



PiSense mit Quadrocopter 2016

Android Applikation

Fernsteuerung

Uni Tübingen

Author:
Christoph Weik

Dienstag 20.08.16

Vorwort

In allen Informatikstudiengängen der Universität Tübingen ist die Teilnahme an einem Softwareprojekt verpflichtend. Das Team-Projekt PiSense mit Quadrocopter, handelte rund um die Anpassung der Low-Level-Treiber, um das Speichern sowie Auslesen von Sensordaten in Datenbank und GUI zu ermöglichen. Der folgenden Unterabschnitt des Projekts hatte zum Ziel, die Entwicklung einer Applikation zur Steuerung der Rotorgeschwindigkeit und Senden vorgegebener Befehlsketten an den Raspberry Pi. Diese wurde von Grund auf neu entworfen und beruht nicht auf Arbeiten vorheriger Semester.

Inhaltsverzeichnis

1	Getting Started (mit Windows 7)	1
1.1	Java 8 installieren	1
1.1.1	Installation des Java Development Kit 8	1
1.1.2	Installation der Java-Dokumentation	2
1.1.3	Installation der Java-Quelltexte	3
1.1.4	Eintragen des Path in die Umgebungsvariable	4
1.2	Android Studio installieren	5
1.2.1	Installation des Hauptprogramms	5
1.2.2	Prüfung der installierten Packages	8
1.2.3	Installation von Intel HAXM	10
1.3	Android Virtual Device und Android Debugging Bridge	12
1.3.1	AVD in Android Studio einrichten	12
1.3.2	Einrichten der Android Debugging Bridge (ADB)	15
1.4	Git	18
1.4.1	Installation von Git	18
1.4.2	Git in Android Studio	19
2	Android Applikation	23
2.1	Einleitung	23
2.1.1	Ressourcen	23
2.1.2	Activity	24
2.1.3	Fragment	25
2.2	Layout	27
2.3	Funktion	29

Abbildungsverzeichnis

1.1	Aktuelle Portierungen des JDK 8.	1
-----	--	---

1.2	Ausgabe der aktuell installierten Java Version in der Kommandozeile.	2
1.3	Kopieren des Docs-Verzeichnis in das JDK-Verzeichnis.	2
1.4	Lesezeichen Dokumentation Java 8 SE.	3
1.5	Entpacken der Archivdatei.	3
1.6	Ausgeben der Versionsnummer des Java Compilers.	4
1.7	Android Studio Setup > Komponenten auswählen.	5
1.8	Android Studio Dialog > Import Settings.	5
1.9	Android Studio Dialog > SDK Components Setup.	6
1.10	Android Studio: Updaten über Quick Start-Menü.	7
1.11	Android Studio: Installierte Packages der SDK Platforms.	8
1.12	Android Studio: Installierte Packages der SDK Tools.	9
1.13	Intel HAXM Setup.	10
1.14	Arbeitsspeicherzuweisung an Intel HAXM.	11
1.15	Intel HAXM: Prüfung auf Systemausführung.	11
1.16	Android Virtual Device Manager über AVD Manager-Icon.	12
1.17	AVD Manager über Menü.	12
1.18	Your Virtual Device Dialog.	13
1.19	AVD Device Selection.	13
1.20	AVD System Image Selection.	14
1.21	Emuliertes Nexus 4 mit Android 6.0.	14
1.22	Android Entwickleroptionen aktivieren.	15
1.23	USB-Debugging und wach bleiben aktivieren.	15
1.24	Angeschlossene Geräte mit Hilfe der ADB anzeigen lassen.	16
1.25	RSA-Schlüssel-Fingerabdruck.	17
1.26	ADB Server Neustart.	17
1.27	Android Gerät erfolgreich als device gelistet.	18
1.28	Help Dokumentation in Atreus.	18
1.29	Version Control Settings.	19
1.30	Git erfolgreich ausgeführt.	19
1.31	Menü Navigation „Create Git Repository“	20
1.32	Starten von Git BASH im Projektverzeichnis	20
1.33	SSH Adresse des Repositories in Atreus.	21
1.34	Hinzufügen eines Remote Repositories in Git BASH.	21
1.35	Ein Projekt mit Android Studio von Git klonen.	21
1.36	Menü Navigation zu Add, Commit und Push.	22
2.1	Beispielhafte Darstellung der String Ressource unserer App.	23
2.2	Callback-Methoden des Activity Lifecycle in Android.	24
2.3	Activity wird vollständig mit Fragment Layout ausgefüllt.	25

2.4	Callback-Methoden des Fragment Lifecycle in Android.	26
2.5	GUI Editor in Android Studio.	27
2.6	For-Schleife zur Bindung von Listenern an View-Objekte.	28
2.7	Innere Arbeiterklasse ClientSend.	29
2.8	Methode udpSend.	29
2.9	Abspeicherung des Seekbarfortschritts in der OnProgressChanged-Methode. .	30
2.10	Auslesen der SharedPreferences XML und Prüfen auf Darstellungsmodus. . .	31

1 Getting Started (mit Windows 7)

1.1 Java 8 installieren

1.1.1 Installation des Java Development Kit 8

Die aktuellste Version des JDK findet man auf der Herstellerwebsite unter:

- <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Java SE Development Kit 8u101		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input checked="" type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.77 MB	jdk-8u101-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.72 MB	jdk-8u101-linux-arm64-vfp-hflt.tar.gz
Linux x86	160.28 MB	jdk-8u101-linux-i586.rpm
Linux x86	174.96 MB	jdk-8u101-linux-i586.tar.gz
Linux x64	158.27 MB	jdk-8u101-linux-x64.rpm
Linux x64	172.95 MB	jdk-8u101-linux-x64.tar.gz
Mac OS X	227.36 MB	jdk-8u101-macosx-x64.dmg
Solaris SPARC 64-bit	139.66 MB	jdk-8u101-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	98.96 MB	jdk-8u101-solaris-sparcv9.tar.gz
Solaris x64	140.33 MB	jdk-8u101-solaris-x64.tar.Z
Solaris x64	96.78 MB	jdk-8u101-solaris-x64.tar.gz
Windows x86	188.32 MB	jdk-8u101-windows-i586.exe
Windows x64	193.68 MB	jdk-8u101-windows-x64.exe

Abbildung 1.1: Aktuelle Portierungen des JDK 8.

Die benötigte Java-Installationsdatei heißt im Falle von Microsoft Windows 7 (64 Bit) `jdk-8u101-windows-x64.exe`.

Vor der Installation ist es jedoch sinnvoll zu Prüfen, ob bereits die aktuellste Version vorliegt. Die geschieht in Windows 7 folgendermaßen:

1. Öffne zuerst die Kommandozeile (Windowstaste + R → im Eingabefeld **cmd** eingeben).
2. In der Kommandozeile **java -version** eingeben.



Abbildung 1.2: Ausgabe der aktuell installierten Java Version in der Kommandozeile.

In diesem Beispiel ist die Java Version 1.7.0_45 bereits veraltet und eine neuere Version sollte installiert werden.

1.1.2 Installation der Java-Dokumentation

Die folgenden Links führen zur Online Dokumentation von Java 8 SE und der Download-Seite:

- <http://docs.oracle.com/javase/8/docs/index.html>
- <http://www.oracle.com/technetwork/java/javase/documentation/jdk8-doc-downloads-2144771.html>

Für uns ist nur die obere mit der Bezeichnung „Java SE Development Kit 8u101 Documentation“ relevant. Diese entpacken wir und kopieren sie anschließend in das JDK-Verzeichnis.

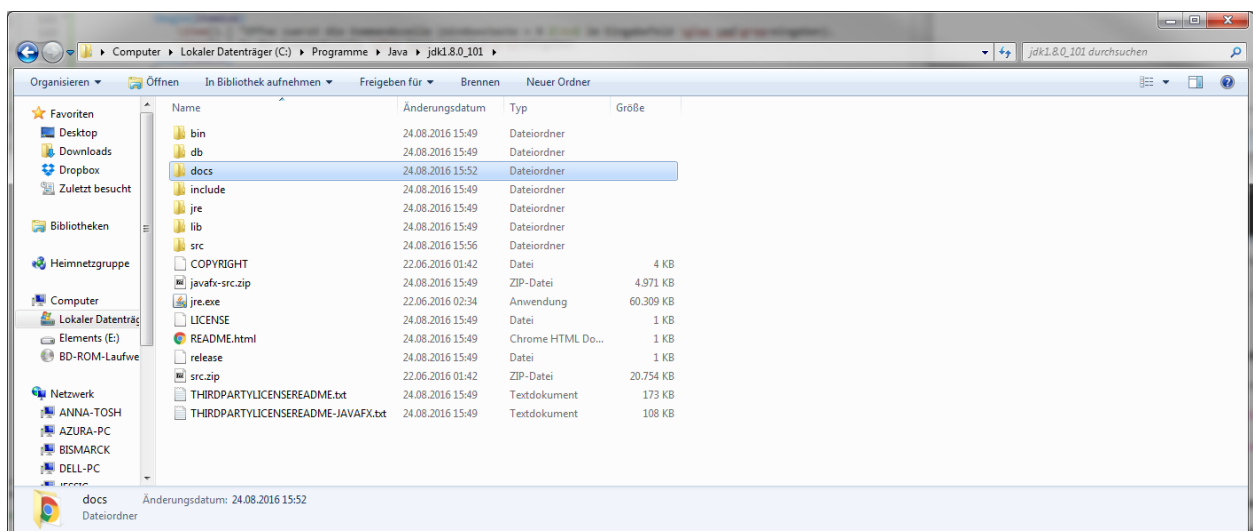


Abbildung 1.3: Kopieren des Docs-Verzeichnis in das JDK-Verzeichnis.

Die Java-Dokumentation wird über einen Browser betrachtet. Es bietet sich daher an ein Lesezeichen anzulegen, welches auf das Hauptdokument verweist. Wie bspw.:

file:///localhost/C:/Program%20Files/Java/jdk1.8.0_101/docs/index.html

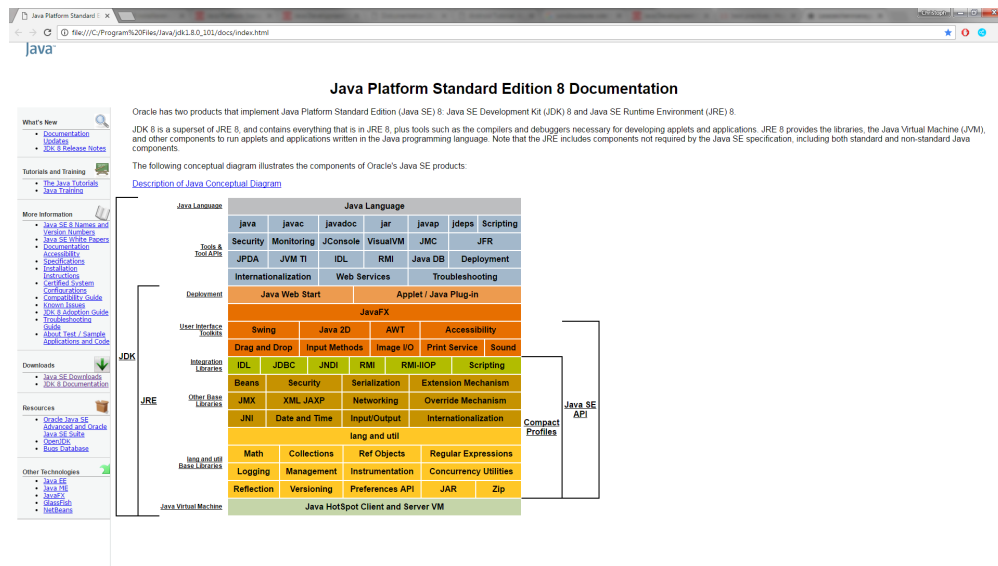


Abbildung 1.4: Lesezeichen Dokumentation Java 8 SE.

1.1.3 Installation der Java-Quelltexte

Die Quelltexte befinden sich bereits im Installationsverzeichnis des JDK 8 und müssen nur noch entpackt werden.

