Laz To Apk
Erste Schritte

Datum 06/2020
Windows 7

Android Programmierung mithilfe von LazToApk

Inhaltsverzeichnis

Vorwort

Ich bin kein Profiprogrammierer. Alle Schritte auf eigene Gefahr. Ich beschreibe hier den Weg, der bei mir funktioniert hat. Trotzdem besteht kein Anspruch auf Richtigkeit.

Diese Beschreibung dient als Unterstützung, weil ich weiß das Apk erstellen mit Lazarus eine nervenaufreibende Sache ist.

Ich nutze LazToApk als Unterstützung, dass die Komponenten downloadet und installiert. Mit etwas Glück, kann man danach direkt starten (Installationanleitung von LazToApk beachten!!!).
Bevor man beginnt, sollte man sicherstellen dass ausreichend Speicherplatz zur Verfügung steht! Der Installationsprozess dauert einige Zeit!

Falls man, wie ich, bereits vorher versuche unternommen hat und das System nicht mehr „frei“ von SDK, NDK Installationen u.ä. ist, muss man etwas herumprobieren
Alle Komponenten müssen zueinander passen. Wenn eine Komponente eine andere Version hat o.ä., dann kann es sein, dass es nicht mehr richtig funktioniert. Manche Konstellationen beißen sich. Deshalb immer Stepp by Stepp.

Meine Verwendeten Daten:

(https://sourceforge.net/p/laztoapk/wiki/Home/)
Hier beschriebene Anleitung basiert auf folgende Daten, und sind evtl. nicht die aktuellsten:

Version LazToApk:	0.9.0.41
SDK Build:	29
NDK Api:	22
NDK:	r17c
Virtual Device:	Android 7, 7.0, API Level 24, CPU/ABI: ARM armeabi-v7a
Demo App:	Emulator funktioniert; Motorola X Play funktioniert Android ?; Nokia 2.3 Android 10 funktioniert
OS:	Windows 7
Test-Datum:	2020
laz4android:	2.0.0

Die nötigen Schritte:

1) Installation

setup_laztoapk_vXXXXX.exe starten: Das Tool erstellt ein neues Stammverzeichnis auf dem Laufwerk „laztoapk“. Das Tool downloadet die meisten Komponenten, speichert und entpackt sie in dem Unterordner „laztoapk/Downloads“.
Dieser Prozess dauert eine Weile. Danach befindet sich im Ordner laztoapk eine PDF „Lazarus and Android.pdf“. Diese PDF Installationanleitung muss unbedingt beachtet werden! (Ich gehe hier nicht mehr darauf ein!) Während „Step 2 Download&Instal LazToApk“ schon erledigt ist, muss danach noch separat

Java JDK8 (jdk-8uxxx-windows-i586.exe)

installiert werden. Nach derzeitigen Stand (06/2020) kann man Java JDK 8 nur noch nach Registration bei Oracle downloaden. Bezieht man diese Datei aus anderer Quelle, sollte sichergestellt sein das es die richtige Datei ist (und evtl. auf Viren geprüft werden). Meine Datei nennt sich jdk-8u251-windows-i586.exe (32 Bit; nur 32 Bit verwenden!!) und hat eine Größe von 201,17 MB. Die Installation anschließend starten und die Standartpfade beibehalten.
Es muss unbedingt diese Java JDK8 sein, sonst funktionierts nicht.

Fehler?

In einigen früheren Versuchen hat laztoapk nicht alle Komponenten downloaden/installieren können. Das liegt daran, dass sich evtl. auch Online Pfade verändern können. Ich empfehle jeweils die aktuellste laztoapk zu verwenden um dies zu vermeiden. Sollte trotzdem nicht alles heruntergeladen werden, dann muss man dies per Hand machen. Dazu im Internet nach der fehlenden Komponente suchen und selbst installieren.

Beim Start von laztoapk sucht das Tool selbst die richtigen Pfad und setzt sie richtig in Lazarus ein. Das Programm gibt evtl. auch eine Fehlermeldung sollte es etwas nicht gefunden haben. Dann die richtige Version downloaden und im Verzeichnis laztoapk/Downloads entpacken und installieren in diesem Verzeichnis.

Es müssen vorhanden sein:
- android ndk r17c
- android sdk windows
- apache ant 1.10.3
- gradle 4.10
- laz4android2.0.0

Evtl. entstehen auch leichte Versions Unterschiede, insbesondere wenn die laztoapk Version steigt, könnte sich auch laz4android verändern. Ich verwende laztoapk 0.9.0.41. Die Installationsanleitung von laztoapk das Abbild der Version 0.9.0.38 (geringfügige Unterschiede).

Auch wenn man mit dem SDK Manager arbeitet, sind evtl. manche Versionen nicht mehr verfügbar u.ä. dann stimmen die Versionen mit sdk, ndk nicht zu Hundertprozent. Aber es gibt keine Garantie das dann noch alles funktioniert. Neueres kann auch funktionieren. Die Java Version sollte aber identisch sein.

