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 (HV)
Matthias Mair 5AHET

Betreuer:

DI(FH) Mario Prantl

Innsbruck, am 2020-02-26

Abgabevermerk:

Betreuer:

Datum:

Andreas Grain / Matthias Mair

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 verstanden werden soll.

Andreas Grain / Matthias Mair

Inhaltsverzeichnis

Kurzfassung/Abstract

Die vorliegende Diplomarbeit beschäftigt sich mit der Erweiterung einer Raspberry Pi (RPi)-basierten Videosprechanlage um einer 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 _ und _ an der HTL Anichstraße aus dem Jahr _, welche ein voll funktionsfähiges Videosprechanlagensystem hervorbrachte. Jedoch wurde damals die Hardware-Entwicklung sehr vernachlässigt.

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 erweitert, mit welcher der Fernzugriff auf das System ermöglicht werden soll.

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 _ and _ at the HTL Anichstraße from the year _, which produced a fully functional video intercom system. However, at that time the hardware development was rather neglected.

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.

Andreas Grain / Matthias Mair

Projektergebnis

1 Einleitung

Jedes moderne Wohngebäude ist inzwischen mit einer Gegensprechanlage ausgestattet. Eine solche Anlage werlaubt es dem Wohnungsbesitzer mit jemandem vor der Haustür zu reden, bevor dieser hereingelassen wird. Manche Wohnungen besitzen bereits eine Videosprechanlage, die nicht nur den Ton, sondern zusätzlich ein Video von außen dem Bewohner 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 an der HTL Anichstraße entwickelten _ und _ eine Videogegensprechanlage, welche mehrere Innenstationen und die bidirektionale Video- und Tonübertragung unterstützt. Die Idee ist, dass in einem Wohnungskomplex pro Partei eine Innestation 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 kommuniziernen. Die Stationen wurden mit Hilfe eines RPi realisiert, um die Kosten gering zu halten.

Am Anfang des 5. Schuljahres bat uns unser FI-Professor DI(FH) Mario Prantl an, 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. 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 Gäste empfangen werden können.

Paketdienst Die Paketübergabe erfolgt üblicherweise persönlich. Wenn der Empfänger nicht zuhause ist, nimmt der Paketdienst die Ware wieder mit, und deponiert sie bei einer Paketabholstelle und hinterlässt dem Empfänger lediglich eine Benachrichtigung. Diese Vorgangsweise ist besonders ärgerlich, wenn man den Paketboten 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 Postboten 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 gewünschte bzw. ungewünschte Besucher hat. Über die Smartphone-Applikation ist ein Öffnen des Türschlosses aus mehreren Gründen deaktiviert.

- Wenn der Benutzer nicht zuhause ist, ist eine Türöffnung sowieso irrelevant.
- 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.

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 Multiplatform-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 linuxbasierten Server zuständig. Darüber hinaus organisiert Andreas Grain alle Treffen mit dem Betreuer, achtet auf die Einhaltung des Zeitplans und ist zuständig für das Erstellen des XELATEX -Dokuments

Matthias Mair is verantwortlich für die Hardware-Überarbeitung der Station. Im Zuge dieser sollen die vielen Einzelmodule in einer übersichtlichen Platine vereint werden.

Teil I

Hardware-Erweiterung - Matthias Mair

2 Probleme des Ist-Standes

3 Platine

3.1 Spannungsversorgung

3.2 Mikrofon-Verstärkerschaltung

3.3 Lautsprecher-Verstärkerschaltung

3.4 Mikrocontroller

3.4.1 Programmierung

3.4.2 Watchdog

Teil II

Smartphone-App - Andreas Grain

4 Auswahl Rahmenwerk

4.1 Vergleich

Für die App-Programmierung stehen viele verschiedene Rahmenwerke bereits zur Verfügung. Dieser Abschnitt beschäftigt sich mit dem Vergleich der einzelnen Möglichkeiten, sowie einer endgültigen Auswahl eines der Rahmenwerke zur Verwendung im Diplomprojekt.

Microsoft Xamarin ist ein Rahmenwerk, mit dem man Multiplatform-Applikationen für unter Anderem Android, iOS, UWP und noch viele weitere Platformen erstellen kann. Es basiert auf Microsoft's .NET-Rahmenwerk und dem Mono-Projekt, welches sich als Ziel gesetzt hat, das .NET-Rahmenwerk auf andere Platformen zu portieren. Xamarin bietet mehrere Projekttypen an, darunter Xamarin.Android und Xamarin.iOS für native App-Entwicklung und Xamarin.Forms für großteils platformunabhängige Entwicklung. Eine Xamarin.Forms-Solution besteht daher aus drei Teilen: einem Portable/.NET-Standard-Projekt und je ein natives Xamarin-Projekt für jede Zielplatform. Der Vorteil Xamarin's ist, dass der Großteil des geschriebenen Codes im platformunabhängigen 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.