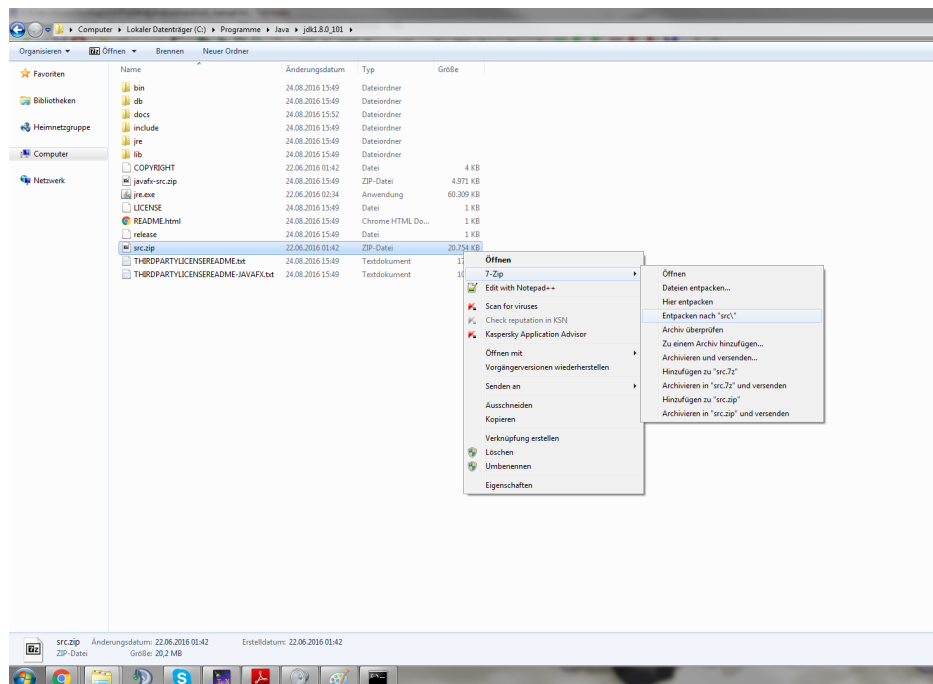


Abbildung 1.5: Entpacken der Archivdatei.

1.1.4 Eintragen des Path in die Umgebungsvariable

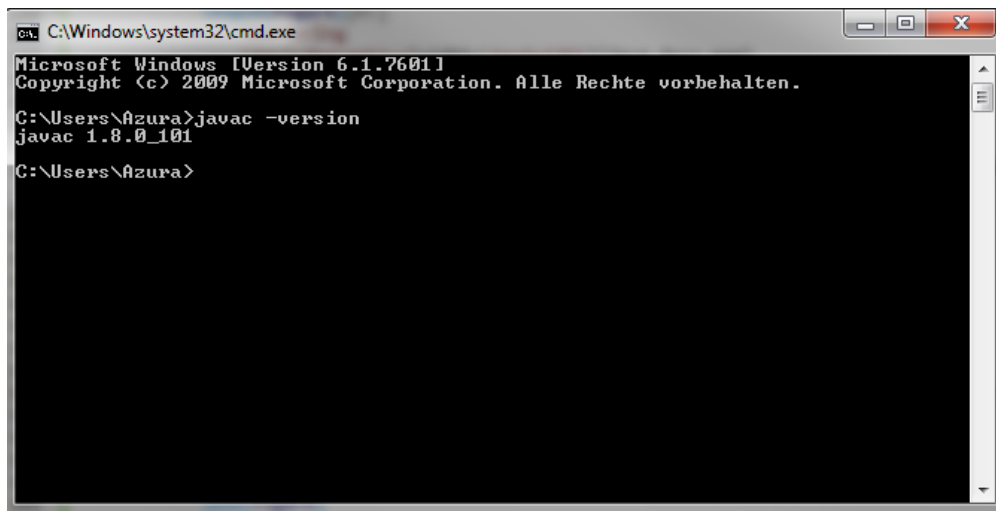
Unter Windows 7 kann die Systemvariable Path folgendermaßen geändert werden:

1. Windowstaste + Pause, dann erweiterte Systemeinstellungen.
2. Auf dem Reiter Erweitert wählen wir Umgebungsvariablen. . .
3. Füge den Pfad des bin-Ordners der JDK-Installation in die Systemvariable Path ein.

Die Path Variable könnte dann beispielsweise so aussehen:

`C:\WINDOWS\system32;C:\WINDOWS;C:\ProgramFiles\Java\jdk1.8.0_101\bin`

Nun sollte es auch möglich sein durch den Befehl **javac -version** die Version des Java Compilers auszugeben.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Azura>javac -version
javac 1.8.0_101

