsreglers

Ein Step-down Spannungsregler arbeitet mit einen PWM Signal, das anschließend mittels einer Induktivität geglättet wird.

Bei der Pulsweitenmodulation (engl. Pulse Width Modulation, abgekürzt PWM) wird das Verhältnis zwischen der Einschaltzeit und Periodendauer eines Rechtecksignals bei fester Grundfrequenz variiert. Das Verhältnis zwischen der Einschaltzeit tein und der Periodendauer T=tein+taus wird als das Tastverhältnis p bezeichnet. (laut DIN-IEC 60469-1: Tastgrad) (engl. Duty Cycle, meist abgekürzt DC, nicht zu verwechseln mit Direct Current = Gleichstrom). [16, Einleitung]

Über einen Spannungsteiler kann die Ausgangsspannung variiert werden. Weiters kann der Spannungsregler abgeschaltet werden, wenn 0V am Enable Pin angelegt werden. Dies sorgt für eine komplette Abschaltung der Steuerlogik. Die Verwendung dieser Funktion ist jedoch nicht vorgesehen.

10.2.4 Zusatzbeschaltung des Spannungsreglers

Der Spannungsregler ist universell einsetzbar, weshalb die genauen Betriebseigenschaften über die externe Beschaltung festgelegt werden.

Eingangskondensator: Auf der Eingangsseite, d.h. der 24V Seite wird ein 820uF Kondensator verbaut. Dieser dient in erster Linie zum Abfangen und Ausgleichen von

Spannungsschwankungen. Diese Spannungsschwankungen können aufgrund von sich schlagartig ändernden Leistungsverbräuchen entstehen.

Ausgangskondensator: Nach dem Spannungsregler auf der 5V Seite werden zwei Kondensatoren mit jeweils der Kapazität von 1mF verbaut, das ergibt eine Gesamtkapazität auf der 5V Seite von 2mF. Diese werden zur nahezu vollständigen Glättung der Ausgangsspannung und weiteres Abfangen von Spannungseinbrüchen wegen Leistungsspitzen verwendet.

Spannungsteiler: Der Spannungsteiler (100kOhm; 23,9kOhm) am FB Pin sorgt für die richtige Ausgangsspannung von 5V. Dieser Spannungsteiler kann frei über die folgende Formel konfiguriert werden:

$$R_{FBB} = \frac{R_{FBT}}{\frac{V_{OUT}}{V_{REF}} - 1}$$

Im Datenblatt des Spannungsreglers sind bereits Werte für die am häufigsten verwendeten Spannungen vorgegeben.

Kühlung: Der Spannungsregler muss gekühlt werden, da bei ihm eine nicht vernachlässigbare Verlustleistung auftritt. Unter dem IC ist eine große Massenfläche bereitgestellt, die zur Kühlung dient. Die Kühlfläche des Spannungsreglers wird mit Durchkontaktierungen (Vias) mit der Massefläche auf der Unterseite der Platine verbunden und somit gekühlt.

10.3 Mikrofon-Vorverstärker

10.4 Lautsprecher-Verstärker

10.5 Mikrocontroller

11 Verwendete Software

Nachfolgender Teil ist noch in Bearbeitung durch Matthias Mair

11.1 EAGLE

11.2 Library Loader

12 Hardware-Dokumentation

Nachfolgender Teil ist noch in Bearbeitung durch Matthias Mair

12.1 Schaltung

12.2 Platinen-Design

Literatur und Quellen

[1] Adobe Systems Incorporated. (Okt. 2011). „Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap“, Adresse: https://news.adobe.com/press-release/adobe-creative-cloud-dps/adobe-announces-agreement-acquire-nitobi-creator-phonegap.

[2] Microsoft. (März 2020). „Compare Visual Studio 2019 Editions“, Adresse: https://visualstudio.microsoft.com/vs/compare/.

[3] ——, (Dez. 2019). „Install Visual Studio“, Adresse: https://docs.microsoft.com/en-us/visualstudio/install/install-visual-studio?view=vs-2019.

[4] J. Piiroinen. (Feb. 2008). „Overview of the Common Language Infrastructure“, Wikimedia Commons, Adresse: https://commons.wikimedia.org/wiki/File:Overview_of_the_Common_Language_Infrastructure.svg.

[5] H. Schulzrinne (Columbia University), A. Rao (Netscape Communications) und R. Lanphier (RealNetworks). (Apr. 1998). „RFC 2326 - Real Time Streaming Protocol (RTSP)“, Internet Engineering Task Force, Adresse: https://tools.ietf.org/html/rfc2326.

[6] H. Schulzrinne (Columbia University), A. Rao (Cisco Networks), R Lanphier, M. Westerlund (Ericsson) und M. Stiemerling, Ed. (University of Applied Sciences Darmstadt). (Dez. 2016). „RFC 7826 - Real-Time Streaming Protocol Version 2.0“, Internet Engineering Task Force, Adresse: https://tools.ietf.org/html/rfc7826.

[7] J.-B. Kempf. (Feb. 2016). „15 years of VLC and VideoLAN“, Adresse: http://www.jbkempf.com/blog/post/2016/15-years-of-VLC.

[8] Josh Smith (Microsoft). (Dez. 2016). „Patterns - WPF Apps With The Model-View-ViewModel Design Pattern“, Adresse: https://docs.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern.

[9] Martin Finkel (VideoLAN). (März 2020). „LibVLCSharp README.md“, Adresse: https://code.videolan.org/videolan/LibVLCSharp/-/blob/3.x/README.md.

[10] VAStreaming. (März 2020). „highly optimized and efficient media streaming, encoding, decoding and processing libraries for developers“, Adresse: https://www.vastreaming.net/.

[11] GStreamer. (März 2020). „GStreamer: a flexible, fast and multiplatform multimedia framework“, Adresse: https://gstreamer.freedesktop.org/documentation/?gi-language=c.

[12] Google. (März 2020). „Google Trends - LIVE555“, Adresse: https://trends.google.com/trends/explore?date=all&q=%2Fm%2F0641r5r.

[13] Microsoft. (März 2017). „Part 1 – Understanding the Xamarin Mobile Platform“, Adresse: https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/building-cross-platform-applications/understanding-the-xamarin-mobile-platform.

[14] Lapp Gruppe. (Sep. 2018). „Datenblatt F/UTP Patchkabel CAT.5e“, Adresse: https://www.lappkabel.de/fileadmin/documents/technische_doku/datenblaetter/ETHERLINE/Etherline_Leitungen/DBCE6547DE.pdf.

[15] Texas Instruments. (März 2019). „LMR33630 Datasheet“, Adresse: https://www.ti.com/lit/ds/symlink/lmr33630.pdf.

[16] Mikrocontroller.net. (März 2020). „Pulsweitenmodulation“, Adresse: https://www.mikrocontroller.net/articles/Pulsweitenmodulation.

[17] Atmel. (Jan. 2015). „ATmega328P Datasheet“, Adresse: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf.

[18] Diodes. (Nov. 2012). „PAM8403 Datasheet“, Adresse: https://www.diodes.com/assets/Datasheets/PAM8403.pdf.

Abkürzungen

APK	Android Package
CIL	Common Interface Language
CLR	Common Language Runtime
CPU	Central processing unit
CSS	Cascading Style Sheets
.NET	.NET-Framework
EAGLE	Easily Applicable Graphical Layout Editor
FI	Fachspezifische Softwaretechnik
HTTP	Hypertext Transfer Protocol
IC	Integrated Circuit
IETF	Internet Engineering Task Force
PCB	Printed Circuit Board
PoE	Power over Ethernet
RPi	Raspberry Pi
RTSP	Real Time Streaming Protocol
SDK	Software Development Kit
SMD	Surface-mounted Device
THT	Through-Hole Technology
UI	User interface
URL	Uniform Resource Locator
UWP	Universal Windows Platform
WYSIWYM	What You See Is What You Mean
XAML	Extensible Application Markup Language
XML	Extensible Markup Language

Abbildungsverzeichnis

Alle Abbildungen ohne Referenzmarke „[#]“ sind Eigenfotos

Tabellenverzeichnis

Teil III

Appendix

A Datenblätter

A.1 Mikrocontroller Atmel ATmega328P

Nachfolgend ein Auszug aus dem Datenblatt mit den wichtigsten technischen Daten, der Pinbelegung und einem Blockdiagramm. Die vollständige Dokumentation umfasst über 294 Seiten und kann unter nachfolgender Adresse abgerufen werden.