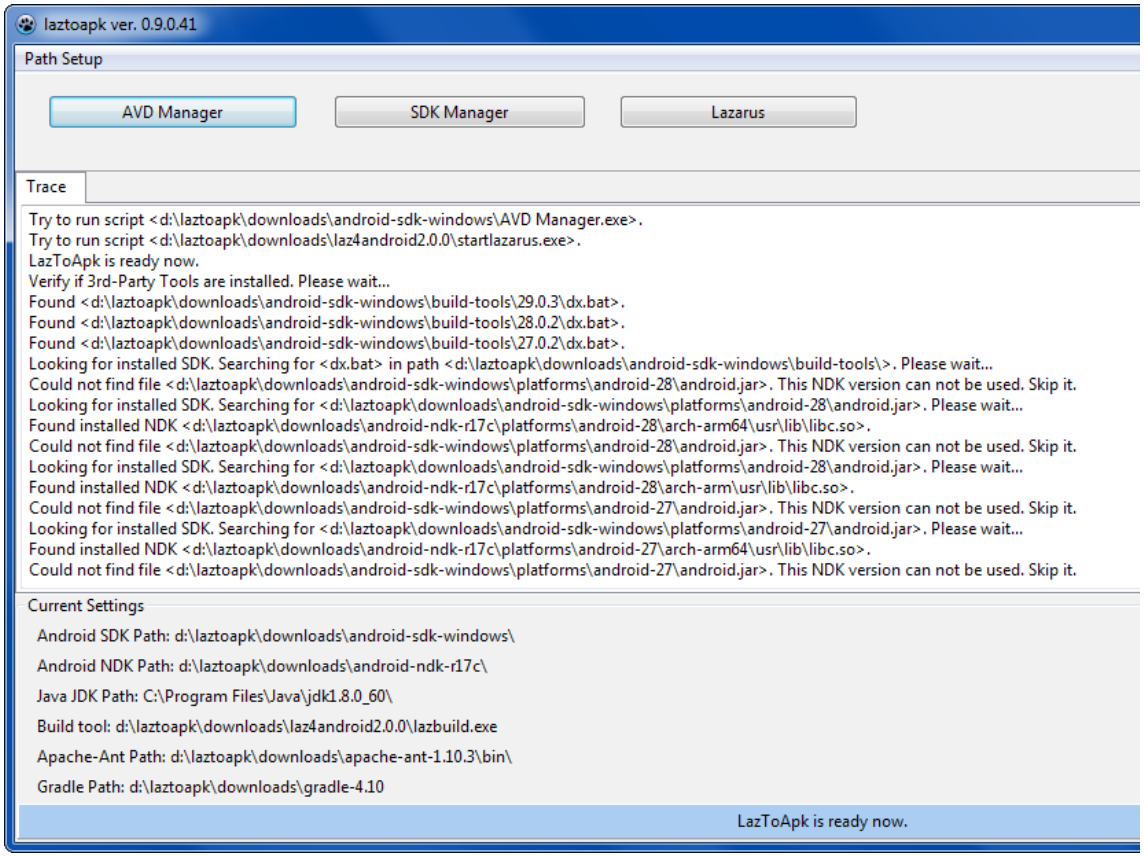
Vorsicht beim SDK Manager: Es ist bereits etliches Vormarkiert für die Installation, dass aber nicht benötigt wird. Hier genau aufpassen! Sonst dauert die Installation unendlich lange und der Speicherplatz wird unnötig aufgebraucht.

Schluss

Im Optimalfall sollte alles problemlos durchlaufen und auch das Java JDK 8 installiert sein.
Laztoapk und das setup arbeiten recht lange bei der Installation. Die „Finish“ Meldung sollte abgewartet werden.
Danach lazToApk, setup schließen. Auch das erstmalige starten kann noch etwas dauern.
Nach dem Neustart von LazToApk sollte in der untersten Leiste „LazToApk is ready now“ stehen. Ansonsten erscheinen hier Fehlermeldungen die zunächst bearbeitet werden müssen.
Hat man etwas installiert und LazToApk kann es nicht finden, dann im Menü „Path Setup“ die Pfade einfügen!

Ist man der Installationsanleitung von LazToApk komplett gefolgt, benötigen wir für die Programmierung nur noch den Button „Lazarus“. Und gelegentlich den Button „AVD Manager“. SDK Manager wird nicht mehr benötigt.

Bei mir sieht alles Fertig dann so aus:



2) AVD Manager

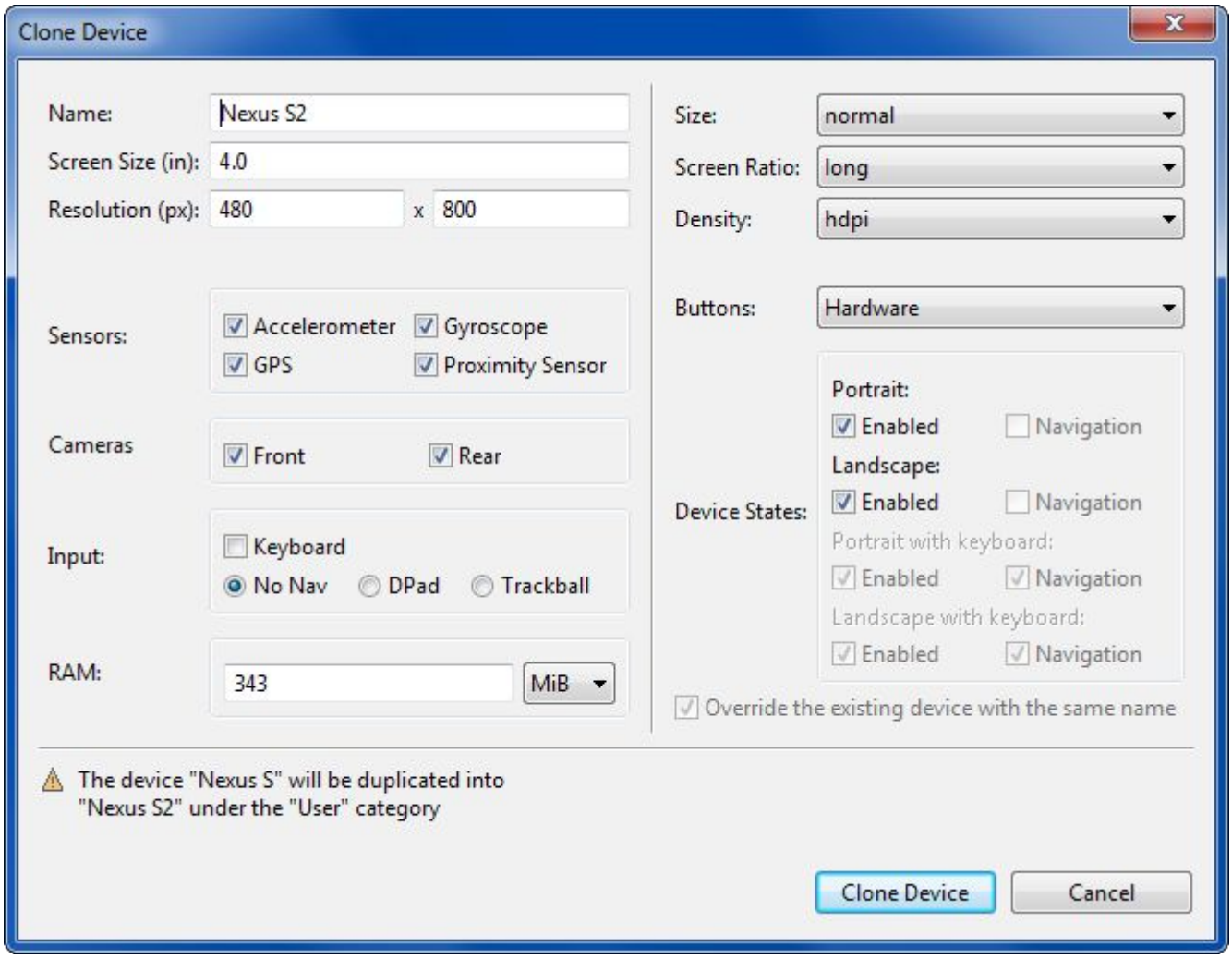
Bevor es nun an das Programmieren geht, muss noch das „Smartphone emuliert“ werden bzw. die Einstellungen vorgenommen und erstellt werden.

In laztoapk den AVD Manager aufrufen.

Den Reiter „Device Definitions“ aufrufen. Hier erstellen wir nun einen Device. Dies ist notwendig, sonst können wir später in Lazarus die apk nicht an das simulierte Smartphone schicken und erhalten eine Fehlermeldung.

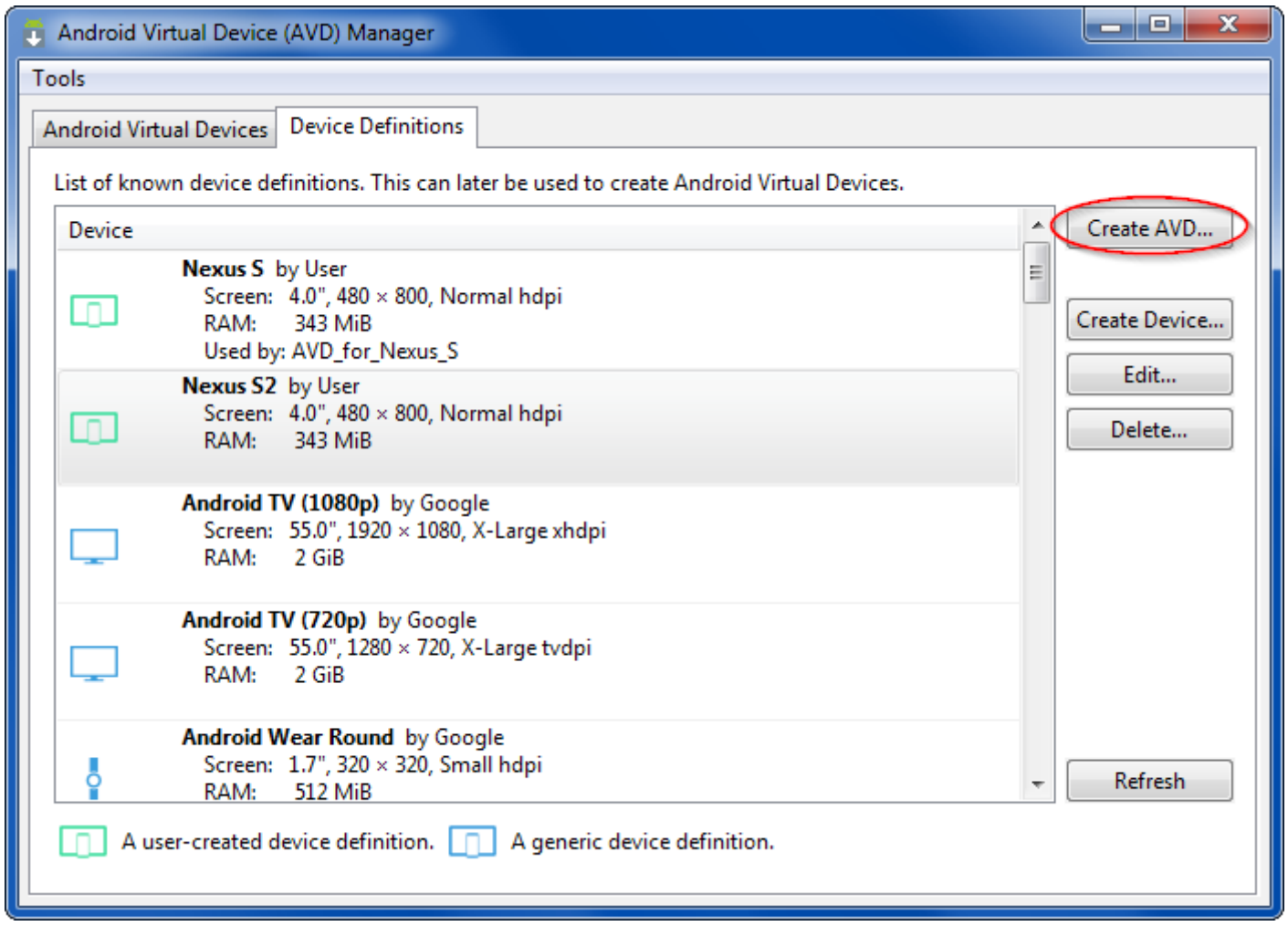
Man kann nun direkt „Create Device“ aufrufen. Noch besser und schneller geht es aber über die Device Liste:

Zum Beispiel:
DOPPELKLICK auf “Nexus S“. Dann werden die Werte automatisch in die Eingabefelder übernommen:

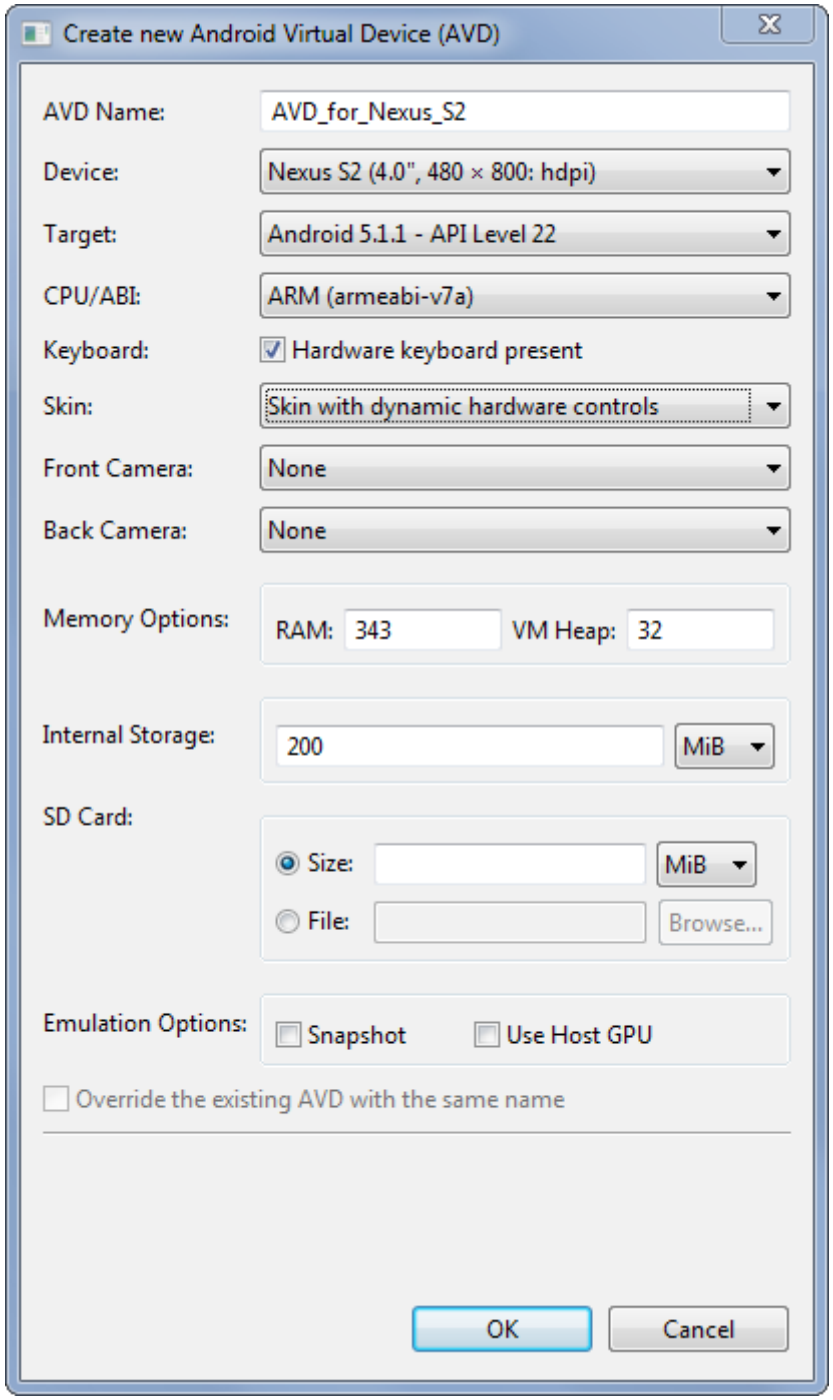


Weitere Angaben/Änderungen sind nicht nötig. Dann „Clone Device“.

Im nächsten Schritt den duplizierten Device auswählen:

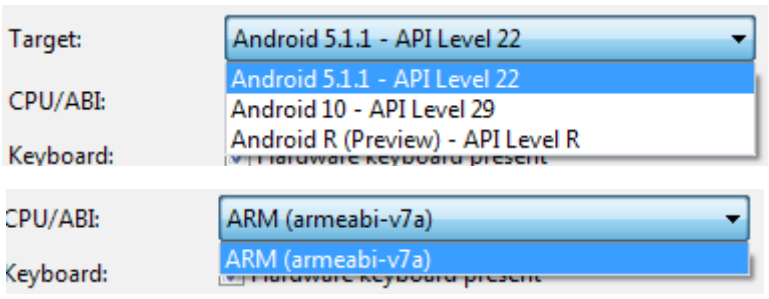


Und auf „Create AVD“ klicken. Man kommt zu folgenden Fenster:



Hier muss bei Skin „Skin with dynamic hardware controls“ ausgewählt werden. In dem Fenster sind wir Fertig.