C:\Users\Azura>
```

Abbildung 1.6: Ausgeben der Versionsnummer des Java Compilers.

1.2 Android Studio installieren

1.2.1 Installation des Hauptprogramms

Die Installationsdatei von Android Studio findet ihr auf der Android Entwickler Webseite. Über die Reiter Develop > Android Studio gelangt ihr zum Download-Bereich.

- <http://developer.android.com/>

Nach dem Starten des Wizards werdet ihr kurze Zeit später gebeten die gewünschten Komponenten auszusuchen. Hierbei wählen wir alles außer das Android Virtual Device, da wir später unser eigenes AVD in Android Studio anlegen werden.

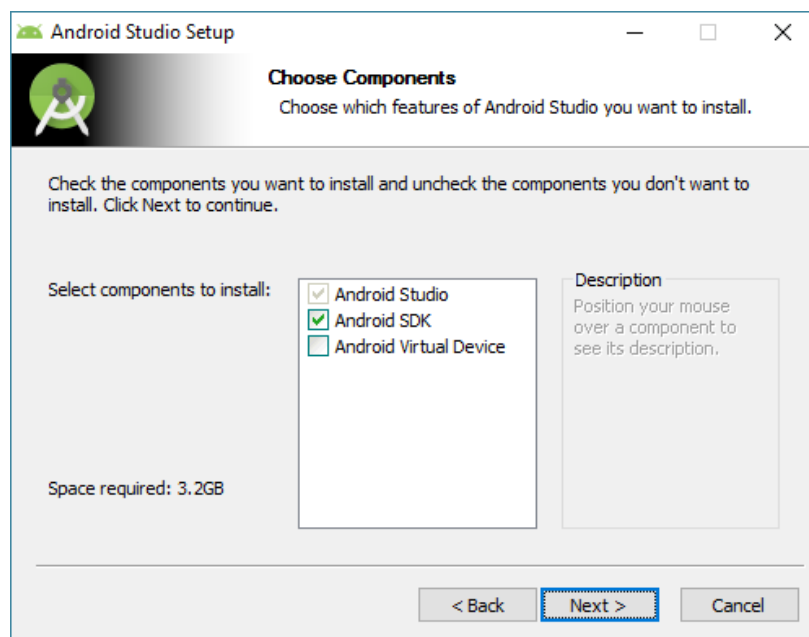


Abbildung 1.7: Android Studio Setup > Komponenten auswählen.

Nach der Beendigung des Wizards erscheint noch der Dialog „Complete Installation“, den wir noch durchlaufen müssen. Da wir von einer erstmaligen Installation von Android Studio ausgehen, wählen wir die untere Option „I do not have a previous version of Studio or I do not want to import my settings“.

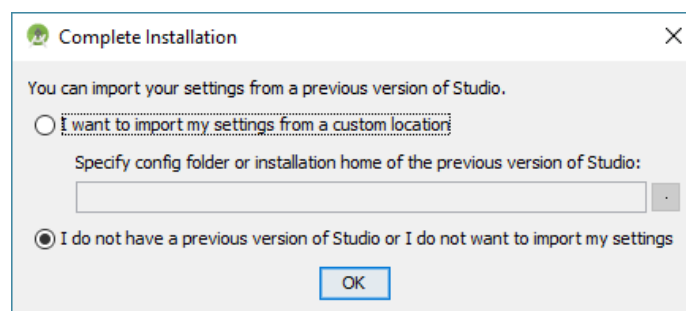


Abbildung 1.8: Android Studio Dialog > Import Settings.

Am SDK Components Setup Dialog angelangt, setzen wir Häkchen bei Android SDK, Android SDK Platform, API 23: Android 6.0 (Marshmallow). Die fehlenden Komponenten werden wir später manuell einrichten.

Wichtig! Bei Android SDK Location den selben Pfad angeben, wie auch schon bei der Android SDK Installation Location.

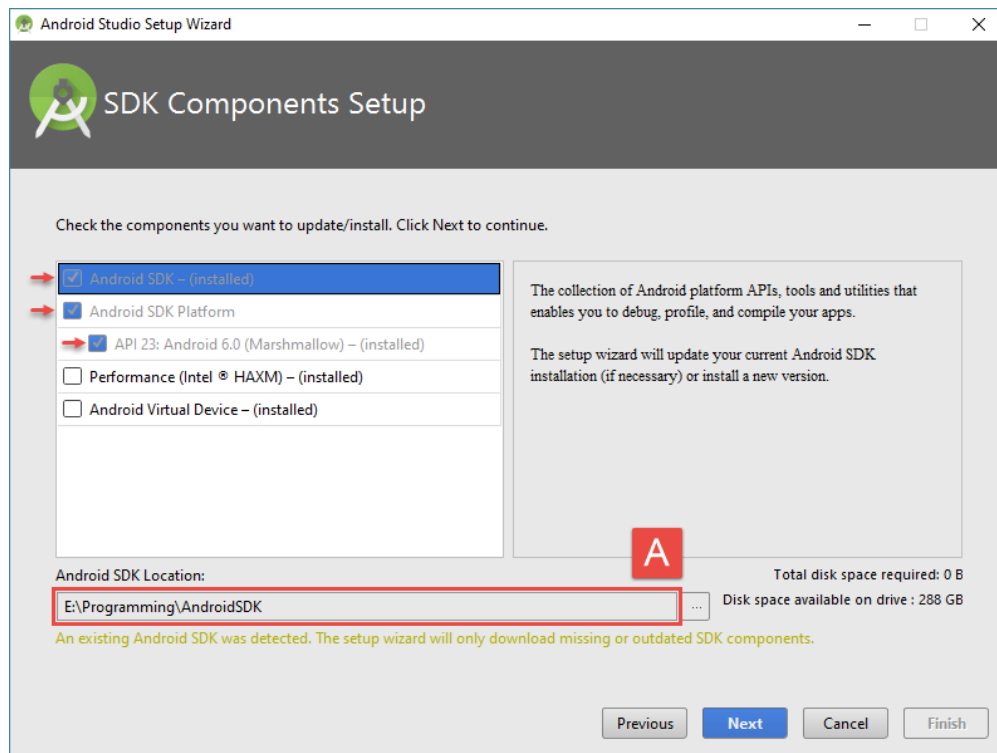


Abbildung 1.9: Android Studio Dialog > SDK Components Setup.

Nach Abschluss der Installation wählen wir im Willkommensbildschirm aus dem Quick Start-Menü „Configure“ und klicken auf den vorletzten Menüeintrag Check for Updates. Ist das Updaten abgeschlossen steht dem Entwickeln einer App nicht mehr viel im Weg.

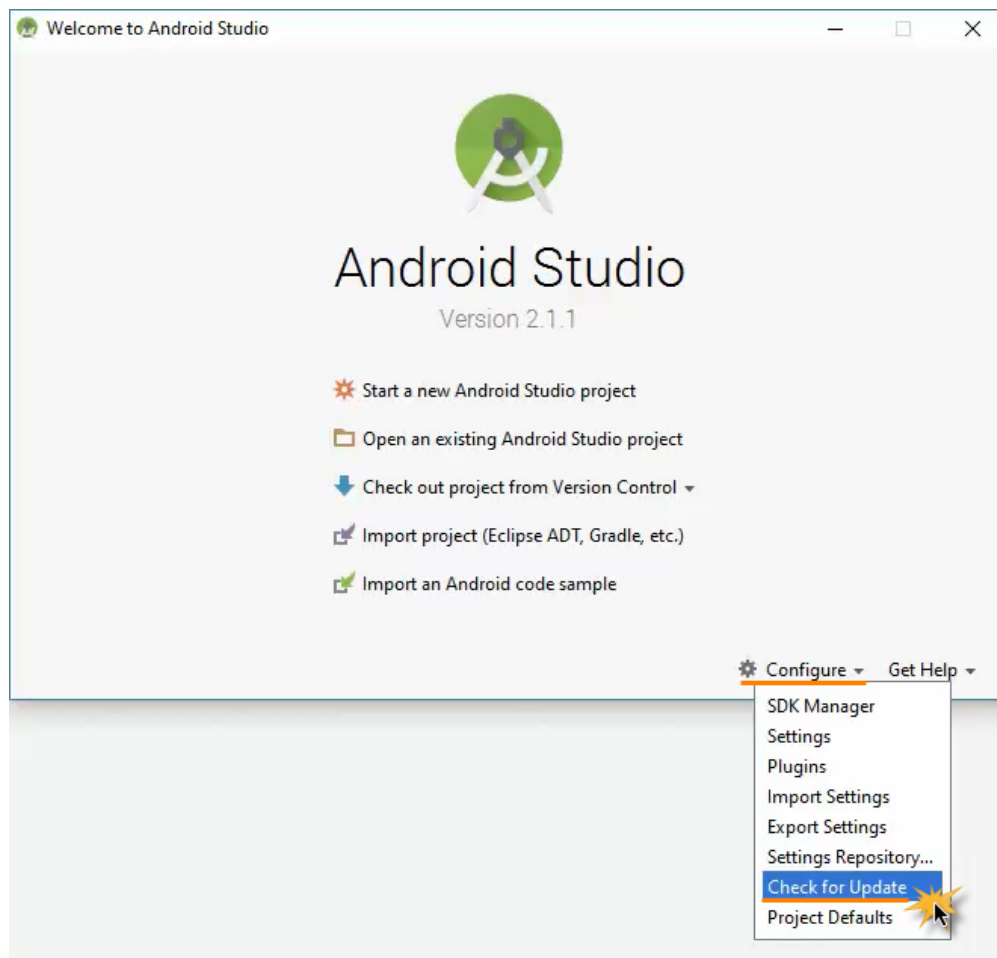


Abbildung 1.10: Android Studio: Updaten über Quick Start-Menü.

1.2.2 Prüfung der installierten Packages

In der Bildserie folgenden werden alle Packages aufgeführt die zur Entwicklung der App verwendet wurden. Es wird empfohlen diese zu installieren, um so maximale Kompatibilität zu gewährleisten. Ihr öffnet den SDK Manager über das Quick Start-Menü. Die jeweiligen Packages werden mit Hilfe des Standalone SDK Managers installiert (siehe Abb. 1.11)

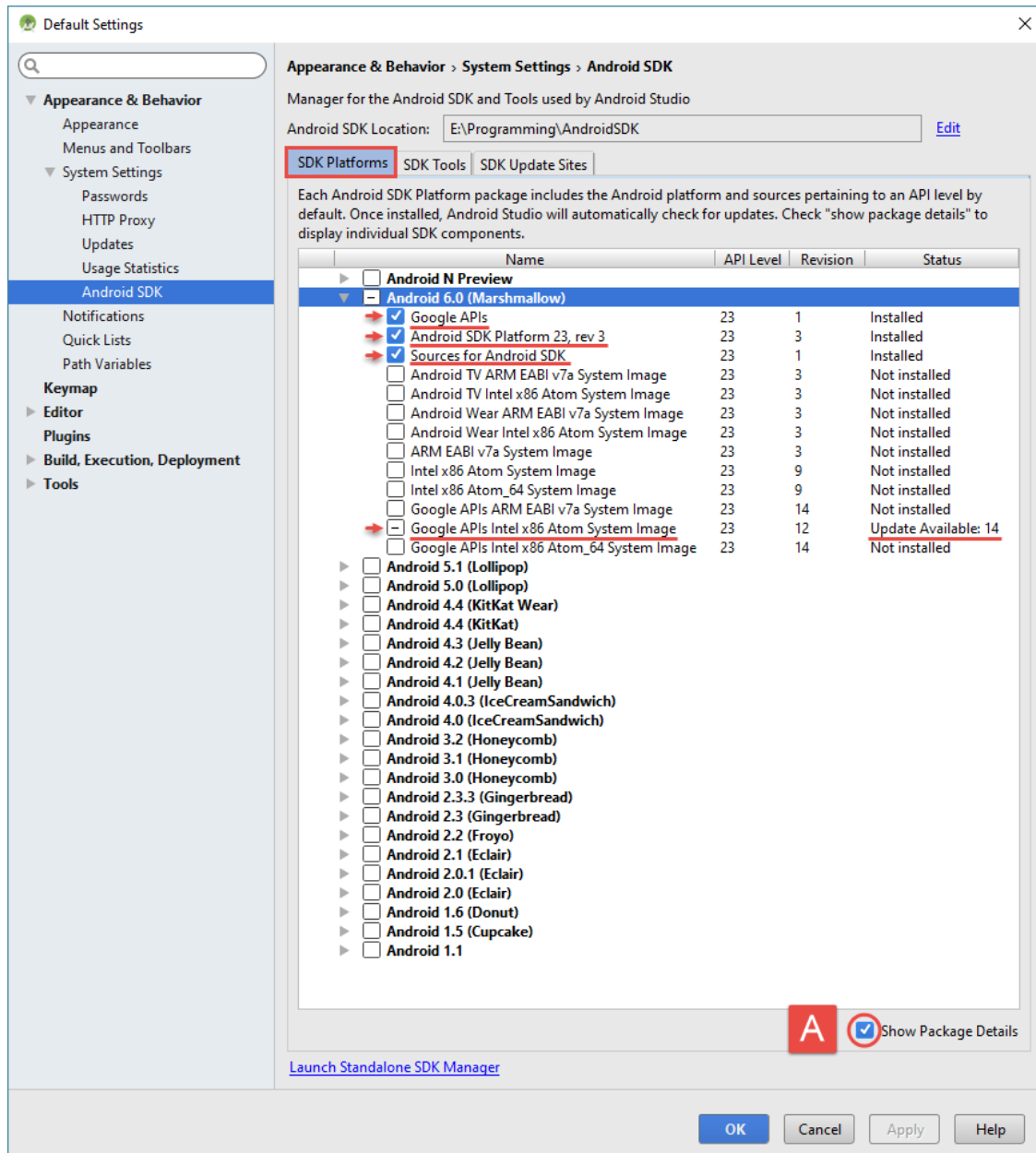


Abbildung 1.11: Android Studio: Installierte Packages der SDK Platforms.

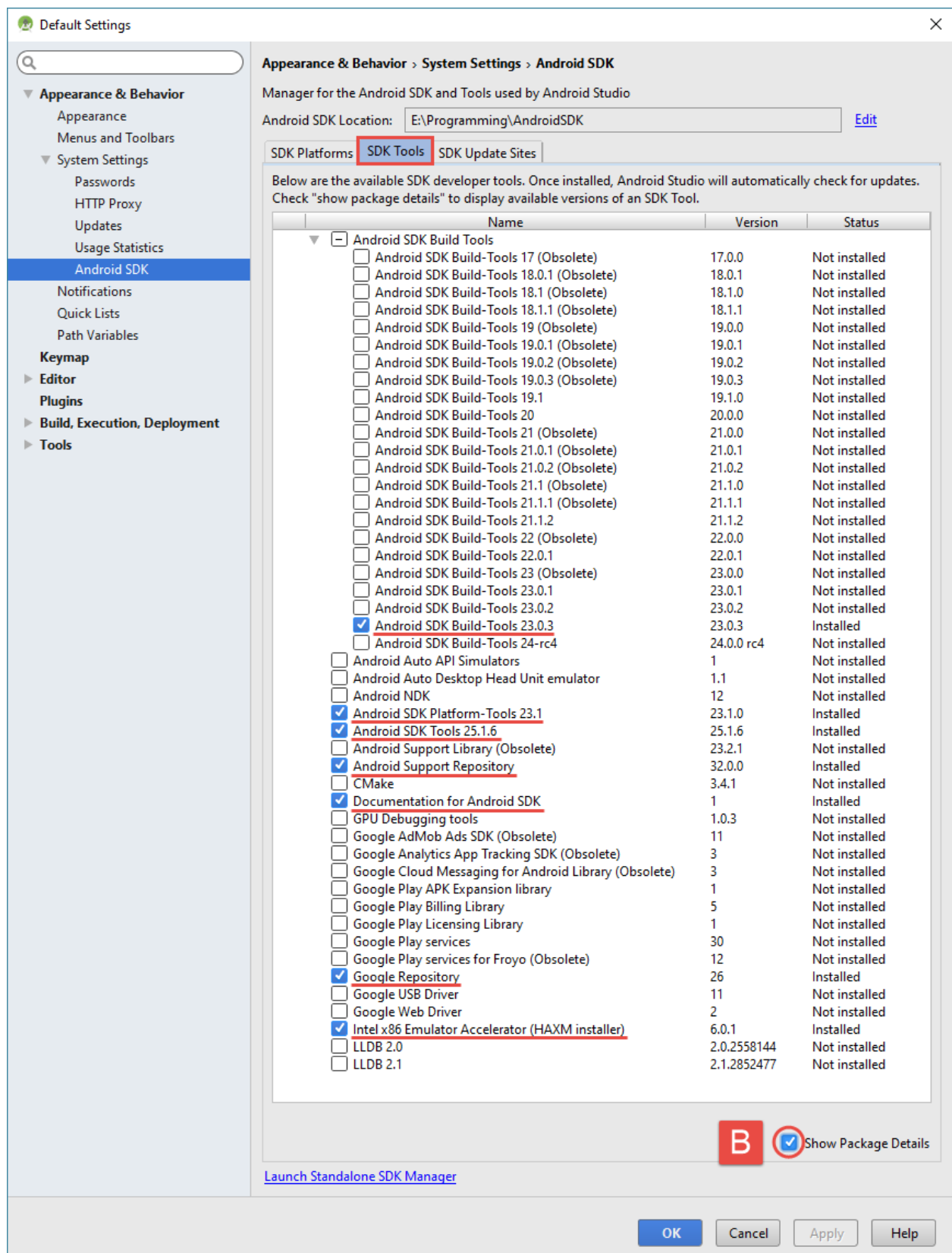


Abbildung 1.12: Android Studio: Installierte Packages der SDK Tools.

1.2.3 Installation von Intel HAXM

Die Installationsdateien des INTEL HAXM (Hardware Accelerated Execution Manager) wurden bei der Installation von Android Studio bereits in das Verzeichnis des Android SDKs kopiert. Das Verzeichnis befindet sich unter folgendem Pfad:

...AndroidSDK\extras\intel\Hardware_Accelerated_Execution_Manager

Nachdem der HAXM-Installer gestartet wurde, öffnet sich der Willkommensbildschirm des Setup-Dialogs. Falls euer System die Systemvoraussetzungen nicht erfüllt, wird die Installation automatisch abgebrochen.



Abbildung 1.13: Intel HAXM Setup.

Im nächsten Schritt wird nun nach einer Obergrenze für den Arbeitsspeicher gefragt, die HAXM verwenden darf. Es kann die Hälfte des auf dem PC verfügbaren Arbeitsspeichers angegeben werden.

Achtung! Dieser Wert sollte nicht zu hoch angesetzt werden, ansonsten könnten andere Programme langsamer ausgeführt werden.