Atmel. (Jan. 2015). „ATmega328P Datasheet“, Adresse: `http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf`

ATmega328P

8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash

DATASHEET

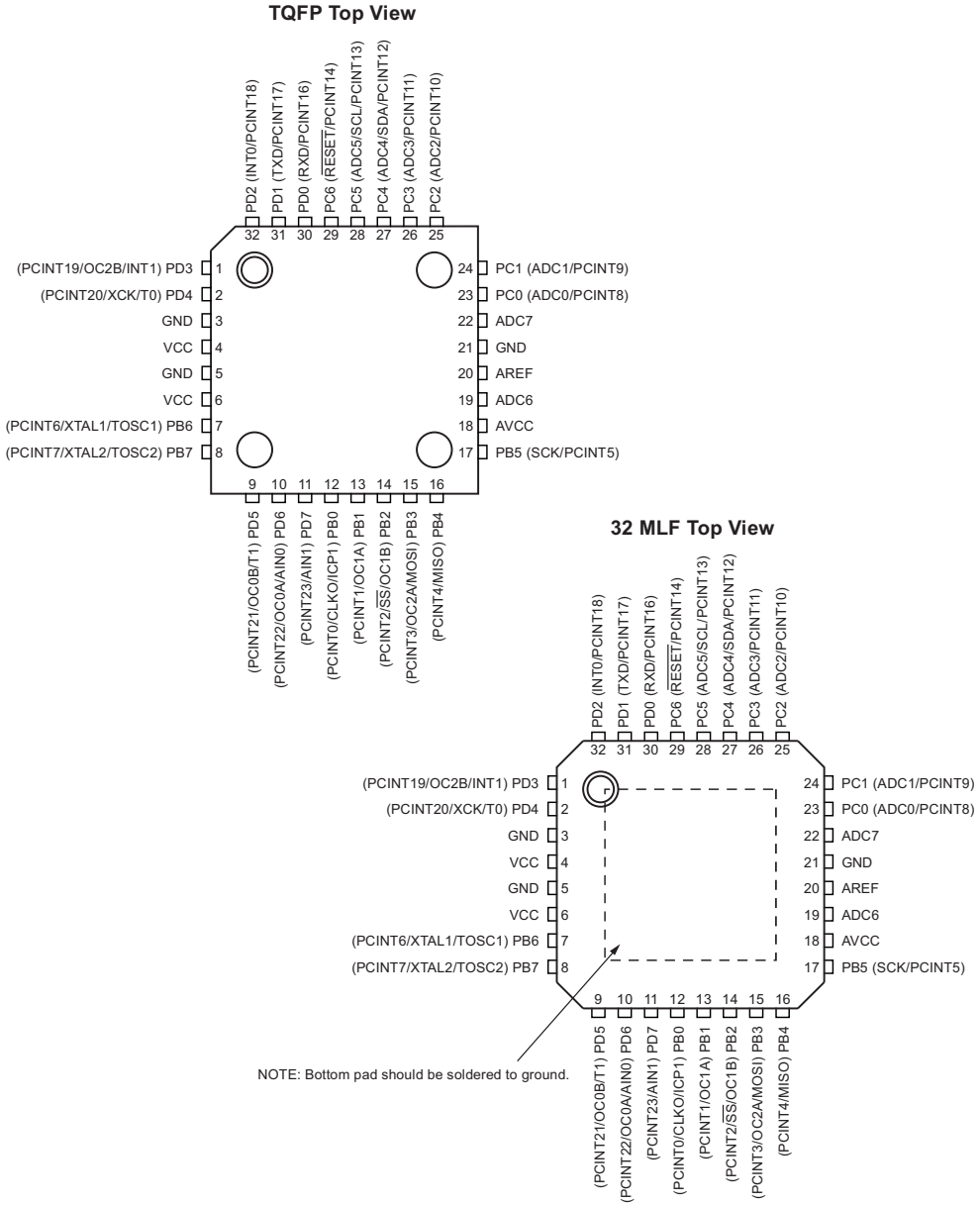
Features

- High performance, low power AVR® 8-bit microcontroller
- Advanced RISC architecture
  - 131 powerful instructions – most single clock cycle execution
  - 32 × 8 general purpose working registers
  - Fully static operation
  - Up to 16MIPS throughput at 16MHz
  - On-chip 2-cycle multiplier
- High endurance non-volatile memory segments
  - 32K bytes of in-system self-programmable flash program memory
  - 1Kbytes EEPROM
  - 2Kbytes internal SRAM
  - Write/erase cycles: 10,000 flash/100,000 EEPROM
  - Optional boot code section with independent lock bits
    - In-system programming by on-chip boot program
    - True read-while-write operation
  - Programming lock for software security
- Peripheral features
  - Two 8-bit Timer/Counters with separate prescaler and compare mode
  - One 16-bit Timer/Counter with separate prescaler, compare mode, and capture mode
  - Real time counter with separate oscillator
  - Six PWM channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature measurement
  - Programmable serial USART
  - Master/slave SPI serial interface
  - Byte-oriented 2-wire serial interface (Phillips I^2C compatible)
  - Programmable watchdog timer with separate on-chip oscillator
  - On-chip analog comparator
  - Interrupt and wake-up on pin change
- Special microcontroller features
  - Power-on reset and programmable brown-out detection
  - Internal calibrated oscillator
  - External and internal interrupt sources
  - Six sleep modes: Idle, ADC noise reduction, power-save, power-down, standby, and extended standby

- I/O and packages
  - 23 programmable I/O lines
  - 32-lead TQFP, and 32-pad QFN/MLF
- Operating voltage:
  - 2.7V to 5.5V for ATmega328P
- Temperature range:
  - Automotive temperature range: –40°C to +125°C
- Speed grade:
  - 0 to 8MHz at 2.7 to 5.5V (automotive temperature range: –40°C to +125°C)
  - 0 to 16MHz at 4.5 to 5.5V (automotive temperature range: –40°C to +125°C)
- Low power consumption
  - Active mode: 1.5mA at 3V - 4MHz
  - Power-down mode: 1µA at 3V

1. Pin Configurations

Figure 1-1. Pinout

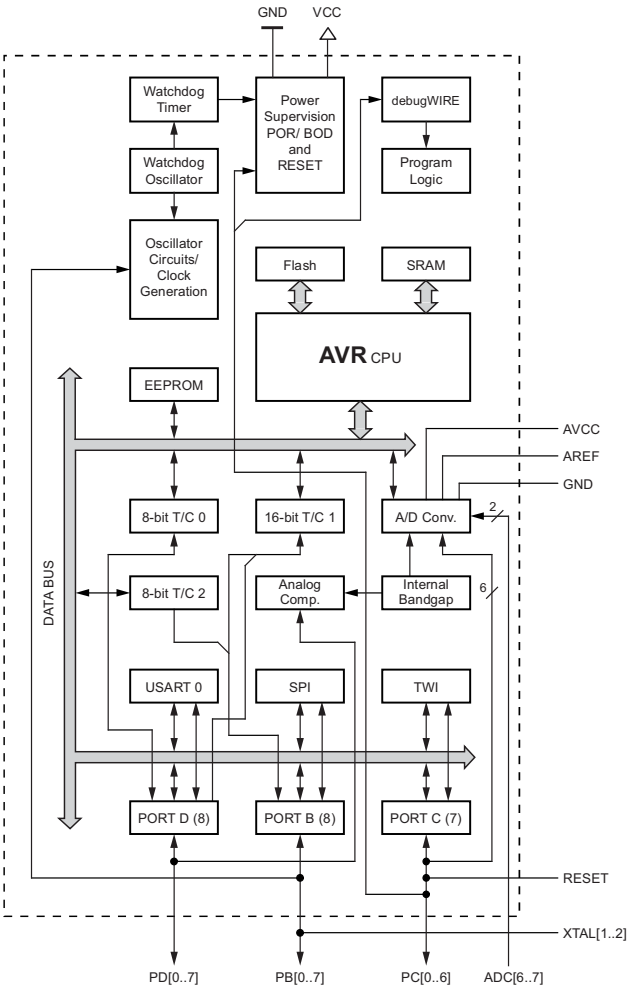


2. Overview

The Atmel® ATmega328P is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega328P achieves throughputs approaching 1MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



A.2 Audio-Verstärker PAM8403

Nachfolgend ein Auszug aus dem Datenblatt mit den wichtigsten technischen Daten, der Pinbelegung und einem Blockdiagramm. Die vollständige Dokumentation umfasst über 11 Seiten und kann unter nachfolgender Adresse abgerufen werden.

Diodes. (Nov. 2012). „PAM8403 Datasheet“, Adresse: `https://www.diodes.com/assets/Datasheets/PAM8403.pdf`

PAM8403

FILTERLESS 3W CLASS-D STEREO AUDIO AMPLIFIER

Description

The PAM8403 is a 3W, class-D audio amplifier. It offers low THD+N, allowing it to achieve high-quality sound reproduction. The new filterless architecture allows the device to drive the speaker directly, requiring no low-pass output filters, thus saving system cost and PCB area.