ABER diese Daten müssen wir uns für später merken!:



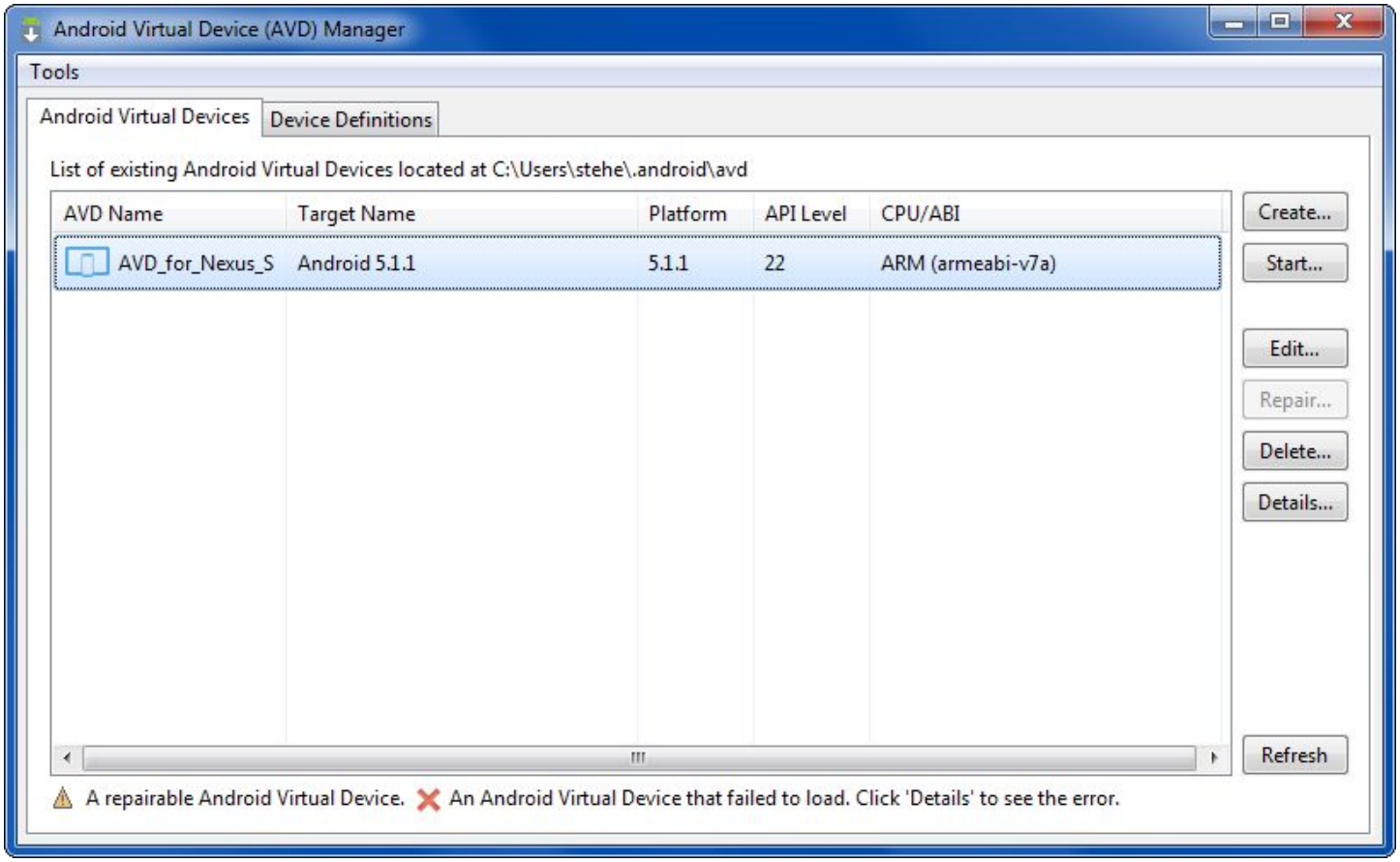
Je nach vorhigen Installationen steht hier unterschiedliches. Am besten wählt man hier „Android 5.1.1. - API Level 22“ bzw. das niedrigste.

DIESE Daten müssen später in Lazarus übereinstimmen – sonst klappts nicht!

(Wählt man anderes aus z.B. Android 10 – API Level 29 bzw. Änderung der CPU/ABI muss sowohl die Emulation als auch der Lazarus entsprechende Compiler usw. haben, sonst klappt es nicht. In diesem Beispiel ist es nicht vorhanden, klappt also nicht.)

Dann OK. Die darauffolgende Kurzinfo mit der Zusammenfassung bestätigen.

Dann sind wir fertig. Es sieht dann bei mir so aus (Tab „Android Virtual Devices“):



Bei „Start“ wird das Smartphone simuliert. Der erste Start dauert in der Regel länger. Prinzipiell je nach System ist das simulierte Android langsam/schnell…

Bei „Edit“ kann man noch nachträglich die Werte verändern (Target, CPU/ABI) bzw. nochmal nachlesen!

Sollte es später keine Probleme mehr geben und ist man mit der Emulation zufrieden, muss man im AVD Manager nichts mehr machen. Außer evtl. mal auf „start“ klicken.