Apache Cordova ist ein Appentwicklungs-Rahmenwerk welches ursprünglich von der Firma Nitobi entwickelt wurde. Im Jahr 2011 wurde Nitobi vone Adopbe aufgekauft und das Rahmenwerk wurde zu PhoneGap umgetauft. Später wurde eine quelloffene Version unter dem ursprünglichen Namen veröffentlicht. Apps werden mittels gängiger Webtechnologie, wie zum Beispiel JavaScript, CSS und Ähnlichen, entwickelt und realisiert, weshalb das Rahmenwerk alle gängigen Platformen unterstützt. Der Vorteil von Cordova liegt im enormen Support dieser Webtechnologien, jedoch sind Sprachen wie JavaScript eher weniger für Back-End-Programmierung geeignet. Noch dazu kommt, dass an der HTL Anichstraße großteils die Programmierung nur in C und C# gelehrt wird, welshalb die Entwicklung mit JavaScript nicht realistisch ist.

Andere Neben Apache Cordova existieren noch viele weitere Rahmenwerke, die wie Cordova auf Webtechnologie aufbauen, um so die platformunabhängigkeit garantiern zu können. Allerdings bringen diese alle dieselben Probleme wie Cordova mit sich.

4.2 Build-Umgebung

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.

Visual Studio 2019 organisiert den Programmcode in sogenannten Solutions. 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 unterschtützt 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 zwischen 7GB und 10GB. Für eine solche Installation muss im Visual Studio Installer das Mobile development with .NET"-Paktet ausgewählt und installiert werden.

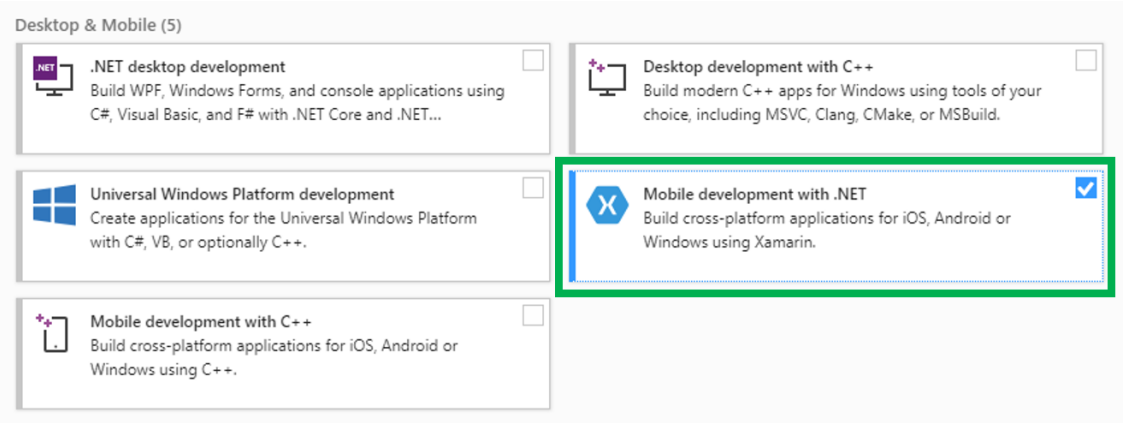

Abbildung 4.1: Auswahl des Xamarin-Rahmenwerks bei der Installation

Auf nähere Details der Installation von Visual Studio wird hier nicht eingegangen, diese können aber auf der Microsoft-Dokumentationsseite [Mic19] nachgeschalgen werden.

4.2.1 Installation Visual Studio

Die minimalen Hardware-, bzw. Software-Anforderungen können der Microsoft-Dokumentation des Visual Studio entnommen werden:

Unterstützte Betriebssysteme	- Windows 10, Version 1703 oder höher (LTSC und S werden nicht unterstützt) - Windows Server 2019 - Windows Server 2016 - Windows 8.1 (mit Update 2919355) - Windows Server 2012 R2 (mit Update 2919355) - Windows 7 SP1 (mit neuesten Windows-Updates)
Hardware	- 1,8-GHz-Prozessor oder schneller; Quad-Core oder besser empfohlen - 2 GB RAM; 8 GB RAM empfohlen - Speicherplatz auf der Festplatte: Mindestens 800 MB, je nach installierten Features bis zu 210 GB des verfügbaren Speicherplatzes. Eine typische Installation erfordert 20–50 GB freien Speicherplatz. - Festplattengeschwindigkeit: Zur Verbesserung der Leistung installieren Sie Windows und Visual Studio auf einem Festkörperlaufwerk (SSD). - Grafikkarte, die eine Auflösung von mindestens 720p (1280 x 720) unterstützt. Visual Studio funktioniert am besten mit einer Auflösung von WXGA (1366 x 768) oder höher.