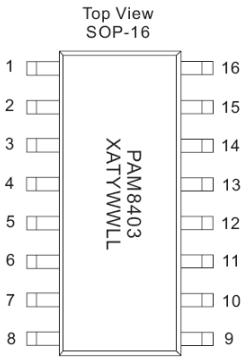
With the same numbers of external components, the efficiency of the PAM8403 is much better than that of Class-AB cousins. It can extend the battery life, which makes it well-suited for portable applications.

The PAM8403 is available in SOP-16 package.

Features

- 3W Output at 10% THD with a 4Ω Load and 5V Power Supply
- Filterless, Low Quiescent Current and Low EMI
- Low THD+N
- Superior Low Noise
- Efficiency up to 90%
- Short Circuit Protection
- Thermal Shutdown
- Few External Components to Save the Space and Cost
- Pb-Free Package

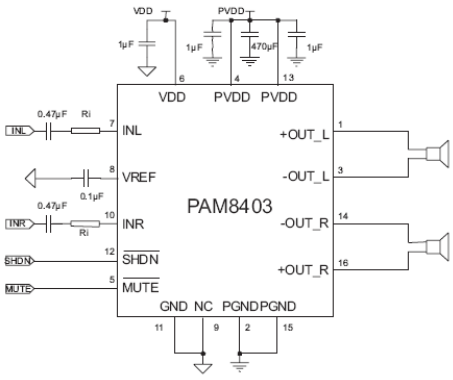
Pin Assignments

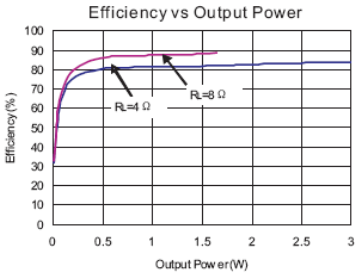


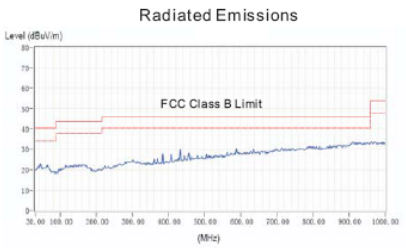
Applications

- LCD Monitors / TV Projectors
- Notebook Computers
- Portable Speakers
- Portable DVD Players, Game Machines
- Cellular Phones/Speaker Phones

Typical Applications Circuit



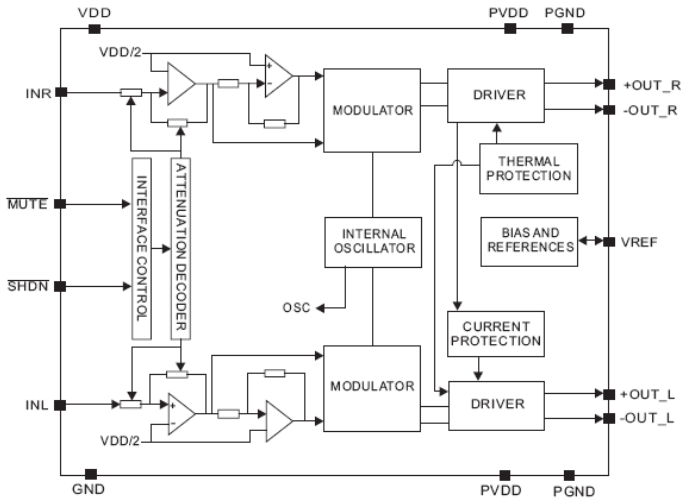




Pin Descriptions

Pin Number	Pin Name	Function
1	+OUT_L	Left Channel Positive Output
2	PGND	Power GND
3	-OUT_L	Left Channel Negative Output
4	PVDD	Power VDD
5	MUTE	Mute Control Input (active low)
6	VDD	Analog VDD
7	INL	Left Channel Input
8	VREF	Internal analog reference, connect a bypass capacitor from VREF to GND.
9	NC	No Connact
10	INR	Right Channel Input
11	GND	Analog GND
12	SHND	Shutdown Control Input (active low)
13	PVDD	Power VDD
14	-OUT_R	Right Channel Negative Output
15	PGND	Power GND
16	+OUT_R	Right Channel Positive Output

Functional Block Diagram



Absolute Maximum Ratings (@T_A = +25°C, unless otherwise specified.)

These are stress ratings only and functional operation is not implied. Exposure to absolute maximum ratings for prolonged time periods may affect device reliability. All voltages are with respect to ground.

Parameter	Rating	Unit
Supply Voltage	6.0	V
Input Voltage	-0.3 to V_{DD} +0.3V	
Operation Temperature Range	-40 to +85	°C
Maximum Junction Temperature	150	
Operation Junction Temperature	-40 to +125	
Storage Temperature	-65 to +150	
Soldering Temperature	300, 5 sec	

Recommended Operating Conditions (@T_A = +25°C, unless otherwise specified.)

Parameter	Rating	Unit
Supply Voltage Range	2.5 to 5.5	V
Operation Temperature Range	-40 to +85	°C
Junction Temperature Range	-40 to +125	°C

Thermal Information

Parameter	Package	Symbol	Max	Unit
Thermal Resistance (Junction to Ambient)	SOP-16	θ_{JA}	110	°C/W
Thermal Resistance (Junction to Case)	SOP-16	θ_{JC}	23	

Electrical Characteristics (@T_A = +25°C, V_{DD} = 5V, Gain = 24dB, R_L = 8Ω, unless otherwise specified.)

Symbol	Parameter	Test Conditions		Min	Typ	Max	Units
V_{DD}	Supply Voltage			2.5		5.5	V
P_O	Output Power	THD+N = 10%, f = 1KHz, R_L = 4Ω	V_{DD} = 5.0V		3.2		W
			V_{DD} = 3.6V		1.6		
			V_{DD} = 3.2V		1.3		
		THD+N = 1%, f = 1KHz, R_L = 4Ω	V_{DD} = 5.0V		2.5		W
			V_{DD} = 3.6V		1.3		
			V_{DD} = 3.2V		0.85		
		THD+N = 10%, f = 1KHz, R_L = 8Ω	V_{DD} = 5.0V		1.8		W
			V_{DD} = 3.6V		0.9		
			V_{DD} = 3.2V		0.6		
		THD+N = 1%, f = 1KHz, R_L = 8Ω	V_{DD} = 5.0V		1.4		W
			V_{DD} = 3.6V		0.72		
			V_{DD} = 3.2V		0.45		
THD+N	Total Harmonic Distortion Plus Noise	V_{DD} = 5.0V, P_O = 1W, R_L = 8Ω	f = 1kHz		0.15		%
		V_{DD} = 3.6V, P_O = 0.1W, R_L = 8Ω			0.11		
		V_{DD} = 5.0V, P_O = 0.5W, R_L = 4Ω	f = 1kHz		0.15		%
		V_{DD} = 3.6V, P_O = 0.2W, R_L = 4Ω			0.11		
G_V	Closed Loop Gain	V_{DD} = 3V to 5V			24		dB
PSRR	Power Supply Ripple Rejection	V_{DD} = 5.0V, Inputs AC-Grounded with C_{IN} = 0.47μF	f = 100Hz		-59		dB
			f = 1kHz		-58		
C_S	Crosstalk	V_{DD} = 5.0V, P_O = 0.5W, R_L = 8Ω, G_V = 20db	f = 1kHz		-95		dB
SNR	Signal-to-Noise Ratio	V_{DD} = 5.0V, V_{ORMS} = 1V, G_V = 20db	f = 1kHz		80		dB
V_N	Output Noise	V_{DD} = 5.0V, Inputs AC-Grounded with C_{IN} = 0.47μF	No A-Weighting		100		μV
			A-Weighting		150		
Dyn	Dynamic Range	V_{DD} = 5.0V, THD = 1%	f = 1kHz		90		dB
η	Efficiency	R_L = 8Ω, THD = 10%	f = 1kHz		87		%
		R_L = 4Ω, THD = 10%			83		
I_Q	Quiescent Current	V_{DD} = 5.0V	No Load		16		mA
		V_{DD} = 3.6V			10		
		V_{DD} = 3.0V			8		
I_{MUTE}	Muting Current	V_{DD} = 5.0V	V_{MUTE} = 0.3V		3.5		mA
I_{SD}	Shutdown Current	V_{DD} = 2.5V to 5.5V	V_{SD} = 0.3V		< 1		μA
$R_{DS(ON)}$	Static Drain-to-Source On-State Resistor	I_{DS} = 500mA, V_{GS} = 5V	PMOS		180		mΩ
			NMOS		140		
f_{SW}	Switching Frequency	V_{DD} = 3.0V to 5.0V			260		kHz
V_{OS}	Output Offset Voltage	V_{IN} = 0V, V_{DD} = 5.0V			10		mV
V_{IH}	Enable Input High Voltage	V_{DD} = 5.0V		1.5	1.4		V
V_{IL}	Enable Input Low Voltage	V_{DD} = 5.0V			0.7	0.4	
V_{IH}	MUTE Input High Voltage	V_{DD} = 5.0V		1.5	1.4		V
V_{IL}	MUTE Input Low Voltage	V_{DD} = 5.0V			0.7	0.4	
OTP	Over Temperature Protection	No Load, Junction Temperature	V_{DD} = 5.0V		140		V
OTH	Over Temperature Hysterisis				30		V

A.3 Spannungsregler LMR33630CDDAR

Nachfolgend ein Auszug aus dem Datenblatt mit den wichtigsten technischen Daten, der Pinbelegung und einem Blockdiagramm. Die vollständige Dokumentation umfasst über 11 Seiten und kann unter nachfolgender Adresse abgerufen werden.