Abbildung 1.14: Arbeitsspeicherzuweisung an Intel HAXM.

Nach abgeschlossener Installation gilt es noch zu Prüfen, ob Intel HAXM ordnungsgemäß auf dem System ausgeführt wird. Hierfür starten wir die Kommandozeile als Administrator und verwenden folgenden Befehl: **sc query intelhaxm**.

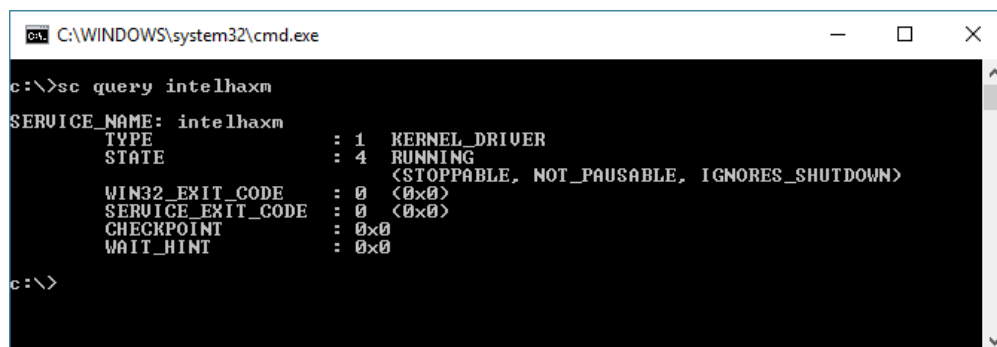


Abbildung 1.15: Intel HAXM: Prüfung auf Systemausführung.

Zeigt die Konsole **STATE: 4 running** oder ähnliches an, wird Intel HAXM korrekt auf dem Rechner ausgeführt.

1.3 Android Virtual Device und Android Debugging Bridge

1.3.1 AVD in Android Studio einrichten

Zur Ausführung einer App auf dem PC benötigen wir einen Android Emulator. Android Studio stellt uns für diesen Zweck den Android Virtual Device Manager zur Verfügung. Die beiden Möglichkeiten auf diesen zuzugreifen werden in den Abbildungen 1.16 und 1.17 aufgeführt.

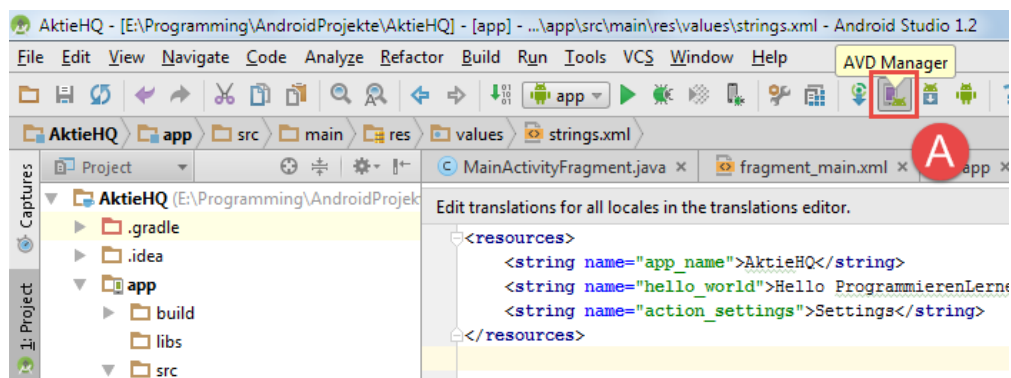


Abbildung 1.16: Android Virtual Device Manager über AVD Manager-Icon.

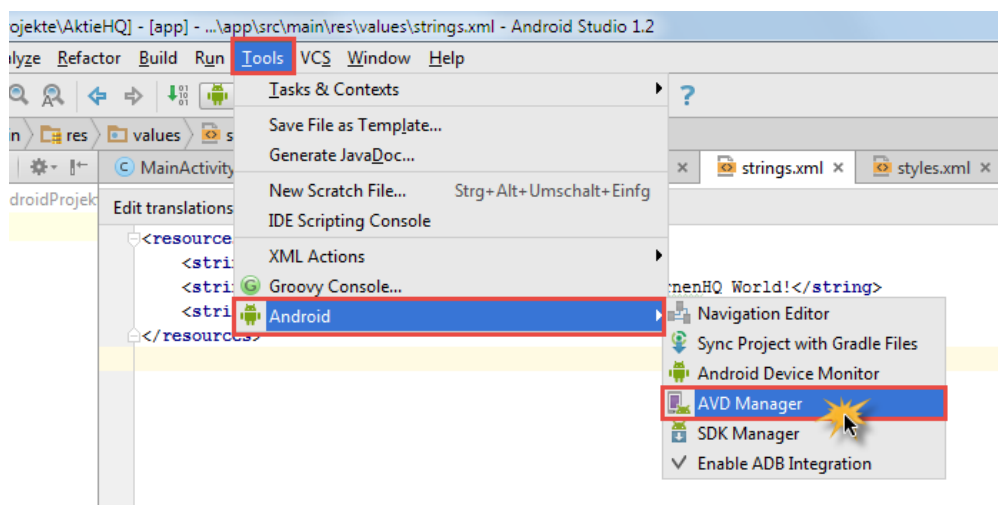


Abbildung 1.17: AVD Manager über Menü.

Nachdem der AVD Manager das erste Mal gestartet wurde, öffnet sich der Your Virtual Devices-Dialog des Managers. An dieser Stelle besteht die Möglichkeit ein Android Virtual Device einzurichten.

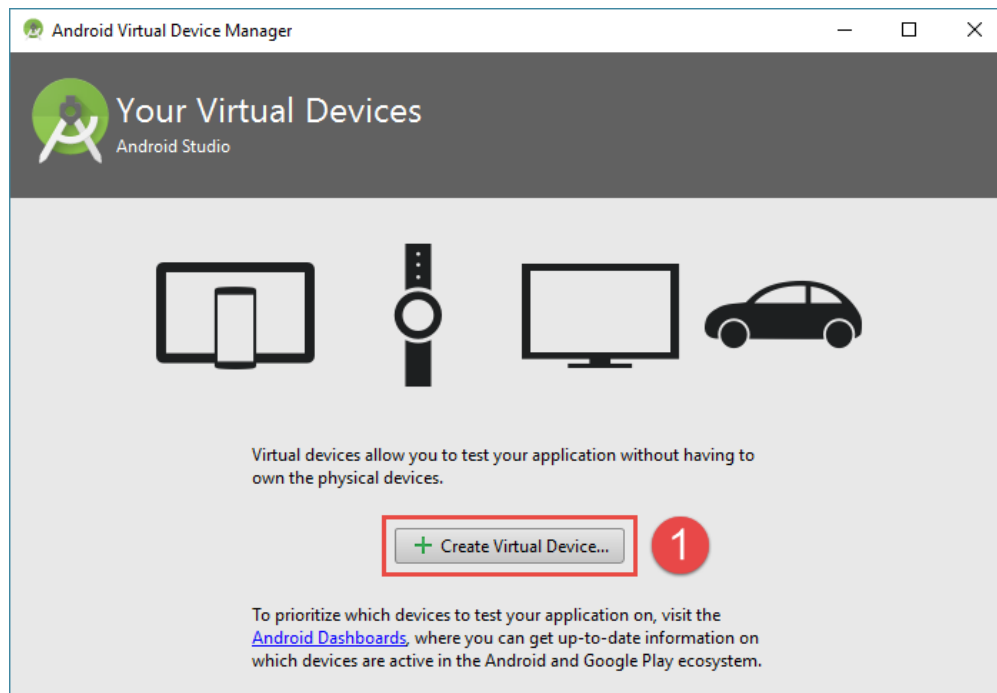


Abbildung 1.18: Your Virtual Device Dialog.

In den folgenden Bildschirmen lassen sich zu emulierendes Gerät, System Image und Bildschirmorientierung bestimmen (Siehe Abb. 1.19 und 1.20). Weitere Images können nach belieben zusätzlich heruntergeladen werden.

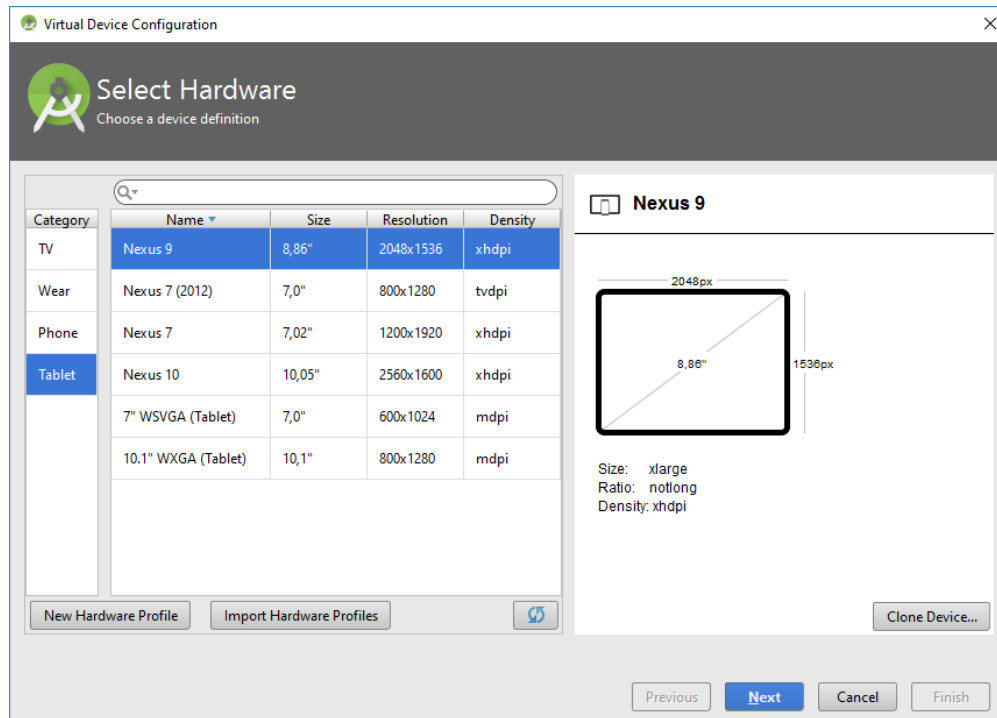


Abbildung 1.19: AVD Device Selection.

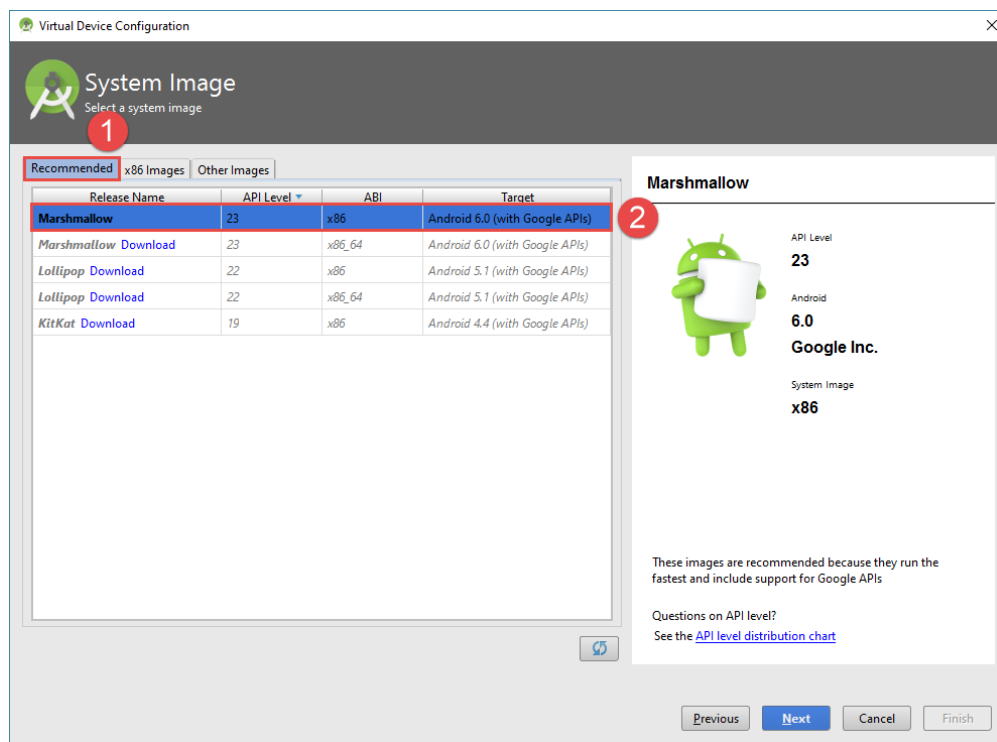


Abbildung 1.20: AVD System Image Selection.

Anschließend lässt sich das emulierte Gerät entweder über den AVD Manager oder das Run-Icon starten. Dieser Vorgang benötigt unter Umständen einige Minuten.

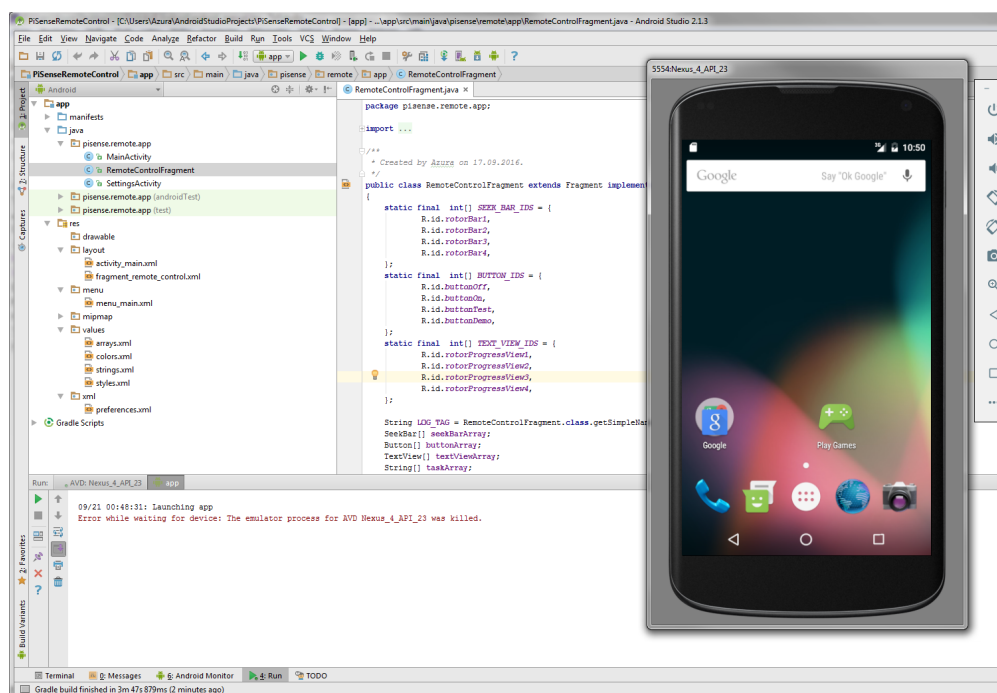


Abbildung 1.21: Emuliertes Nexus 4 mit Android 6.0.

1.3.2 Einrichten der Android Debugging Bridge (ADB)

Das Aufbauen einer Android Debugging Bridge, benötigt zunächst das Aktivieren der Entwicklereinstellungen auf dem Smartphone (Siehe Abb. 1.22). Diese sind auf Geräten mit Android 4.2 oder neuer standardmäßig versteckt.



Abbildung 1.22: Android Entwickleroptionen aktivieren.

Nachdem die Android Entwickleroptionen aktiviert wurden. Müssen zwingend bei „wach bleiben“ und „USB-Debugging“ Häkchen gesetzt werden (Siehe Abb. 1.23). USB-Debugging erlaubt dem angeschlossenen Rechner die Kommunikation mit dem Android Gerät.

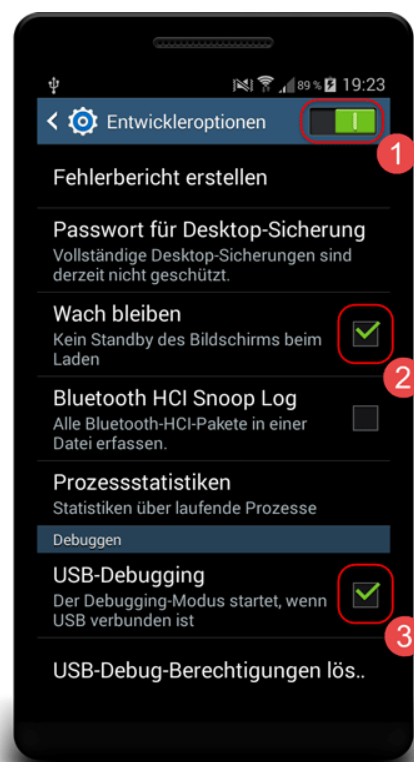


Abbildung 1.23: USB-Debugging und wach bleiben aktivieren.

Damit eine Verbindung des PCs mit dem Android Gerät möglich ist, muss der passende USB Treiber für das Smartphone oder Tablet auf unserem System installiert werden:

- Android Developer Phones, wie Nexus One oder Nexus S, benötigen die Google USB Treiber: <http://developer.android.com/sdk/win-usb.html>
- Einer Liste anderer Hersteller mit Treiber URL findet man hier: <http://developer.android.com/tools/extras/oem-usb.html#Drivers>

Das Ansteuern des Android Gerätes erfolgt über die Konsole im Verzeichnis `..\AndroidSDK\platform-tools` oder das Terminal in Android Studio. Mit dem Befehl **adb devices** lassen sich angeschlossene Geräte anzeigen.