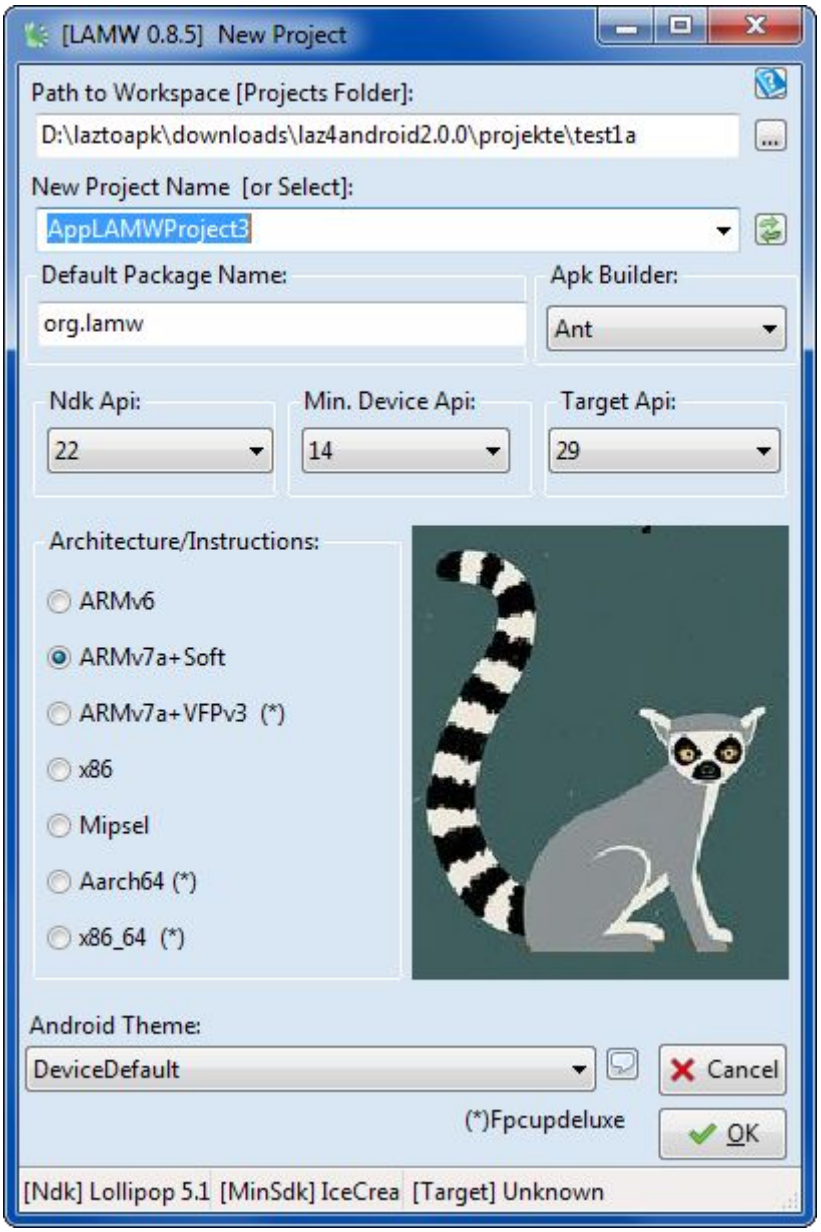
In dem Beispiel habe ich „Nexus S“ als Vorgabe gewählt, weil es den üblichen Smartphone Geräten sehr ähnlich ist.
Man kann auch ein anderes Gerät als Vorgabe wählen wenn man z.B. noch zusätzliche Hardware Tasten simulieren möchte, für einen TV oder Smartwatch usw. Ich habe das aber nicht getestet.

3) Lazarus

Lazarus starten

Projekt erstellen

1. Datei; Neu …
2. LAMW GUI Android Module



Lamw 0.8.5 New Projekt
Path to Workspace: Projektordner angeben
New Project Name: Programmname
Default Package Name: org.lamw
Apk Builder: Gradle
NDK Api: 22
NKD Version: 11
Min Device Api: 14
Target Api: 29
Architecture: ARMv7a+Soft
Android Theme: DeviceDefault

Das Projektverzeichnis kann z.B. hier sein: laztoapk\downloads\laz4android2.0.0\projekte\deinprojekt