Texas Instruments. (März 2019). „LMR33630 Datasheet“, Adresse: `https://www.ti.com/lit/ds/symlink/lmr33630.pdf`

LMR33630 SIMPLE SWITCHER® 3.8-V to 36-V, 3-A synchronous step-down voltage converter

1 Features

- Configured for rugged industrial applications
  - Input voltage range: 3.8 V to 36 V
  - Output voltage range: 1 V to 24 V
  - Output current: 3 A
  - 75-mΩ/50-mΩ R_{DS-ON} power MOSFETs
  - Peak-current-mode control
  - Short minimum on-time of 68 ns
  - Frequency: 400 kHz, 1.4 MHz, 2.1 MHz
  - Junction temperature range –40°C to +125°C
  - Low EMI and low switching noise
  - Integrated compensation network
- High-efficiency solution
  - Peak efficiency > 95%
  - Low shutdown quiescent current of 5 µA
  - Low operating quiescent current of 25 µA
- Flexible system interface
  - Power-good flag and precision enable
- Create a custom design using the LMR33630 with the WEBENCH® Power Designer

2 Applications

- Motor drive systems: drones, AC inverters, VF drives, servos
- Factory and building automation systems: PLC CPU, HVAC control, elevator control
- General purpose wide V_{IN} power supplies

3 Description

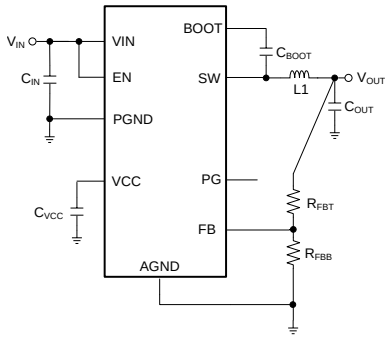
The LMR33630 SIMPLE SWITCHER® regulator is an easy-to-use, synchronous, step-down DC/DC converter that delivers best-in-class efficiency for rugged industrial applications. The LMR33630 drives up to 3 A of load current from an input of up to 36 V. The LMR33630 provides high light load efficiency and output accuracy in a very small solution size. Features such as a power-good flag and precision enable provide both flexible and easy-to-use solutions for a wide range of applications. The LMR33630 automatically folds back frequency at light load to improve efficiency. Integration eliminates most external components and provides a pinout designed for simple PCB layout. Protection features include thermal shutdown, input undervoltage lockout, cycle-by-cycle current limit, and hiccup short-circuit protection. The LMR33630 is available in an 8-pin HSOIC package and in a 12-pin 3 mm × 2 mm VQFN package with wettable flanks. This device is also available in an AEC-Q100-qualified version.

Device Information(1)

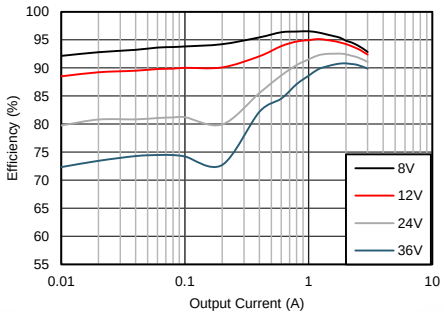
PART NUMBER	PACKAGE	BODY SIZE (NOM)
LMR33630	HSOIC (8)	5.00 mm × 4.00 mm
LMR33630	VQFN (12)	3.00 mm × 2.00 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Simplified Schematic

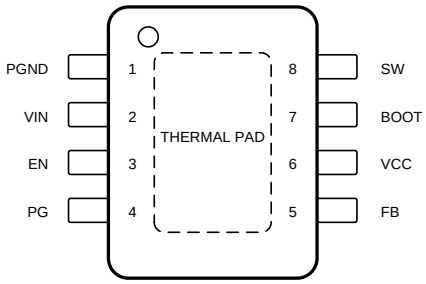


Efficiency vs Output Current
V_{OUT} = 5 V, 400 kHz, VQFN



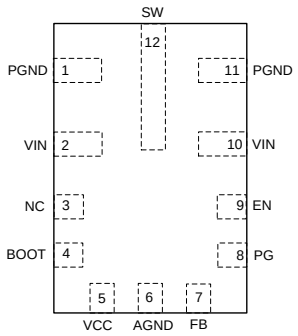
6 Pin Configuration and Functions

DDA Package
8-Pin HSOIC With PowerPAD™
Top View



Not to scale

RNX Package
12-Pin VQFN
Top View



7 Specifications

7.1 Absolute Maximum Ratings

Over the recommended operating junction temperature range(1)

PARAMETER		MIN	MAX	UNIT
Voltages	VIN to PGND	–0.3	38	V
	EN to AGND(2)	–0.3	V_{IN} + 0.3	
	FB to AGND	–0.3	5.5	
	PG to AGND(2)	0	22	
	AGND to PGND	–0.3	0.3	
	SW to PGND	–0.3	V_{IN} + 0.3	V
	SW to PGND less than 100-ns transients	–3.5	38	
	BOOT to SW	–0.3	5.5	
	VCC to AGND(3)	–0.3	5.5	
T_J	Junction temperature(4)	–40	150	°C
T_{stg}	Storage temperature	–55	150	°C

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) The voltage on this pin must not exceed the voltage on the VIN pin by more than 0.3 V
(3) Under some operating conditions the VCC LDO voltage may increase beyond 5.5V.
(4) Operating at junction temperatures greater than 125°C, although possible, degrades the lifetime of the device.

7.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM) (1)	±2500	V
		Charged-device model (CDM) (2)	±750	

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

7.3 Recommended Operating Conditions

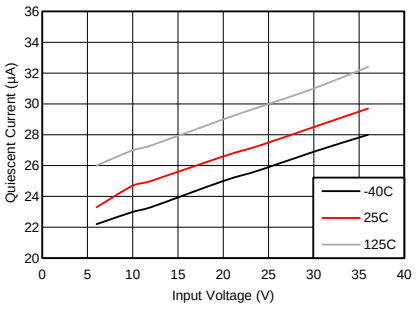
Over the recommended operating temperature range of –40 °C to 125 °C (unless otherwise noted) (1)

		MIN	MAX	UNIT
Input voltage	VIN to PGND	3.8	36	V
	EN (2)	0	V_{IN}	
	PG(2)	0	18	
Adjustable output voltage	V_{OUT}(3)	1	24	V
Output current	I_{OUT}	0	3	A

(1) Recommended operating conditions indicate conditions for which the device is intended to be functional, but do not ensure specific performance limits. For ensured specifications, see *Electrical Characteristics*.
(2) The voltage on this pin must not exceed the voltage on the VIN pin by more than 0.3 V.
(3) The maximum output voltage can be extended to 95% of V_{IN}; contact TI for details. Under no conditions should the output voltage be allowed to fall below zero volts.

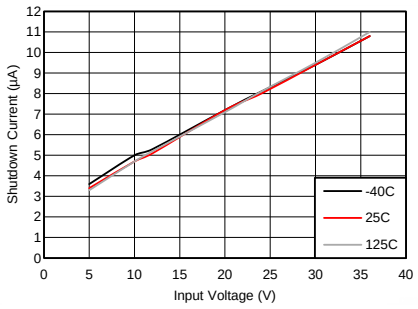
7.8 Typical Characteristics

Unless otherwise specified the following conditions apply: T_A = 25°C and V_{IN} = 12 V