Tabelle 4.1: Soft- und Hardware-Anforderungen Visual Studio

Auf der oben genannten Website steht der Visual Studio 2019 Installer zum Download bereit. Hier ist bereits die geünschte Version, in diesem Fall die Community-Edition, auszuwählen.

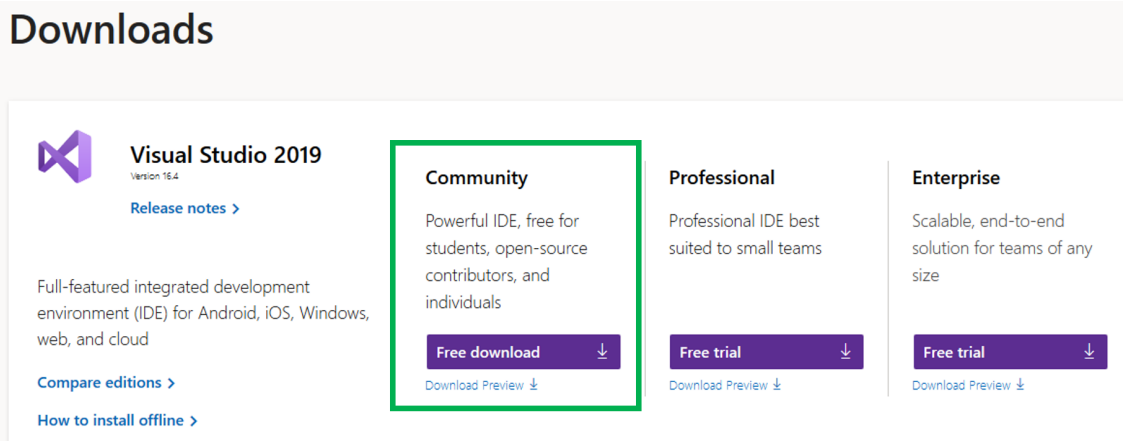

Abbildung 4.2: Auswahl der Visual-Studio-Edition

Nach dem Download und anschließender Installation können die gewünschten Programmteile ausgewählt werden. Für die Entwicklung von Xamarin-Applikationen ist das Mobile development with .NET"-Pakted erforderlich.

Keine weiteren Einstellungen oder Auswahlen sind zwingend erforderlich, allerdings empfielt es sich noch zusätzlich das „.NET desktop development"-Paket zu installieren. Mit diesem kann man neue Technologien des .NET Frameworks in einer einfacheren Konsolenanwendung ausprobieren und kennenlernen.

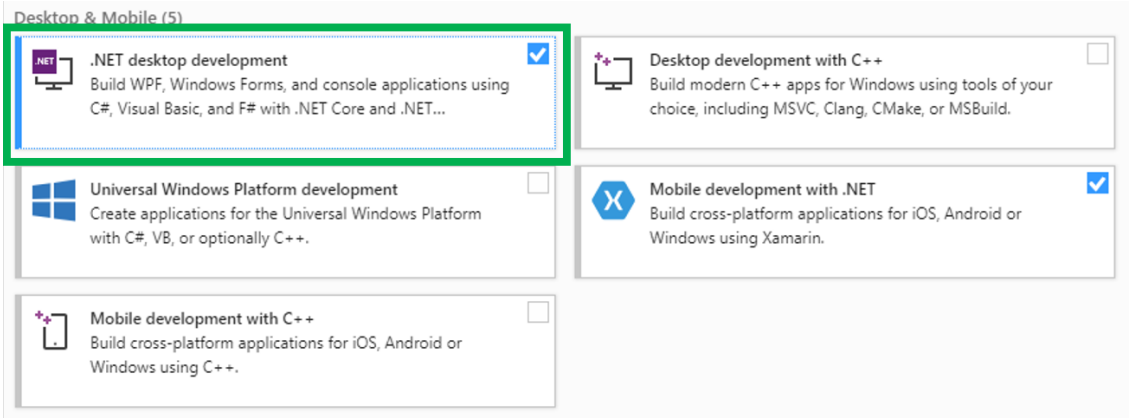

Abbildung 4.3: Zusatzpaket für Desktop-Entwicklung mit .NET

Nach Fertigstellung der Installation von Visual Studio 2019 erscheint dieses am Desktop und im Startmenü als neues Icon.