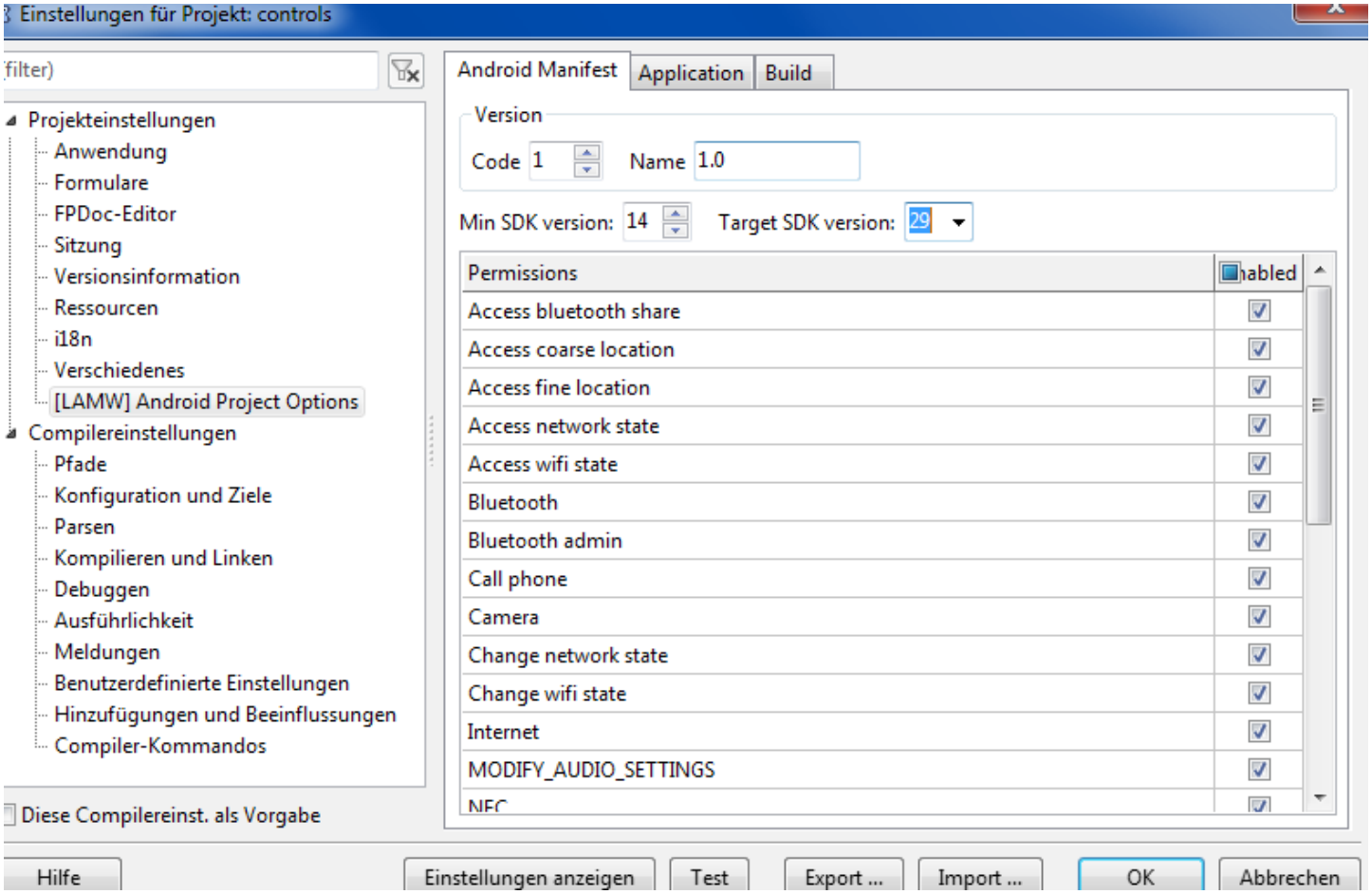
Einstellungen lassen in sich später unter Lazarus bei Werkzeuge/LAMW Android Modul Wizard/Paths settings verändern

Ein vorheriges Projekt aufrufen

Menü:
Projekt; Projekt öffnen; Programm Hauptverzeichnis öffnen; Ordner“jni“ → controls.lpi öffnen