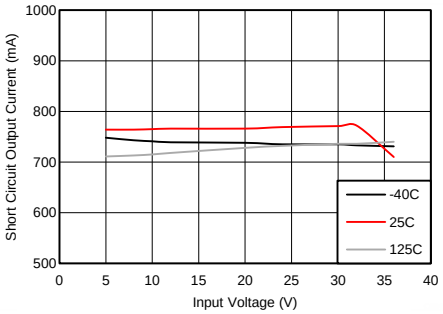
V_{FB} = 1.2 V

Figure 1. Non-Switching Input Supply Current



EN = 0 V

Figure 2. Shutdown Supply Current



V_{OUT} = 0 V f_S = 400 kHz See Figure 50

Figure 3. Short-Circuit Output Current

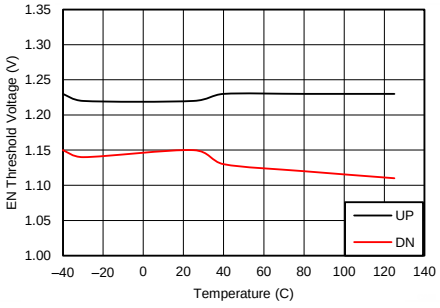
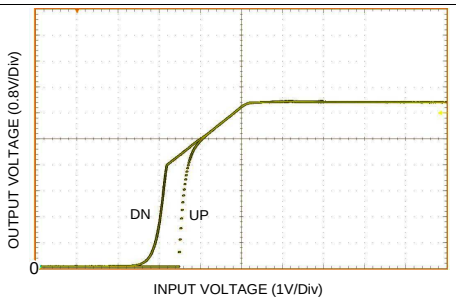
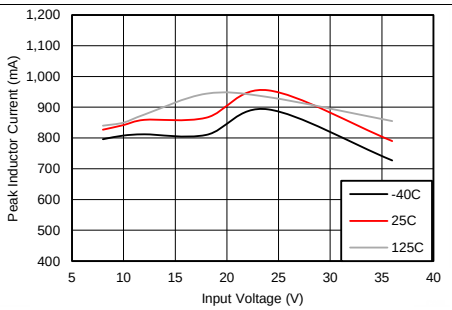


Figure 4. Precision Enable Thresholds



I_{OUT} = 1 mA See Figure 50

Figure 5. UVLO Thresholds



I_{OUT} = 0 A V_{OUT} = 5 V See Figure 50
f_{SW} = 400 kHz

Figure 6. $I_{PEAK-MIN}$

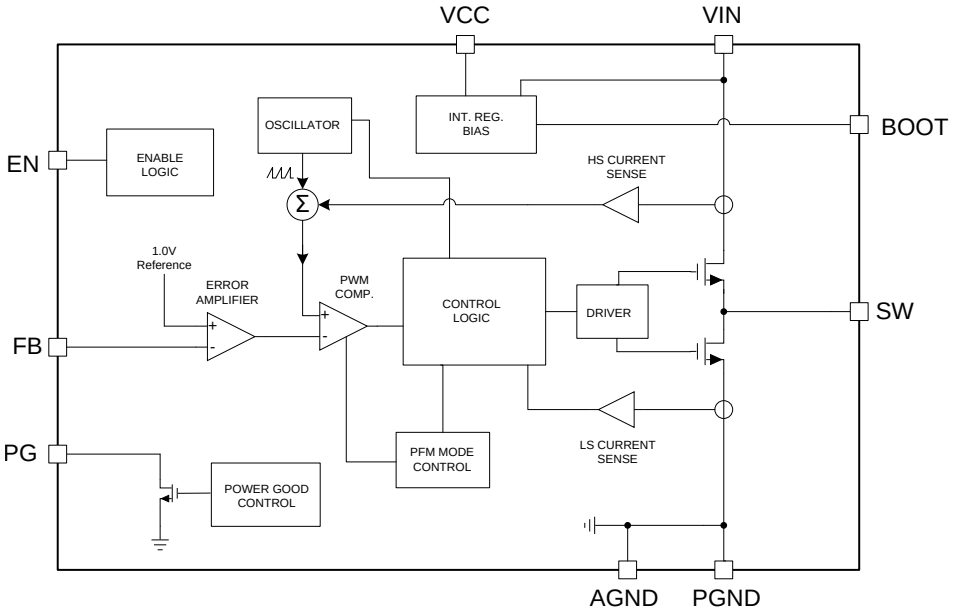
8 Detailed Description

8.1 Overview

The LMR33630 is a synchronous peak-current-mode buck regulator designed for a wide variety of industrial applications. Advanced high speed circuitry allows the device to regulate from an input voltage of 20 V, while providing an output voltage of 3.3 V at a switching frequency of 2.1 MHz. The innovative architecture allows the device to regulate a 3.3 V output from an input of only 3.8 V. The regulator automatically switches modes between PFM and PWM depending on load. At heavy load, the device operates in PWM at a constant switching frequency. At light loads the mode changes to PFM, with diode emulation allowing DCM. This reduces the input supply current and keeps efficiency high. The device features internal loop compensation which reduces design time and requires fewer external components than externally compensated regulators.

The LMR33630 is available in an ultra-miniature VQFN package with wettable flanks. This package features extremely small parasitic inductance and resistance, enabling very high efficiency while minimizing switch node ringing and dramatically reducing EMI. The VIN/PGND pin layout is symmetrical on either side of the VQFN package. This allows the input current magnetic fields to partially cancel, resulting in reduce EMI generation.

8.2 Functional Block Diagram



B Fertigungsdokumentation

B.1 Code-Listing

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell\MainPage.xaml.cs

```
 1 using System;
 2 using System.Collections;
 3 using System.Collections.Generic;
 4 using System.Collections.ObjectModel;
 5 using System.Collections.Specialized;
 6 using System.ComponentModel;
 7 using System.Linq;
 8 using System.Text;
 9 using System.Threading.Tasks;
10 using LibVLCSharp.Shared;
11 using PiBell.Classes;
12 using Xamarin.Forms;
13 using LibVLCSharp.Forms.Shared;
14 using Microsoft.AppCenter.Push;
15 using Microsoft.AppCenter;
16
17 namespace PiBell
18 {
19     public partial class MainPage : ContentPage
20     {
21         private Player _player;
22         private Streamer _streamer;
23         private double _width, _height;
24         private bool _wasConnected;
25         private float _prevPosition;
26
27         public MainPage()
28         {
29             InitializeComponent();
30             RegenerateVideoView();
31             Core.Initialize();
32
33             _player = new Player();
34             _streamer = new Streamer();
35
36             if (!AppCenter.Configured)
37                 Push.PushNotificationReceived += async (sender, e) =>
38                 {
39 #if DEBUG
40                     Console.WriteLine("DEBUG - Push-Notification recieved");
41 #endif
42                     foreach (string key in e.CustomData.Keys)
43                     {
44 #if DEBUG
45                         Console.WriteLine($"DEBUG - Custom Data: {key}:{e.
   CustomData[key]}");
46 #endif
47                         if (key == "mrl")
48                             _player.Mrl = e.CustomData[key].ToString();
49                     }
50
51                     if (await DisplayAlert("Incoming Call", "Connect now?", "
   Yes", "No"))
52                     {
53                         _player.StartCall();
54                         _streamer.StartCall();
55                     }
56                 };
57             MessagingCenter.Subscribe<string>(this, "OnPause", app =>
58             {
```

```
 59                 _wasConnected = _player.MediaPlayer.IsPlaying;
 60                 _player.MediaPlayer.Pause();
 61                 _prevPosition = _player.MediaPlayer.Position;
 62                 _player.MediaPlayer.Stop();
 63 
 64                 MainGrid.Children.Remove(VideoView);
 65             });
 66 
 67             MessagingCenter.Subscribe<string>(this, "OnRestart", app =>
 68             {
 69                 RegenerateVideoView();
 70                 VideoView.MediaPlayer = _player.MediaPlayer;
 71                 if (_wasConnected)
 72                 {
 73                     _player.MediaPlayer.Play();
 74                     _player.MediaPlayer.Position = _prevPosition;
 75                 }
 76 
 77                 _prevPosition = 0;
 78             });
 79 
 80             _player = new Player();
 81             _streamer = new Streamer();
 82             //_player.MediaPlayer.Volume = 0;
 83 
 84             BindingContext = _player;
 85         }
 86 
 87         void RegenerateVideoView()
 88         {
 89             VideoView = new VideoView();
 90             MainGrid.Children.Add(VideoView, 0, 1);
 91         }
 92 
 93         protected override void OnAppearing()
 94         {
 95             base.OnAppearing();
 96             _player.MediaPlayer = new MediaPlayer(_player.LibVlc);
 97         }
 98 
 99         private void BtToggleSpeaker_Clicked(object sender, EventArgs e)
100         {
101             VisualStateManager.GoToState((ImageButton) sender, _player.
    ToggleSpeaker() ? "Unmute" : "Mute");
102         }
103 
104         private void BtToggleMic_Clicked(object sender, EventArgs e)
105         {
106             VisualStateManager.GoToState((ImageButton) sender, _streamer.
    ToggleMic() ? "Unmute" : "Mute");
107         }
108 
109         private void BtConnect_Clicked(object sender, EventArgs e)
110         {
111             _player.StartCall();
112             _streamer.StartCall("192.168.43.78");
113         }
114 
115         private void BtDisconnect_Clicked(object sender, EventArgs e)
116         {
```

```
117             _player.EndCall();
118             _streamer.EndCall();
119         }
120 
121         protected override void OnSizeAllocated(double width, double height)
122         {
123             base.OnSizeAllocated(width, height);
124             if (_width != width || _height != height)
125             {
126                 _width = width;
127                 _height = height;
128                 if (width < height) //Portrait
129                 {
130                     MainGrid.ColumnDefinitions.Clear();
131                     MainGrid.ColumnDefinitions.Add(new ColumnDefinition
132                         {Width = new GridLength(1, GridUnitType.Star)});
133 
134                     MainGrid.RowDefinitions.Clear();
135                     MainGrid.RowDefinitions.Add(new RowDefinition
136                         {Height = new GridLength(0.7, GridUnitType.Star)});
137                     MainGrid.RowDefinitions.Add(new RowDefinition
138                         {Height = new GridLength(3, GridUnitType.Star)});
139                     MainGrid.RowDefinitions.Add(new RowDefinition
140                         {Height = new GridLength(2, GridUnitType.Star)});
141 
142                     Grid.SetColumnSpan(EditMrl, 1);
143                     Grid.SetColumn(ButtonGrid, 0);
144                     Grid.SetRow(ButtonGrid, 2);
145                 }
146                 else //Landscape
147                 {
148                     MainGrid.ColumnDefinitions.Clear();
149                     MainGrid.ColumnDefinitions.Add(new ColumnDefinition
150                         {Width = new GridLength(2, GridUnitType.Star)});
151                     MainGrid.ColumnDefinitions.Add(new ColumnDefinition
152                         {Width = new GridLength(1, GridUnitType.Star)});
153 
154                     MainGrid.RowDefinitions.Clear();
155                     MainGrid.RowDefinitions.Add(new RowDefinition
156                         {Height = new GridLength(0.7, GridUnitType.Star)});
157                     MainGrid.RowDefinitions.Add(new RowDefinition
158                         {Height = new GridLength(5, GridUnitType.Star)});
159 
160                     Grid.SetColumnSpan(EditMrl, 2);
161                     Grid.SetColumn(ButtonGrid, 1);
162                     Grid.SetRow(ButtonGrid, 1);
163                 }
164             }
165         }
166     }
167 }
```