4.2.2 Installation Android SDK

Das Xamarin-Rahmenwerk benötigt zum Entwickeln von Android-Apps noch zusätzlich mehrere Teile des Android SDK der Firma Google. Mit diesem wird während dem Kompi-

lieren das Programm in ein installierbares Android-Packet, kurz APK, umgewandelt. Damit dies richtig funktioniert, müssen alle Komponenten, auf die im Programm zugegriffen wird, über den Android-SDK-Manager installiert werden.

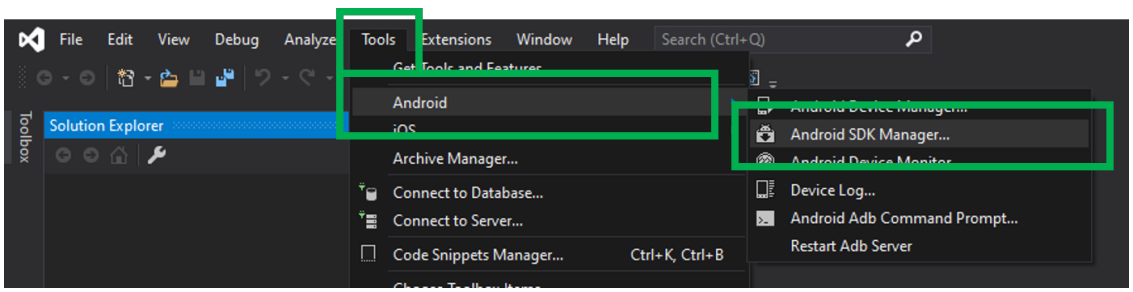

Abbildung 4.4: Android SDK Manager

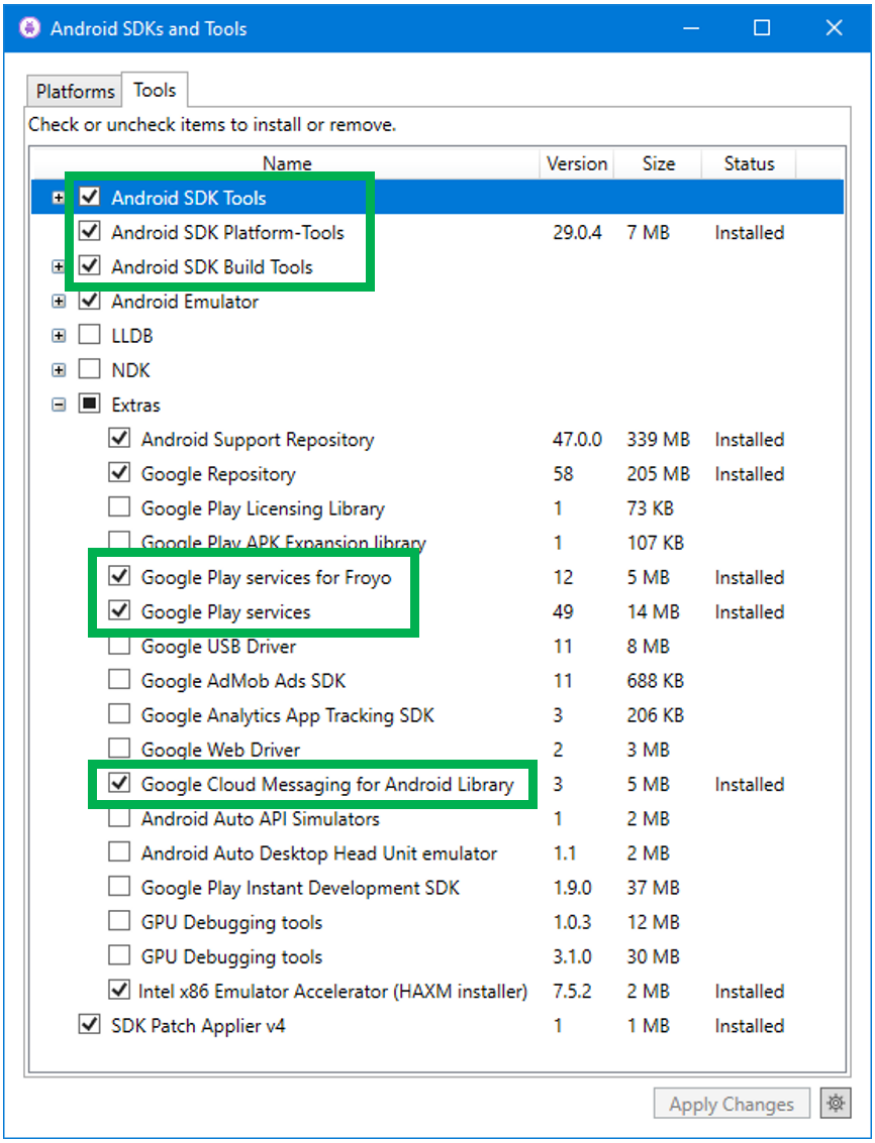

Abbildung 4.5: Notwendige Teile des Android SDK

Besonders wichtig sind hier die Pakete der Google Play Services und die Goole Cloud Messaging Library, welche für das Empfangen von Push-Benachrichtung zwingend erforderlich sind. Diese Benachrigtung wird in einem eigenen Kapitel näher erklärt. Die Android SDK Tools sind für jede Art von Android-Applikation mit Xamarin erforderlich. Bei Bedarf kann über dieses Menü auch der offizielle Android-Emulator installiert werden.

Nach dem Start von Visual Studio muss eine neue „Solution” angelegt werden. Diese beinhaltet alle Programmteile, welche für die Xamarin-Entwicklung benötigt werden.

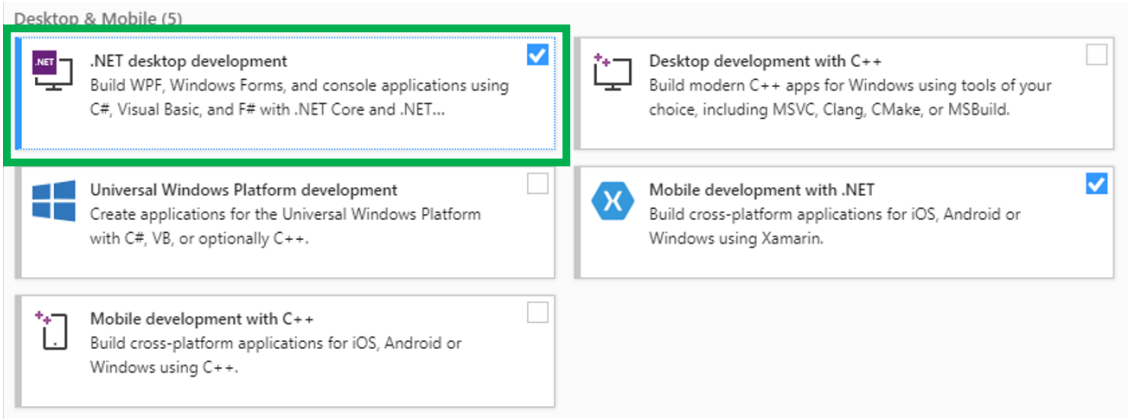

Abbildung 4.6: Anlegen einer neuen Solution

4.3 Gedanken

5 RTSP

6 LibVLC Sharp

7 Push-Benachrichtigung

7.1 Push Notification Service

7.2 Azure Notification Hubs

8 Programm-Dokumentation