```

C:\Windows\system32\cmd.exe

E:\Programming\AndroidSDK\platform-tools>dir
Datenträger in Laufwerk E: ist Data HD
Volumeseriennummer: 74BC-A172

Verzeichnis von E:\Programming\AndroidSDK\platform-tools 1
24.04.2015  22:53    <DIR>          .
24.04.2015  22:53    <DIR>          ..
15.02.2015  01:12    1.011.200    adb.exe        2
15.02.2015  01:12    96.256    adbwinapi.dll
15.02.2015  01:12    60.928    adbwinusbapi.dll
15.02.2015  01:12    <DIR>          api
15.02.2015  01:12    73.728    dmtracedump.exe
15.02.2015  01:12    338.944    etcdtool.exe
15.02.2015  01:12    196.608    fastboot.exe
15.02.2015  01:12    43.008    hprof-conv.exe
15.02.2015  01:12    816.004    NOTICE.txt
15.02.2015  01:12    39    source.properties
15.02.2015  01:12    700.928    sqlite3.exe
15.02.2015  01:12    <DIR>          systrace
10 Datei(en),    3.337.643 Bytes
4 Verzeichnis(se), 697.724.141.568 Bytes frei

E:\Programming\AndroidSDK\platform-tools>adb.exe devices
List of devices attached
0b077249    unauthorized 3
E:\Programming\AndroidSDK\platform-tools> 4
  
```

Abbildung 1.24: Angeschlossene Geräte mit Hilfe der ADB anzeigen lassen.

Das Gerät ist zu diesem Zeitpunkt noch nicht autorisiert und eine Verbindung unmöglich. Hierbei handelt es sich um einen Schutzmechanismus aller Android Geräte ab Android 4.2.2, um vor unberechtigten Zugriff zu schützen. Die Autorisierung erfolgt über das Android Gerät, in einem Dialog bittet es um einen RSA-Handshake.

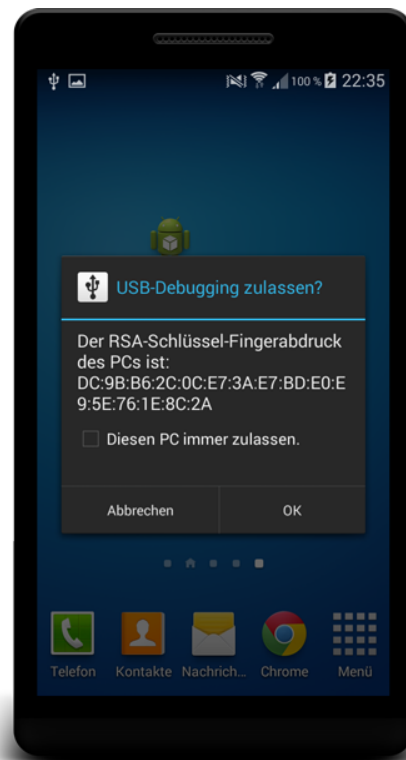


Abbildung 1.25: RSA-Schlüssel-Fingerabdruck.

Um den oben beschriebenen USB-Debugging Dialog anzeigen zu lassen und eine Verbindung zum Android Gerät anzufordern, verwenden wir folgende ADB-Befehle in der Konsole (die Dateieindung .exe ist nicht zwingend erforderlich):

- Beenden des ADB Servers mit: **adb.exe kill-server**
- Starten des ADB Servers mit: **adb.exe start-server**

Achtung! Sicher stellen, dass der Bildschirm des Android Gerätes entriegelt ist!

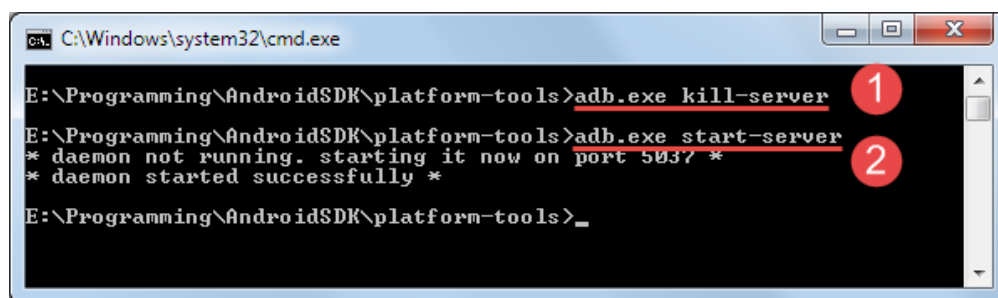


Abbildung 1.26: ADB Server Neustart.

Nach erfolgreicher Ausführung, sollte der Befehl **adb devices** das Android Gerät als device führen. Eine Installation der App auf dem Zielgerät, ist nun mit Hilfe des Run-Icons in Android Studio möglich.

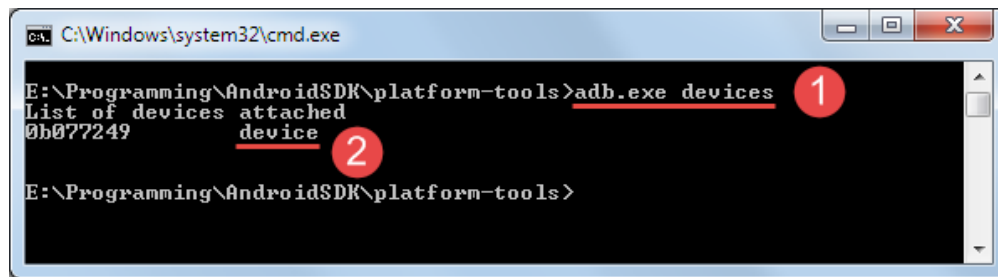


Abbildung 1.27: Android Gerät erfolgreich als device gelistet.

1.4 Git

1.4.1 Installation von Git

Die Verwendung von Git auf Windows 7 erfordert zunächst die Installation einer Git Shell, wie beispielsweise Git BASH.

- <https://git-for-windows.github.io/>

Mit Hilfe der Help Dokumentation im Atreus lassen sich schnell Probleme, wie die Initialisierung des global user.names, der global user.email und das Generieren eines SSH-Keys bewerkstelligen (Siehe Abb 1.28).

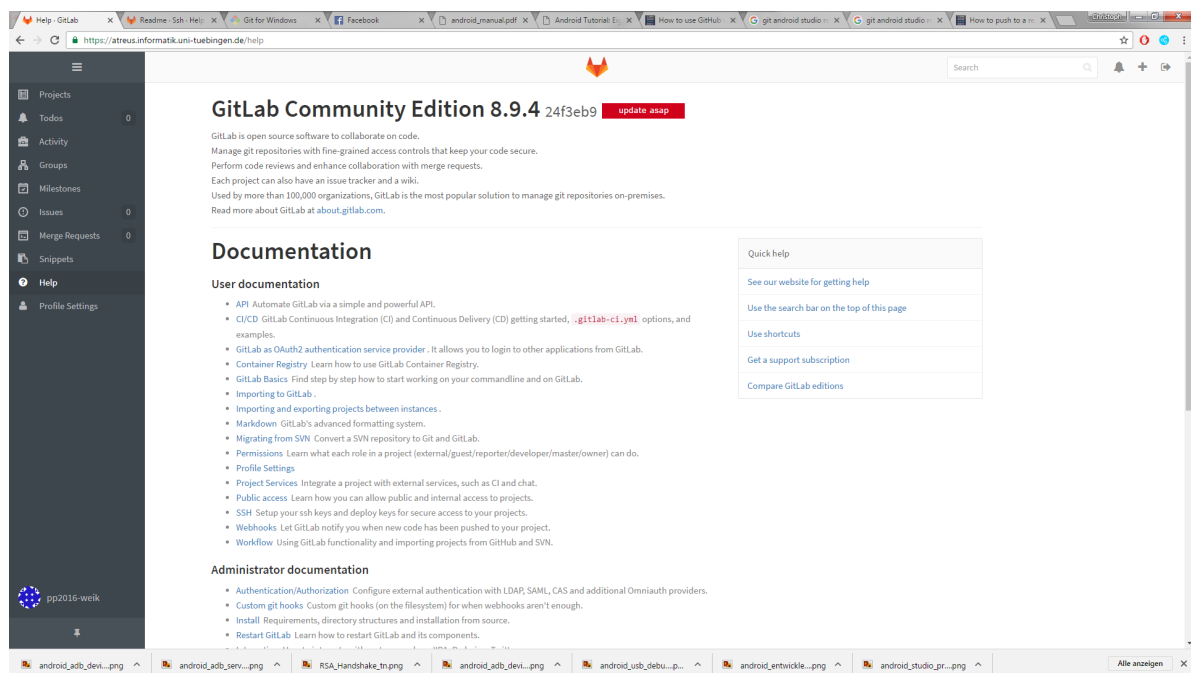


Abbildung 1.28: Help Dokumentation in Atreus.

1.4.2 Git in Android Studio

Im Folgenden handelt es sich um eine Step-by-Step Anleitung zur Einrichtung von Git als Version Control in Android Studio. Zu Beginn navigieren wir zu den Git Version Control Settings über File > Settings > Version Control > Git (Siehe Abb. 1.29). In Windows 7 ist default Pfad der git.exe `C:\Program Files\Git\cmd\git.exe` diesen tragen wir in „Path to executable“ ein. Ein erfolgreicher Test sollte eine Rückmeldung wie in Abbildung 1.30 geben.

Wichtig! SSH-executable auf Built-in lassen.

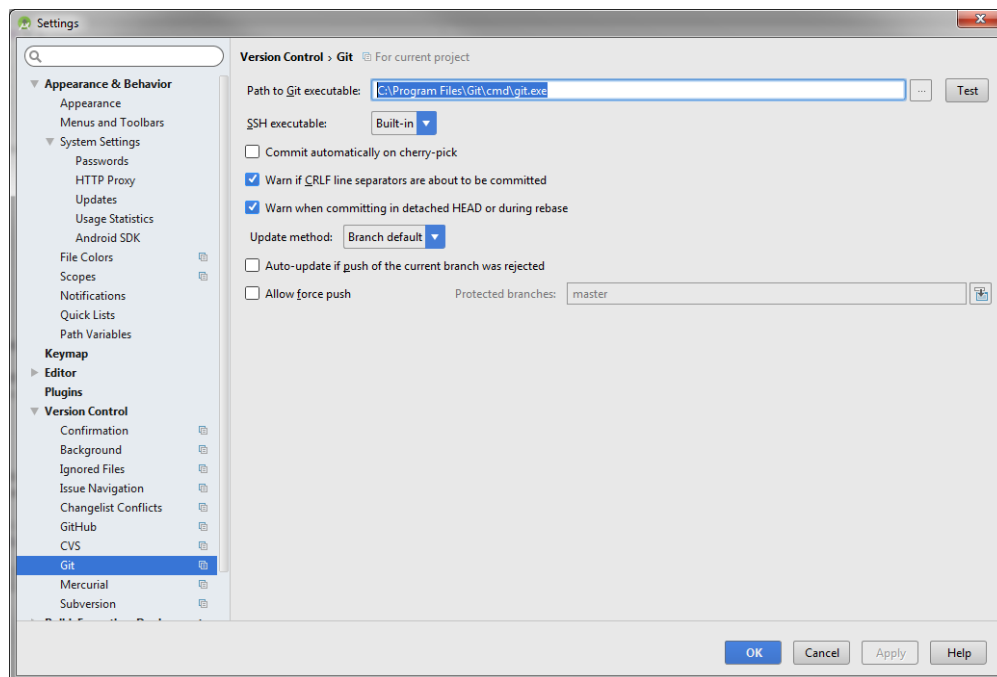


Abbildung 1.29: Version Control Settings.

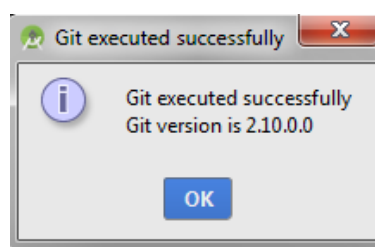


Abbildung 1.30: Git erfolgreich ausgeführt.

Anschließend schließen wir das aktuelle Fenster wieder und erstellen mit den nächsten Schritten ein lokales Git Repository in Android Studio. Dazu navigieren wir über VCS > Import into Version Control > Create Git Repository (Siehe Abb.)

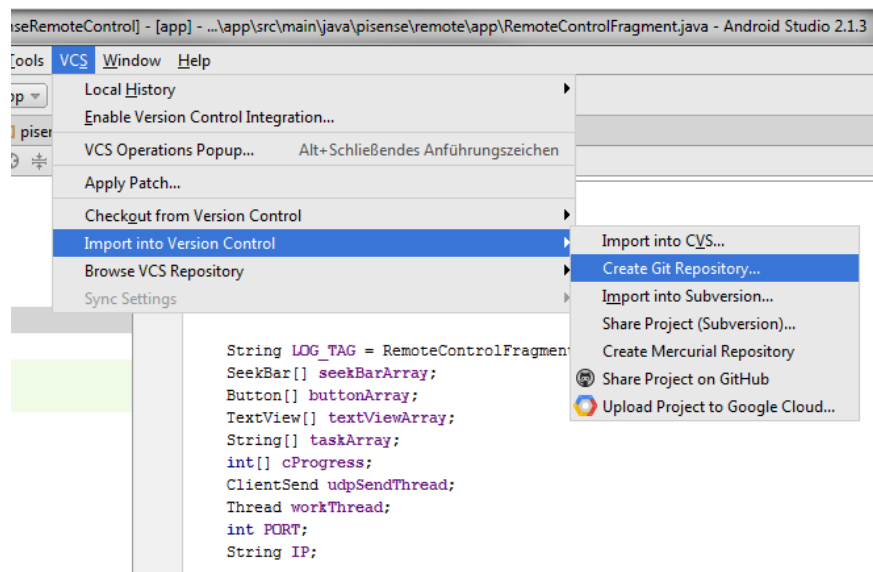


Abbildung 1.31: Menü Navigation „Create Git Repository“

Als nächstes designieren wir das Verzeichnis, welches von Git initialisiert werden soll. Hier bietet es sich an das Projektverzeichnis zu wählen. Im Folgenden öffnen wir den Pfad unseres Projektverzeichnisses und starten über das Rechtsklickmenü Git BASH (Siehe Abb. 1.32)

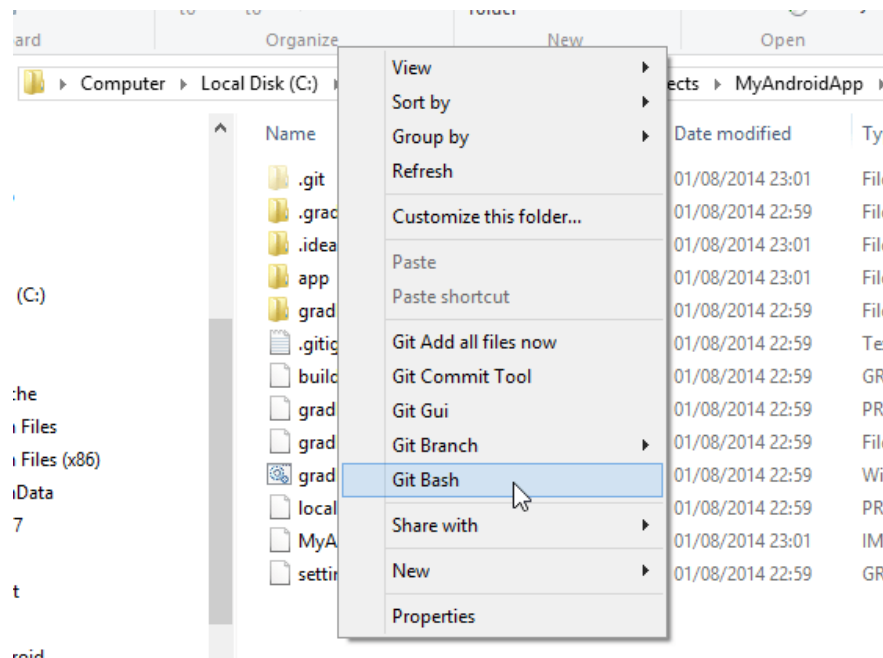


Abbildung 1.32: Starten von Git BASH im Projektverzeichnis

Im Git BASH angekommen möchten wir das gewünschte Remote Repository hinzufügen. Dies geschieht über den Befehl:

```
git remote add origin ssh://[user]@[server_address]/[git_repo_url]
```

In unserem Fall wäre die gewünschte Adresse (aus Atreus):

```
git@atreus.informatik.uni-tuebingen.de:Programmierprojekt2016/QuadrocopterAndroid.git.
```

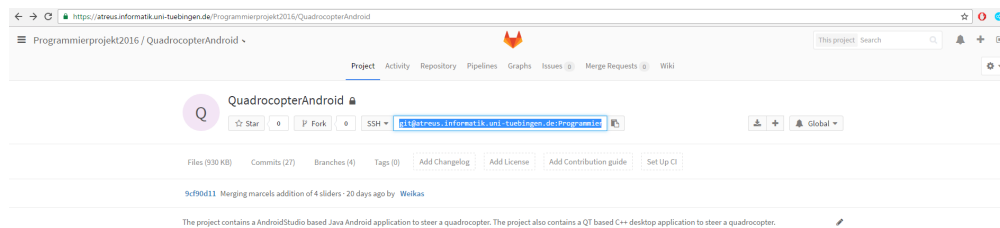


Abbildung 1.33: SSH Adresse des Repositories in Atreus.

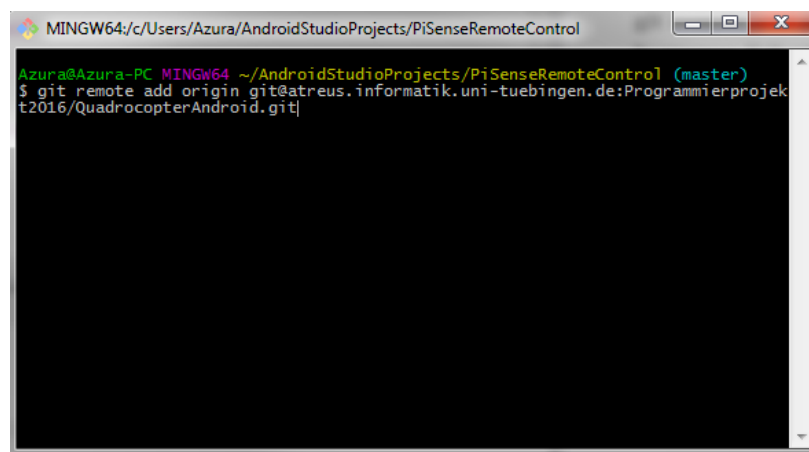


Abbildung 1.34: Hinzufügen eines Remote Repositories in Git BASH.

Um nun erfolgreich an der bereits gestellten App weiter programmieren zu können muss das Projekt erst von Git geklont werden. Dies ist möglich über VCS > Checkout from Version Control > Git. Im folgenden Fenster tragen wir die Ziel URL ein, sowie den Namen des Branches den wir klonen wollen (Siehe Abb. 1.35).

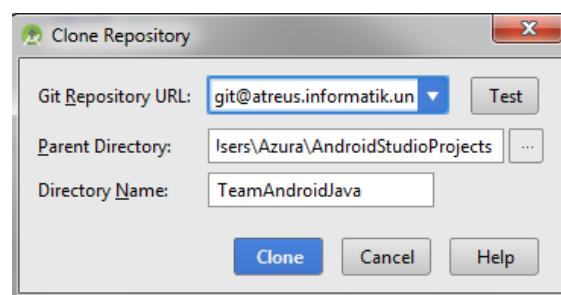
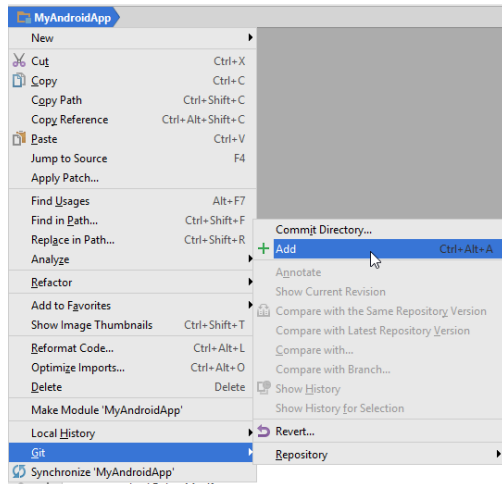
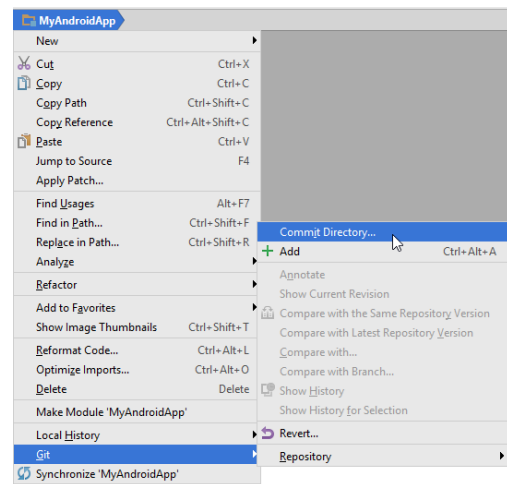


Abbildung 1.35: Ein Projekt mit Android Studio von Git klonen.

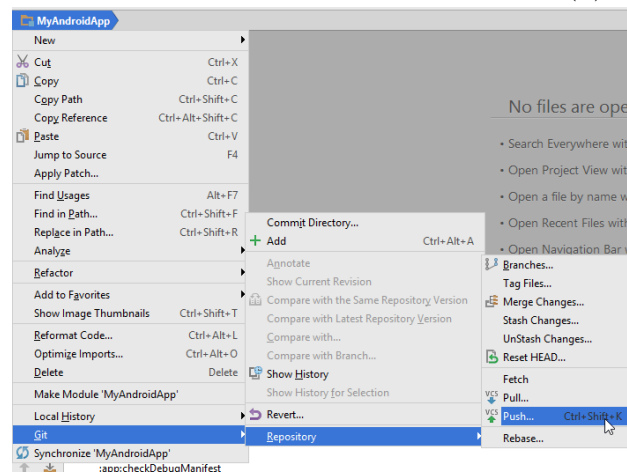
Der Ablauf eines erfolgreichen Commits ist wie folgt: Hinzufügen der Daten zum Git Repository > Comitten des Verzeichnisses und hinterlassen einer Commit Nachricht > Pushen aller Änderungen auf das Remote Repository und angegebenen Branch.



(a) Add



(b) Commit



(c) Push.

Abbildung 1.36: Menü Navigation zu Add, Commit und Push.

2 Android Applikation

2.1 Einleitung

2.1.1 Ressourcen

Effektives Ressourcen-Management in Android Studio gestaltet die Entwicklung einer App sehr unkompliziert und übersichtlich. Sämtliche Informationen über GUI Elemente, Strings, Menübausteine oder auch Präferenzen werden in XML Dokumente leicht zugänglich abgelegt und kommen der Lesbarkeit des Codes zugute.

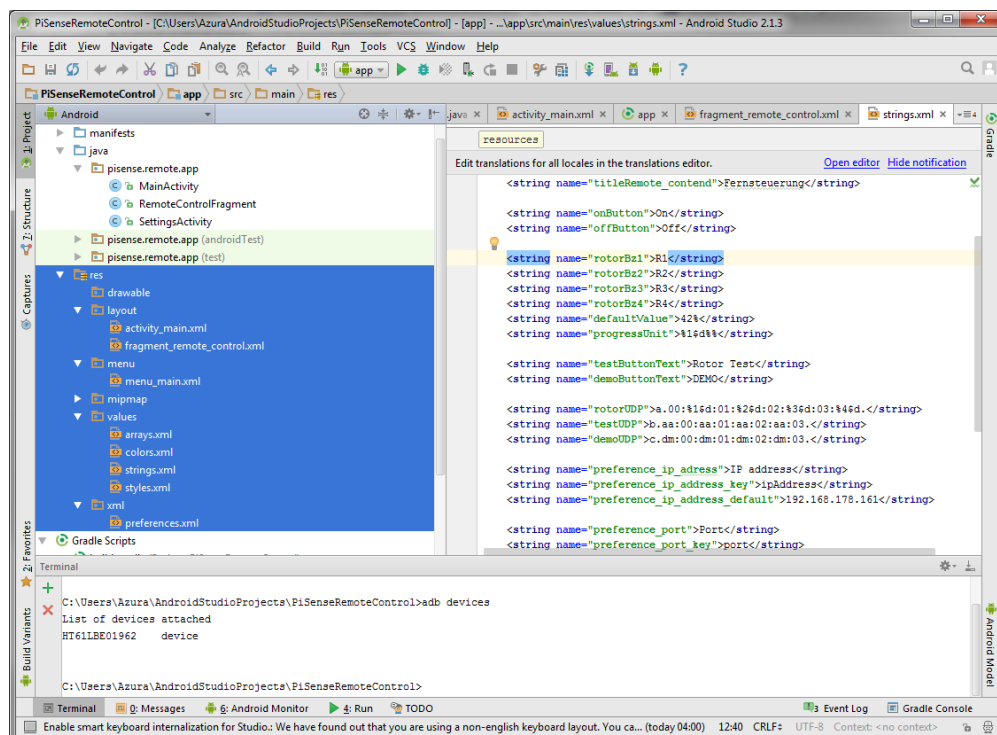


Abbildung 2.1: Beispielhafte Darstellung der String Ressource unserer App.

2.1.2 Activity

Eine Activity ist ein Bestandteil einer Anwendung, die einen Bildschirm zur Verfügung stellt, mit dem der Benutzer interagieren kann. Im Regelfall besteht eine App aus mehreren Activities die lose miteinander verbunden sind. Jede Activity besitzt die Fähigkeit eine andere Activity zu starten, dabei wird die Aktuelle gestoppt und die Gestartete rückt in den Fokus. Der Zustand der gestoppten Activity bleibt jedoch im Android System erhalten. Dies ist der Moment wenn sogenannte Lifecycle-Callbacks in Kraft treten. Diese informieren die Activity über eintretende Zustandsänderungen und geben die Gelegenheit bestimmte, notwendige Arbeit zu verrichten, bevor diese Zustandsänderung eintritt.

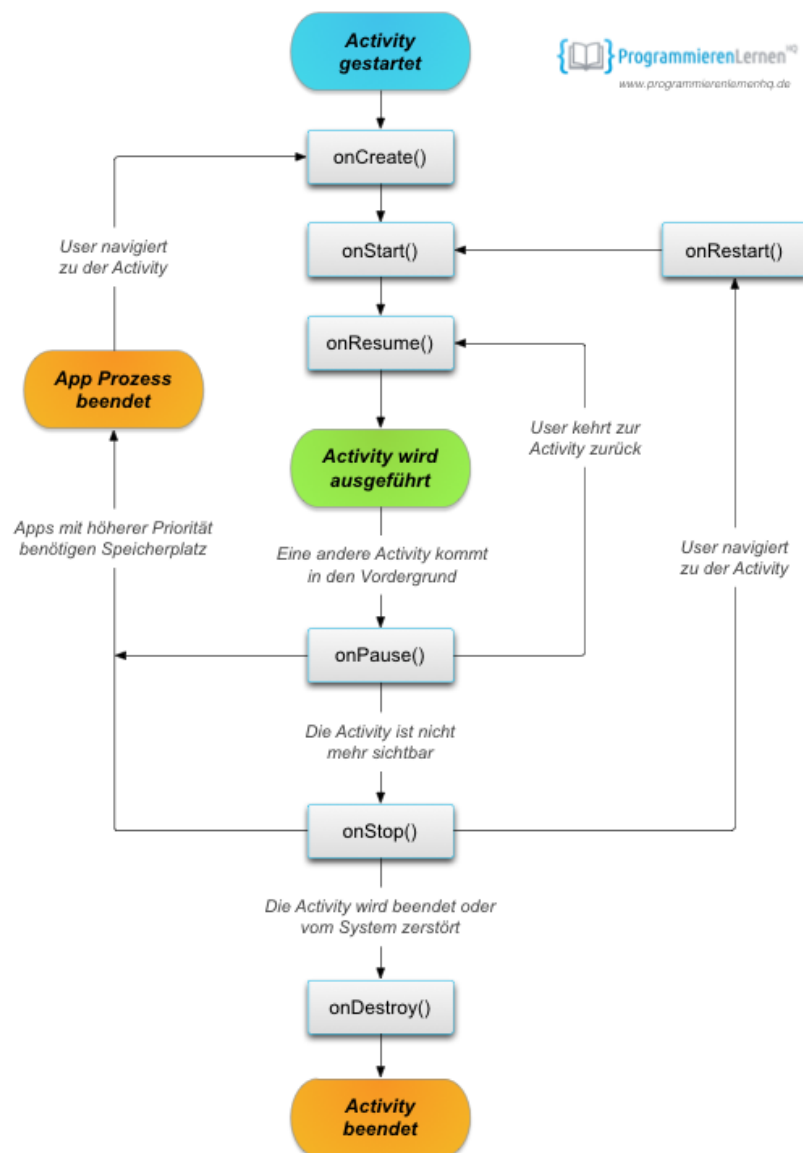


Abbildung 2.2: Callback-Methoden des Activity Lifecycle in Android.

Achtung! Implementiert man eine dieser Lifecycle-Methoden, ist es zwingen notwendig zuerst die Implementierung der Basisklasse aufzurufen.