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell.iOS\AudioRecordingService.cs

```
 1 using System;
 2 using System.Collections.Generic;
 3 using System.Linq;
 4 using System.Text;
 5
 6 using UIKit;
 7 using System.Net.Sockets;
 8 using System.Net;
 9 using static PiBell.Classes.Streamer;
10 using System.Threading;
11 using Plugin.AudioRecorder;
12
13 namespace PiBell.iOS
14 {
15     class AudioRecordingService : IAudioRecordingService
16     {
17         Socket socket;
18         NetworkStream stream;
19         Thread recordingThread;
20         AudioRecorderService recorder;
21
22         public AudioRecordingService()
23         {
24             socket = new Socket(SocketType.Dgram, ProtocolType.Udp);
25             stream = new NetworkStream(socket);
26             recorder = new AudioRecorderService();
27             recordingThread = new Thread(RecordingFunction);
28         }
29
30         private void RecordingFunction()
31         {
32             stream = new NetworkStream(socket);
33             recorder.StartRecording();
34             while(true)
35             {
36                 try
37                 {
38                     recorder.GetAudioFileStream().CopyTo(stream);
39                 }
40                 catch(ThreadAbortException)
41                 {
42                     break;
43                 }
44                 catch(Exception e)
45                 {
46                     Console.WriteLine(e.Message);
47                     break;
48                 }
49             }
50             recorder.StopRecording();
51         }
52
53         public void Start(IPEndPoint target)
54         {
55             socket.Connect(target);
56             recordingThread.Start();
57         }
58
59         public void Stop()
60         {
```

```
61              recordingThread.Abort();
62              socket.Close();
63          }
64
65          public bool ToggleMic()
66          {
67              throw new NotImplementedException();
68          }
69      }
70  }
```

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell\Classes\MediaClasses.cs

```
 1 using System;
 2 using System.Collections.Generic;
 3 using System.ComponentModel;
 4 using System.IO;
 5 using System.Net;
 6 using System.Net.Sockets;
 7 using System.Runtime.CompilerServices;
 8 using System.Runtime.InteropServices;
 9 using System.Text;
10 using System.Threading;
11 using LibVLCSharp.Shared;
12 using PiBell.Annotations;
13 using Xamarin.Forms;
14
15 namespace PiBell.Classes
16 {
17     public class Player : INotifyPropertyChanged
18     {
19         public event PropertyChangedEventHandler PropertyChanged;
20
21         [NotifyPropertyChangedInvocator]
22         protected virtual void OnPropertyChanged([CallerMemberName] string
   propertyName = null)
23         {
24             PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(
   propertyName));
25         }
26
27         private bool _speakerActive;
28
29         public Player(string mrl = "https://www.freedesktop.org/software/
   gstreamer-sdk/data/media/sintel_trailer-480p.webm")
30         {
31             LibVlc = new LibVLC();
32             MediaPlayer = new MediaPlayer(LibVlc) { NetworkCaching = 0 };
33             Mrl = mrl;
34         }
35
36         public bool ToggleSpeaker()
37         {
38             MediaPlayer.Volume = _speakerActive ? 0 : 100;
39             return _speakerActive = !_speakerActive;
40         }
41
42         public void StartCall()
43         {
44             MediaPlayer.Media = new Media(LibVlc, Mrl, FromType.FromLocation
   );
45             MediaPlayer.Play();
46             MediaPlayer.Volume = _speakerActive ? 100 : 0;
47         }
48
49         public void EndCall()
50         {
51             MediaPlayer.Stop();
52         }
53
54         private string _mrl;
55         public string Mrl
56         {
```

```
 57              get => _mrl;
 58              set
 59              {
 60                  _mrl = value;
 61                  OnPropertyChanged();
 62              }
 63          }
 64
 65          private LibVLC _libVlc;
 66          public LibVLC LibVlc
 67          {
 68              get => _libVlc;
 69              set
 70              {
 71                  _libVlc = value;
 72                  OnPropertyChanged();
 73              }
 74          }
 75
 76          private MediaPlayer _mediaPlayer;
 77          public MediaPlayer MediaPlayer
 78          {
 79              get => _mediaPlayer;
 80              set
 81              {
 82                  _mediaPlayer = value;
 83                  OnPropertyChanged();
 84              }
 85          }
 86      }
 87
 88      public class Streamer
 89      {
 90          private IAudioRecordingService Service { get; set; }
 91
 92          public Streamer()
 93          {
 94              Service = DependencyService.Get<IAudioRecordingService>();
 95          }
 96
 97          public void StartCall(string targetIp = "127.0.0.1", int targetPort
     = 10000)
 98          {
 99              Service.Start(new IPEndPoint(IPAddress.Parse(targetIp),
    targetPort));
100          }
101
102          public void EndCall()
103          {
104              Service.Stop();
105          }
106
107          public bool ToggleMic()
108          {
109              return Service.ToggleMic();
110          }
111
112          public interface IAudioRecordingService
113          {
114              void Start(IPEndPoint target);
```

```
115             void Stop();
116             bool ToggleMic();
117         }
118     }
119 }
```

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell\App.xaml.cs

```
 1 using System;
 2 using System.Diagnostics;
 3 using LibVLCSharp.Shared;
 4 using Microsoft.AppCenter;
 5 using Microsoft.AppCenter.Analytics;
 6 using Microsoft.AppCenter.Crashes;
 7 using Microsoft.AppCenter.Push;
 8 using Xamarin.Forms;
 9 using Xamarin.Forms.Xaml;
10
11 [assembly: XamlCompilation(XamlCompilationOptions.Compile)]
12
13 namespace PiBell
14 {
15     public partial class App : Application
16     {
17         public App()
18         {
19             InitializeComponent();
20             Core.Initialize();
21             MainPage = new MainPage();
22
23
24             AppCenter.LogLevel = Microsoft.AppCenter.LogLevel.Verbose;
25             AppCenter.Start("android={Your Android App secret here};" +
26                             "uwp={Your UWP App secret here};" +
27                             "ios={Your iOS App secret here}",
28                 typeof(Analytics), typeof(Crashes), typeof(Push));
29         }
30
31         protected override void OnStart()
32         {
33             // Handle when your app starts
34         }
35
36         protected override void OnSleep()
37         {
38             // Handle when your app sleeps
39         }
40
41         protected override void OnResume()
42         {
43             // Handle when your app resumes
44         }
45     }
46 }
```