8.1 Übersicht

Grundsätzlich besteht das Software-Projekt aus drei Teilen:

PiBell ist das portable Porjekt, auch .NET-Standard-Projekt genannt. Dieses beinhaltet den ganzen platformunabhängigen Code, sowie die Definition der UI-Oberfläche mittels XAML-Dateien.

PiBell.Android beinhaltet allen Code, der platformspezifisch für Android geschrieben wurde. Unter anderem ist hier enthalten die Android-Implementation des Mikrofon-Aufnahme-Services, sowie das Berechtigungs-Management.

PiBell.iOS beinhaltet wie PiBell.Android den platformspezifischen Code für die iOS-Platform. Aufgrund mangelnder Entwicklungswerkzeuge wurde dieser Teil nicht großartig behandelt.

Der immer wieder auftretende Name PiBell ist dabei eine freigeistliche Erfindung. Der Begriff setzt sich zusammen aus RaspberryPi und dem englischen Wort für Glocke oder Klingel (Bell).

Abbildung 8.1: Aufbau der Solution

8.2 Portable Project

8.3 PiBell.Android

8.4 PiBell.iOS

9 Entwicklung

10 Testen

Teil III

Server

11 Grundlagen

11.1 GStreamer

12 Live555 Proxy

A Abkürzungen

RPi Raspberry Pi

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[Mic19] Microsoft. *Install Visual Studio*. Dez. 2019. URL: https://docs.microsoft.com/en-us/visualstudio/install/install-visual-studio?view=vs-2019.

Teil IV

Appendix

B Verwendete Entwicklungswerkzeuge

C Lastenheft

D Zeitplanung

E Kostenübersicht

F Fertigungsdokumentation