```
protected void onPause() {
    super.onPause(); // Aufrufen der Implementierung der Superklasse
    // Hier folgt der eigene Code
}
```

2.1.3 Fragment

Bei Fragmenten handelt es sich um Modulare Bereiche einer Activity. Diese verfügen über ihre eigenen Lifecycle und Callback-Methoden. Der übergeordnete Activity-Lifecycle beeinflusst den Fragment-Lifecycle direkt. Das heißt, pausiert bzw. wird die Activity zerstört, dann tut dies auch das zugehörige Fragment. Eine Activity kann über mehrere Fragmente verfügen und diese können auch in mehreren Activities wiederverwendet werden. Fragmente müssen nicht zwingend Teil des Activity Layouts sein, sie können auch als unsichtbare Arbeiter eingegliedert werden.

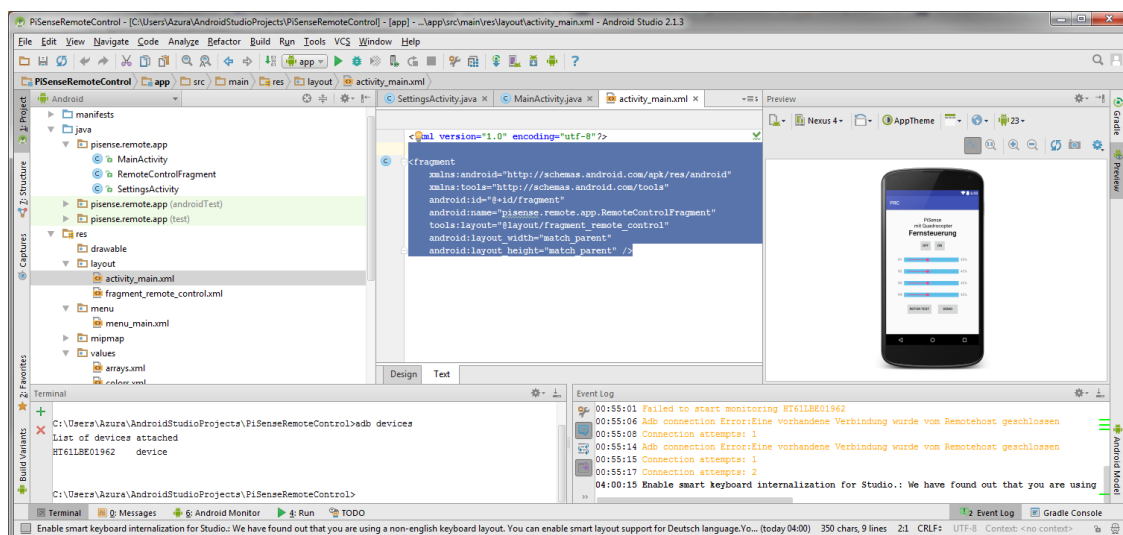


Abbildung 2.3: Activity wird vollständig mit Fragment Layout ausgefüllt.

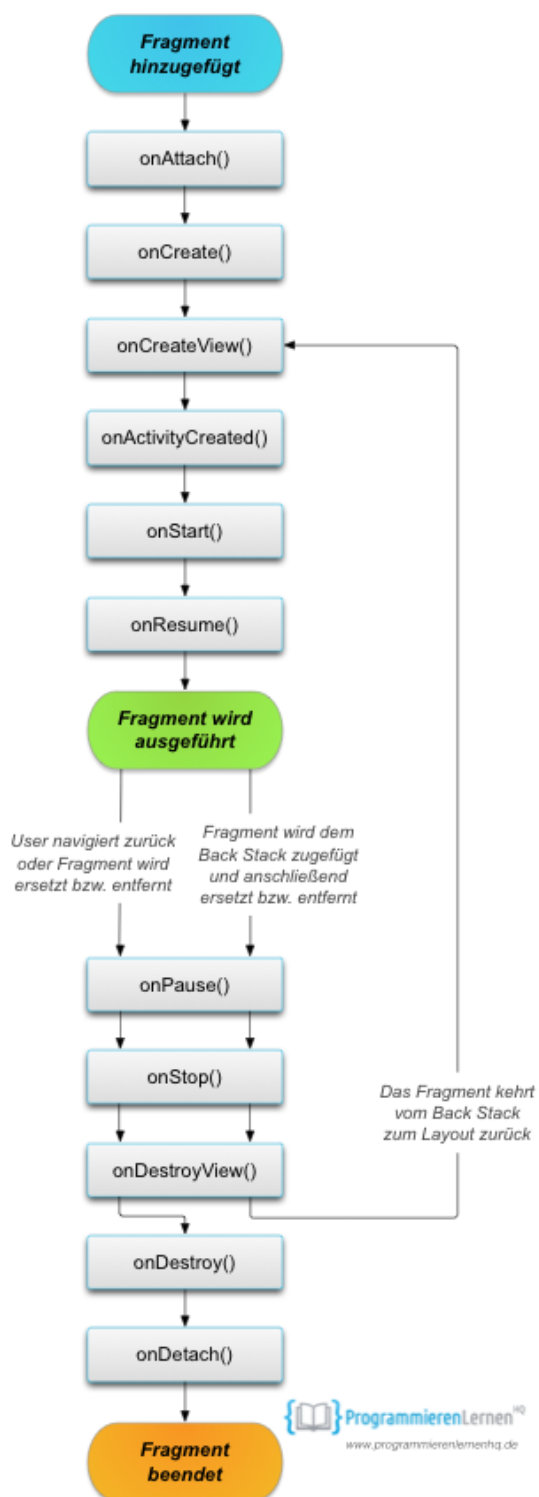


Abbildung 2.4: Callback-Methoden des Fragment Lifecycle in Android.

2.2 Layout

Das Layout der App wurde vollständig mithilfe des GUI Editors von Android Studio erstellt. Nach Erstellung der notwendigen layout.xml erzeugt dieser automatisch das passende XML Gegenstück zum jeweiligen GUI Element. Die in der App verwendete Oberfläche gestaltet sich wenig komplex, hierbei handelt es sich mehrere ineinander geschachtelte horizontale und vertikale Layouts. Eingefügt wurden jeweils die gewünschten Buttons, Textfelder und Seekbars.

Achtung! Editiert wird indem Layouts und GUI-Elemente in den Component Tree rechts oben gezogen werden.

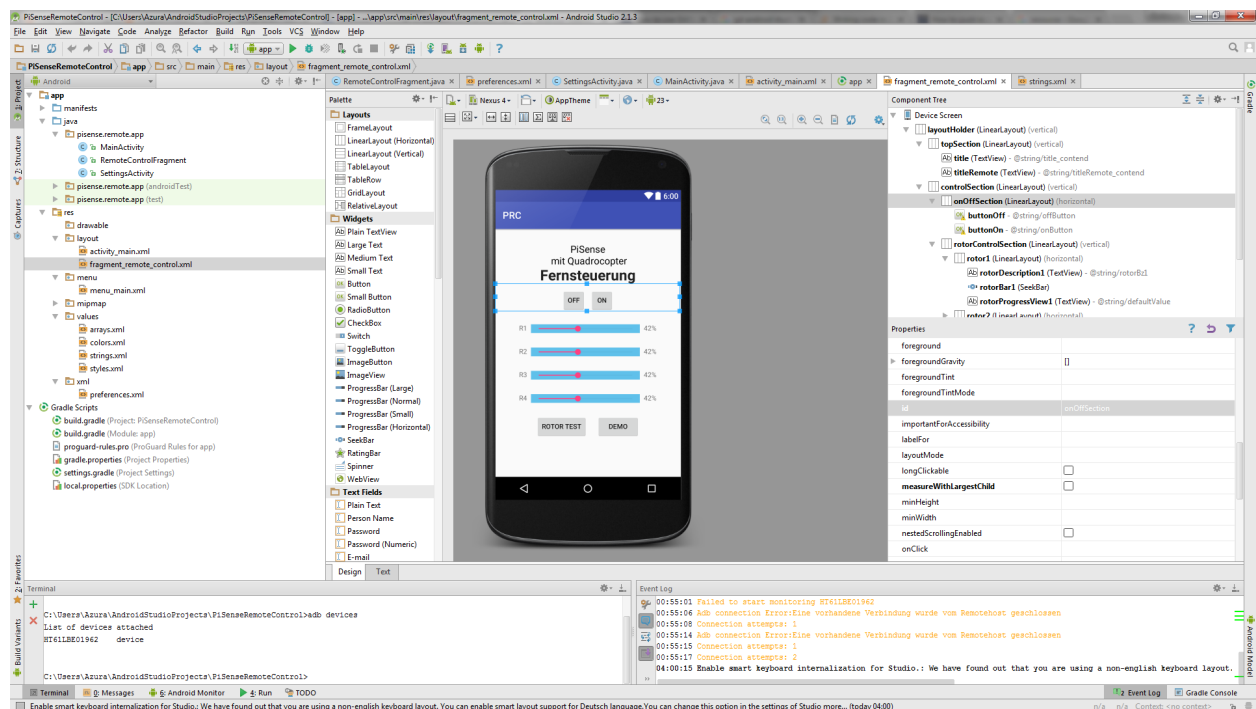


Abbildung 2.5: GUI Editor in Android Studio.

Eigenschaften der jeweiligen Elemente wird durch das Properties Menü unten rechts beeinflusst. Hier lassen sich Ausrichtung (gravity), Abstände (padding, margin...), Farbe und vieles mehr bestimmen. Eines der wichtigsten Attribute ist jedoch die **id** hiermit lassen sich Layout-Elemente im Editor benennen und im Code auf deren Layout zugreifen. Buttons, TextViews und SeekBars werden mithilfe der ID Objekte auf denen dann Arbeit verrichtet werden kann. Für Buttons und SeekBars war es des Weiteren notwendig Listener zu implementieren, die auf Zustandsänderungen reagieren können (Siehe Abb. 2.6).

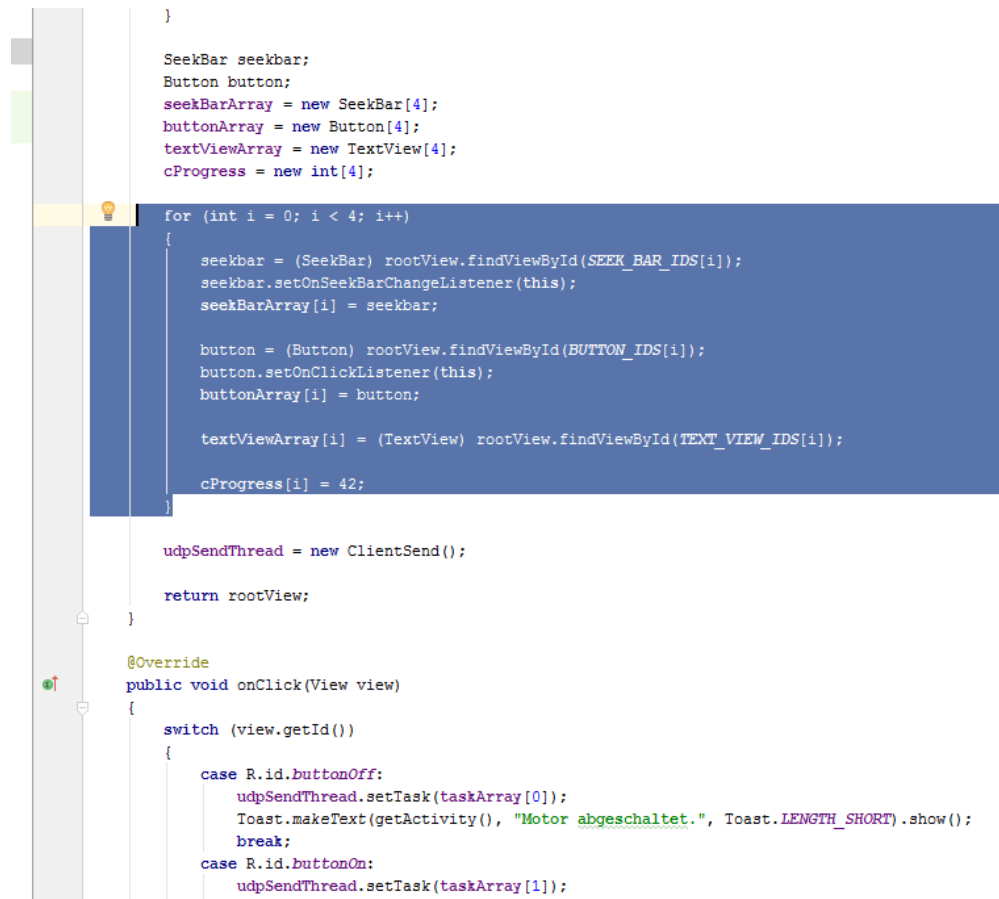


Abbildung 2.6: For-Schleife zur Bindung von Listnern an View-Objekte.

2.3 Funktion

Die angedachte Funktion der Applikation ist es Befehlsketten auf Aktivierung der jeweiligen Buttons per UDP an den Raspberry Pi zu schicken. Dies geschieht mithilfe eines separaten Threads der auf einem Objekt der inneren Klasse ClientSend ausgeführt wird. Diese implementiert das Interface Runnable und muss somit eine Run() Methode aufweisen, die später vom Thread aufgerufen wird. Die volatile String Variable task wird hierbei von der

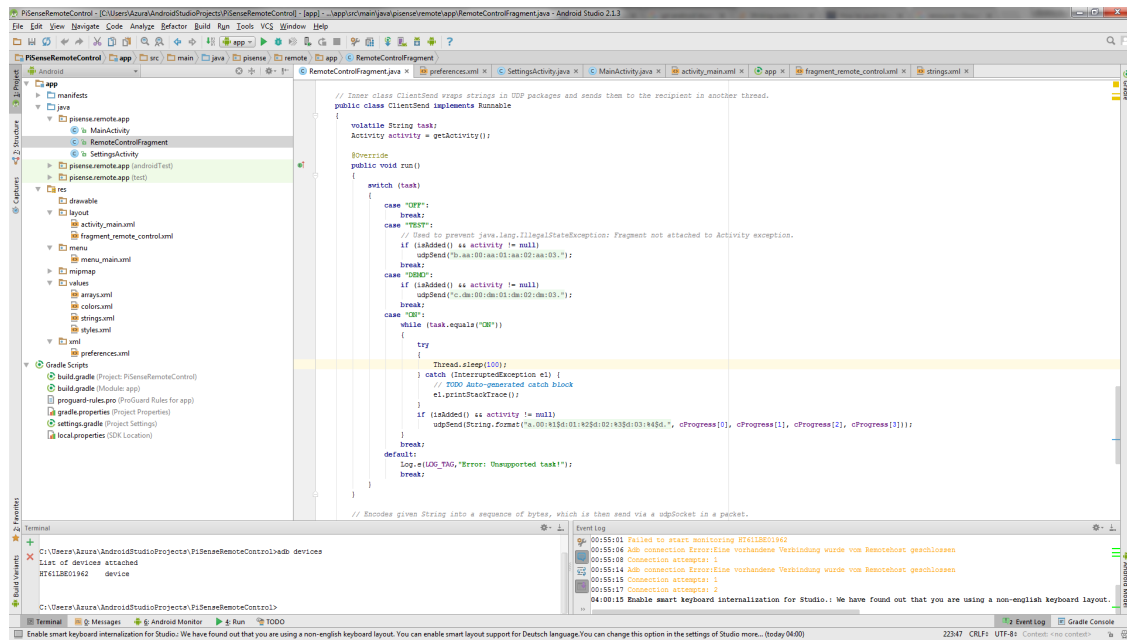


Abbildung 2.7: Innere Arbeiterklasse ClientSend.

Hauptklasse so manipuliert, dass der gewünschte Befehl per UDP weiter gesendet werden kann. Die Befehlsketten sind jeweils in der String resource.xml abgespeichert und werden von der udpSend Methode in ihre Byte-Sequenz zerlegt bevor sie als packet weitergeschickt werden.