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell.Android\MainActivity.cs

```
 1 using System;
 2 using Android.App;
 3 using Android.Content.PM;
 4 using Android.Runtime;
 5 using Android.Views;
 6 using Android.Widget;
 7 using Android.OS;
 8 using LibVLCSharp.Forms.Shared;
 9 using LibVLCSharp.Shared;
10 using Xamarin.Forms;
11 using Microsoft.AppCenter.Push;
12 using Microsoft.AppCenter;
13 using Android.Util;
14 using Android;
15 using Android.Support.V4.Content;
16 using Android.Support.V4.App;
17
18 namespace PiBell.Droid
19 {
20     [Activity(Label = "PiBell", Icon = "@mipmap/icon", Theme = "@style/
   MainTheme", MainLauncher = true,
21         ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.
   Orientation)]
22     public class MainActivity : global::Xamarin.Forms.Platform.Android.
   FormsAppCompatActivity
23     {
24         protected override void OnCreate(Bundle savedInstanceState)
25         {
26             TabLayoutResource = Resource.Layout.Tabbar;
27             ToolbarResource = Resource.Layout.Toolbar;
28
29             base.OnCreate(savedInstanceState);
30             global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
31             LoadApplication(new App());
32
33             if (CheckSelfPermission(Manifest.Permission.RecordAudio) !=
   Permission.Granted)
34             {
35                 RequestPermissions(new[] { Manifest.Permission.RecordAudio
    }, 1);
36             }
37         }
38
39         protected override void OnPause()
40         {
41             base.OnPause();
42             MessagingCenter.Send("app", "OnPause");
43         }
44
45         protected override void OnRestart()
46         {
47             base.OnRestart();
48             MessagingCenter.Send("app", "OnRestart");
49         }
50     }
51 }
```

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell.Android\AudioRecordingService.cs

```
 1 using System;
 2 using System.Collections.Generic;
 3 using System.IO;
 4 using System.Linq;
 5 using System.Text;
 6 using System.Threading;
 7 using Android.App;
 8 using Android.Content;
 9 using Android.Media;
10 using Android.OS;
11 using Android.Runtime;
12 using Android.Views;
13 using Android.Widget;
14 using Java.Nio;
15 using Xamarin.Forms;
16 using PiBell.Classes;
17 using System.Net.Sockets;
18 using System.Net;
19
20 [assembly: Dependency(typeof(PiBell.Droid.AudioRecordingService))]
21
22 namespace PiBell.Droid
23 {
24     class AudioRecordingService : Streamer.IAudioRecordingService
25     {
26         private Thread _recordingThread;
27         private volatile byte[] _audioBuffer;
28         private UdpClient _udp;
29         private IPEndPoint _target;
30         private bool _micActive;
31
32         public AudioRecordingService()
33         {
34             _audioBuffer = new byte[AudioRecord.GetMinBufferSize(48000,
   ChannelIn.Mono, Android.Media.Encoding.Pcm16bit) * 3];
35             //_audioRecord = new AudioRecord(AudioSource.Mic, 48000,
   ChannelIn.Mono, Android.Media.Encoding.Pcm16bit, _audioBuffer.Length);
36             _udp = new UdpClient();
37         }
38
39         public void Start(IPEndPoint target)
40         {
41             this._target = target;
42             _recordingThread = new Thread(RecordAudio) { IsBackground = true
    };
43             _recordingThread.Start();
44 #if DEBUG
45             Toast.MakeText(Android.App.Application.Context, "Started
   Recording", ToastLength.Short).Show();
46 #endif
47         }
48
49         private void RecordAudio()
50         {
51             var audioRecord = new AudioRecord(AudioSource.Mic, 48000,
   ChannelIn.Mono, Android.Media.Encoding.Pcm16bit, _audioBuffer.Length);
52             audioRecord.StartRecording();
53             while (true)
54             {
55                 try
```

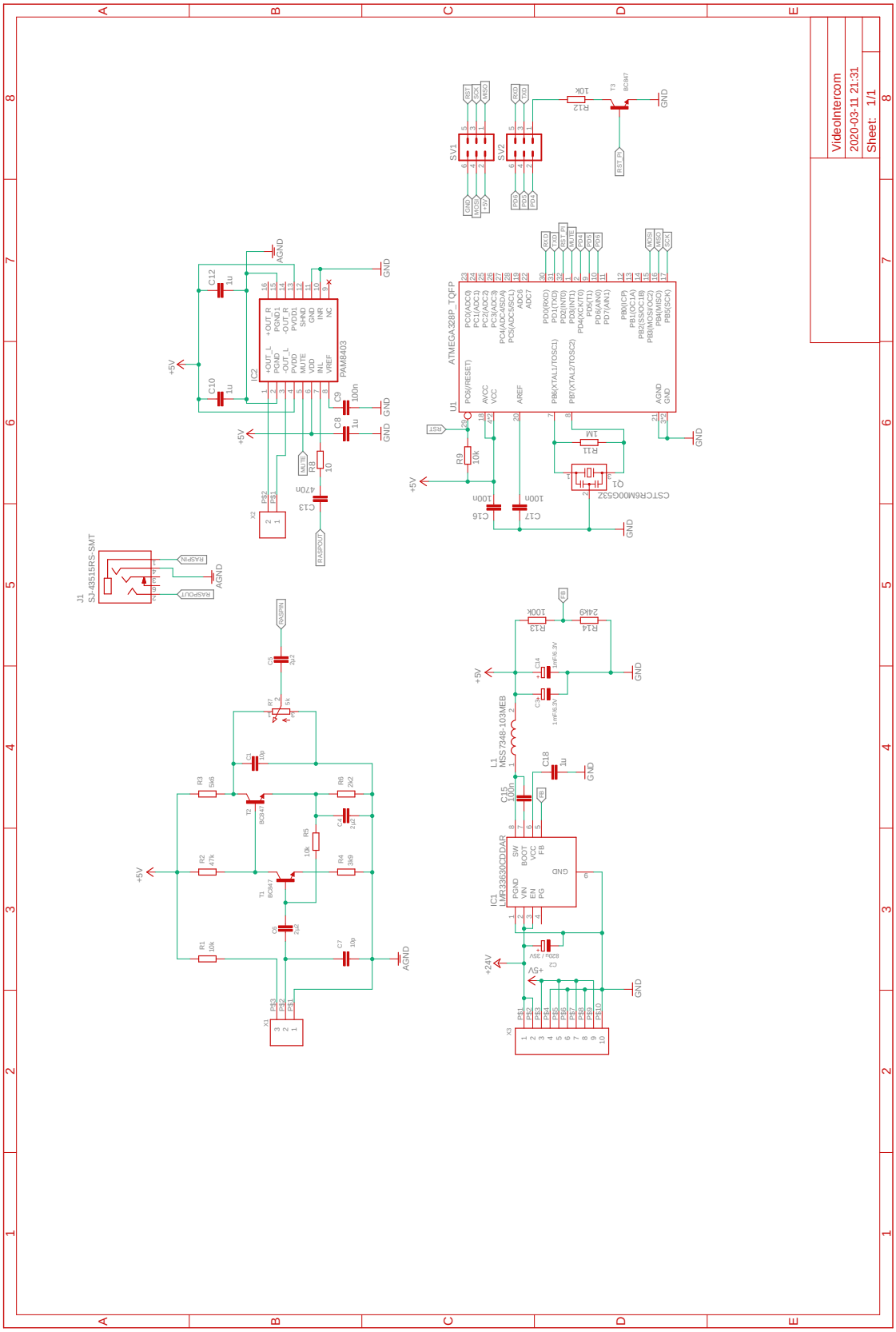
```
56                  {
57                      audioRecord.Read(_audioBuffer, 0, _audioBuffer.Length);
58                      _udp.Send(_micActive ? _audioBuffer : new byte[
   _audioBuffer.Length], _audioBuffer.Length, _target);
59                  }
60                  catch (ThreadAbortException)
61                  {
62                      //before the thread stops
63                      audioRecord.Stop();
64                      audioRecord.Release();
65                      break;
66                  }
67                  catch (Exception e)
68                  {
69                      Console.WriteLine($"AudioRecord: {e.Message}");
70                      break;
71                  }
72              }
73          }
74
75          public void Stop()
76          {
77              _recordingThread.Abort();
78  #if DEBUG
79              Toast.MakeText(Android.App.Application.Context, "Stopped
   Recording", ToastLength.Short).Show();
80  #endif
81          }
82
83          public bool ToggleMic()
84          {
85              return _micActive = !_micActive;
86          }
87      }
88  }
```