```
// Encodes given String into a sequence of bytes, which is then send via a udpSocket in a packet.
public void udpSend (String stringToSend)
{
    try
    {
        DatagramSocket udpSocket = new DatagramSocket(null);
        // Address has to be set to reusable, because there is a minor timeout which prevents using the same address in quick succession.
        udpSocket.setReuseAddress(true);
        udpSocket.bind(new InetSocketAddress(PORT));
        InetAddress serverAddress = InetAddress.getByName(IP);
        byte[] buf = (stringToSend).getBytes();
        DatagramPacket packet = new DatagramPacket(buf, buf.length, serverAddress, PORT);
        udpSocket.send(packet);
        // Prevents memory leak, which is caused by opening too many sockets.
        udpSocket.close();
    } catch (SocketException e) {
        Log.e("Udp:", "Socket Error:", e);
    } catch (IOException e) {
        Log.e("Udp Send:", "IO Error:", e);
    }
}
```

Abbildung 2.8: Methode udpSend.

Da es sich bei der Manipulation der Rotorgeschwindigkeit nicht um statische Daten handelt wurden Klassenvariablen eingeführt, die Veränderung des Seekbar Fortschritts unmittelbar abspeichern. Durch das verwenden der String.Format Methode werden Dezimalwerte nachträglich in die Befehlskette eingefügt und in einer loop gesendet. Auf diese Weise lassen sich die Rotoren bis zum Abschalten beliebig beeinflussen.

```
@Override
public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)
{
    switch (seekBar.getId())
    {
        case R.id.rotorBar1:
            textViewArray[0].setText("{progress}%");
            cProgress[0] = progress;
            break;
        case R.id.rotorBar2:
            textViewArray[1].setText("{progress}%");
            cProgress[1] = progress;
            break;
        case R.id.rotorBar3:
            textViewArray[2].setText("{progress}%");
            cProgress[2] = progress;
            break;
        case R.id.rotorBar4:
            textViewArray[3].setText("{progress}%");
            cProgress[3] = progress;
            break;
    }
}
```

Abbildung 2.9: Abspeicherung des Seekbarfortschritts in der OnProgressChanged-Methode.

Eine letzte Funktionalität der App ist das Abspeichern einer bevorzugten IP-Adresse, sowie eines Ports. Dies ermöglicht das Verbinden der App auch mit anderen Geräten. Es gibt auch einen Default Modus der fest auf die IP und den Port des im Projekt verwendeten Quadrocopters eingestellt ist. Hierbei handelt es sich um eine Checkbox zum de-/und aktivieren. Im Gegensatz zu normalen Layouts bezieht die SettingsActivity ihr Aussehen aus einer preferences.xml. Diese SharedPreferences XML speichert die eingegebenen Einstellungen auch nach verlassen der App.

```
// Selection of chosen IP and port out of SharedPreferences
SharedPreferences sPrefs = PreferenceManager.getDefaultSharedPreferences(getActivity());
String prefIpKey = "ipAddress";
String prefIpDefault = "192.168.178.161";
String ip = sPrefs.getString(prefIpKey, prefIpDefault);

String prefPortKey = "port";
String prefPortDefault = "4999";
String port = sPrefs.getString(prefPortKey, prefPortDefault);

// Checking mode of display.
String prefDefaultModeKey = "default";
Boolean defaultMode = sPrefs.getBoolean(prefDefaultModeKey, false);
```

Abbildung 2.10: Auslesen der SharedPreferences XML und Prüfen auf Darstellungsmodus.