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell\MainPage.xaml

```
 1 <?xml version="1.0" encoding="utf-8"?>
 2
 3 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
 4              xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
 5              xmlns:local="clr-namespace:PiBell"
 6              xmlns:piBell="clr-namespace:PiBell;assembly=PiBell"
 7              xmlns:shared="clr-namespace:LibVLCSharp.Forms.Shared;assembly=
   LibVLCSharp.Forms"
 8              x:Class="PiBell.MainPage"
 9              Title="Main Page"
10              BackgroundColor="#ff1b1b1b">
11
12     <ContentPage.Content>
13         <Grid Margin="10" x:Name="MainGrid">
14             <Grid.RowDefinitions>
15                 <RowDefinition Height=".7*" />
16                 <RowDefinition Height="3*" />
17                 <RowDefinition Height="2*" />
18             </Grid.RowDefinitions>
19             <Grid.ColumnDefinitions>
20                 <ColumnDefinition Width="*" />
21             </Grid.ColumnDefinitions>
22
23             <Entry x:Name="EditMrl" Text="{Binding Mrl, Mode=TwoWay}" Grid.
   Row="0" Grid.Column="0" TextColor="White" />
24
25             <shared:VideoView x:Name="VideoView" Grid.Column="0" Grid.Row="1
   " MediaPlayer="{Binding MediaPlayer}"/>
26
27             <Grid x:Name="ButtonGrid" Grid.Row="2" Grid.Column="0">
28                 <Grid.ColumnDefinitions>
29                     <ColumnDefinition Width="*" />
30                     <ColumnDefinition Width="*" />
31                 </Grid.ColumnDefinitions>
32                 <Grid.RowDefinitions>
33                     <RowDefinition Height="*" />
34                     <RowDefinition Height="*" />
35                 </Grid.RowDefinitions>
36
37                 <ImageButton x:Name="BtConnect" Source="call.png"
   BackgroundColor="LimeGreen" Grid.Row="0"
38                              Grid.Column="0" Margin="10"
39                              Clicked="BtConnect_Clicked"/>
40                 <ImageButton x:Name="BtDisconnect" Source="Hangup.png"
   BackgroundColor="Red" Grid.Row="0"
41                              Grid.Column="1" Margin="10"
42                              Clicked="BtDisconnect_Clicked"/>
43
44                 <ImageButton x:Name="BtToggleSpeaker" Source="SpeakerMute.png
   " BackgroundColor="Transparent"
45                              Grid.Row="1" Grid.Column="0" Margin="10"
46                              Clicked="BtToggleSpeaker_Clicked">
47                     <VisualStateManager.VisualStateGroups>
48                         <VisualStateGroup x:Name="SpeakerStates">
49                             <VisualState Name="Mute">
50                                 <VisualState.Setters>
51                                     <Setter Property="Source"
52                                             Value="SpeakerMute.png" />
53                                 </VisualState.Setters>
54                             </VisualState>
```

```
55                              <VisualState Name="Unmute">
56                                  <VisualState.Setters>
57                                      <Setter Property="Source"
58                                              Value="SpeakerUnmute.png" />
59                                  </VisualState.Setters>
60                              </VisualState>
61                          </VisualStateGroup>
62                      </VisualStateManager.VisualStateGroups>
63                  </ImageButton>
64
65                  <ImageButton x:Name="BtToggleMic" Source="MicMute.png"
   BackgroundColor="Transparent"
66                               Grid.Row="1" Grid.Column="1" Margin="10"
67                               Clicked="BtToggleMic_Clicked">
68                      <VisualStateManager.VisualStateGroups>
69                          <VisualStateGroup x:Name="MicStates">
70                              <VisualState Name="Mute">
71                                  <VisualState.Setters>
72                                      <Setter Property="Source"
73                                              Value="MicMute.png" />
74                                  </VisualState.Setters>
75                              </VisualState>
76                              <VisualState Name="Unmute">
77                                  <VisualState.Setters>
78                                      <Setter Property="Source"
79                                              Value="MicUnmute.png" />
80                                  </VisualState.Setters>
81                              </VisualState>
82                          </VisualStateGroup>
83                      </VisualStateManager.VisualStateGroups>
84                  </ImageButton>
85              </Grid>
86          </Grid>
87      </ContentPage.Content>
88 </ContentPage>
```

B.2 Schaltpläne



B.3 Platinen-Design

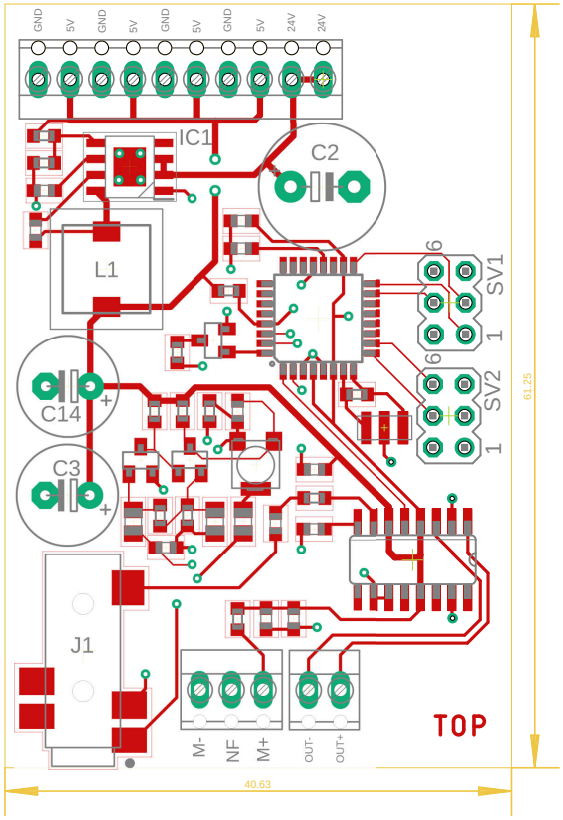


Abbildung B.1: Platine TOP-Layer

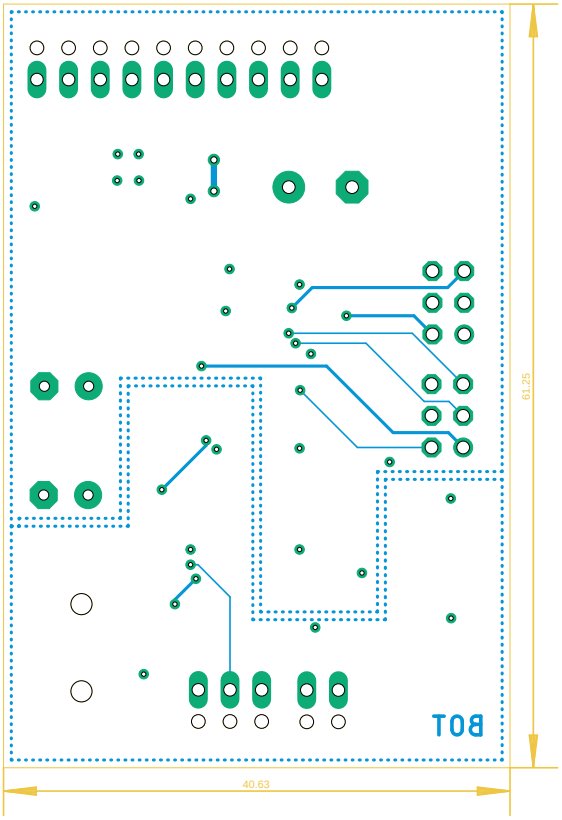


Abbildung B.2: Platine BOT-Layer

B.4 Stückliste

Qty	Value	Package	Parts	Description
1	470nF	C0603	C13	Capacitor
1	820µF/35V	E5-10,5	C2	polarized Capacitor
2	1mF/6.3V	E3,5-8	C3, C14	polarized Capacitor
3	2.2µF	C0805	C4, C5, C6	Capacitor
4	1µF	C0603	C8, C10, C12, C18	Capacitor
4	100nF	C0603	C9, C15, C16, C17	Capacitor
1	LMR33630CDDAR	DDA0008J	IC1	Buck Converter
1	PAM8403	SOP-16	IC2	Dual channel audio amplifier
1	3.5mm/4 conductor	SJ-43515RS-SMT	J1	Audio Jack
1	10µH/2.9A	MSS7348-103MEB	L1	Inductor
1	6Mhz	CSTCR6M00G53Z	Q1	Resonator
1	1MΩ	R0603	R11	Resistor
2	10kΩ	R0603	R1, R5	Resistor
1	100kΩ	R0603	R13	Resistor
1	24.9kΩ	R0603	R14	Resistor
1	47kΩ	R0603	R2	Resistor

Qty	Value	Package	Parts	Description
1	5.6kΩ	R0603	R3	Resistor
1	3.9kΩ	R0603	R4	Resistor
1	2.2kΩ	R0603	R6	Resistor
1	5kΩ	POT_33/64W	R7	Potentiometer
1	10Ω	R0603	R8	Resistor
2	10kΩ	R0603	R9, R12	Resistor
2	2x3pin	MA03-2	SV1, SV2	Pin Header
3	BC847	SOT23	T1, T2, T3	NPN Transistor
1	ATmega328P	TQFP32-08	U1	Microcontroller
1	3pin	RS-CONN-2.54-3P	X1	Screw Terminal
1	2pin	RS-CONN-2.54-2P	X2	Screw Terminal
1	10pin	RS-CONN-2.54-10P	X3	Screw Terminal