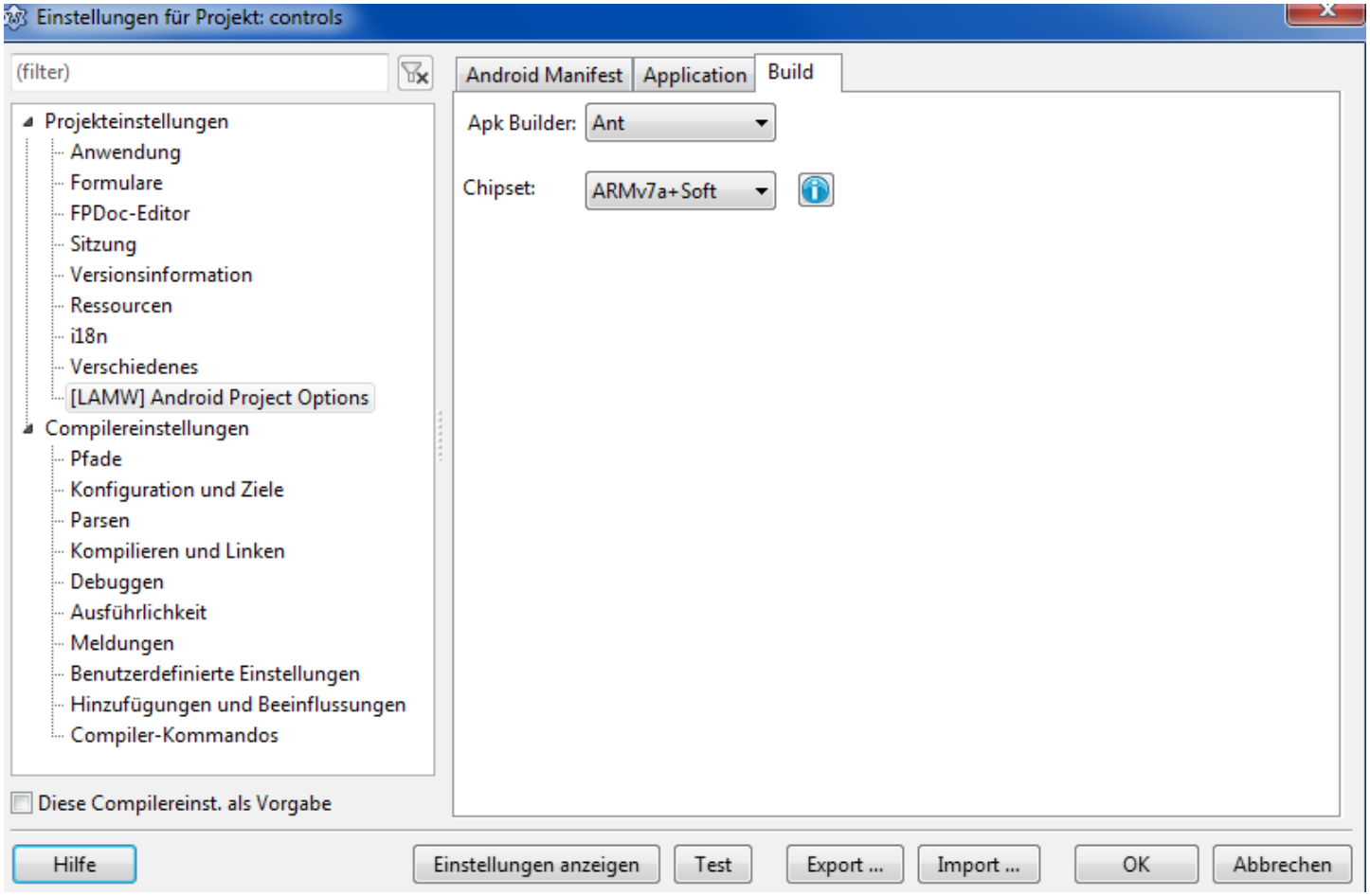
Projekteinstellungen

Man kann die Einstellungen, die man beim Projektanlegen gemacht hat, später noch verändern:
Menü:
Projekt/Projekteinstellung/LAMW (linke seite) und auf der rechten Seite mit den drei Tabs.



Das sind hier die Standarteinstellungen.

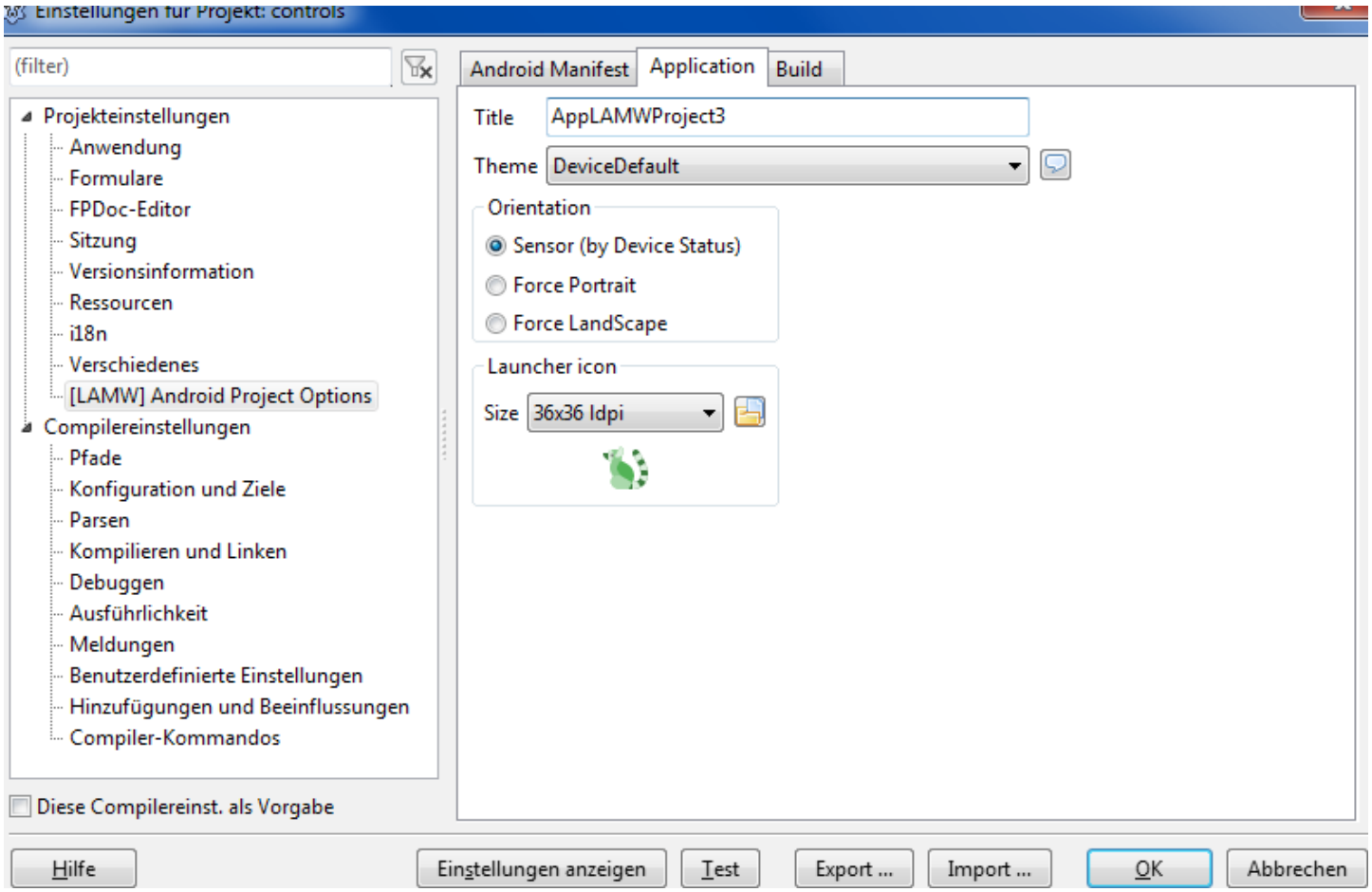
Target SDK version
Unterschiede in der Funktionfähigkeit konnte ich nicht feststellen. Sowohl mit Target SDK version 29 als auch mit der Target SDK version 22 lief meine Testanwendung.



ApK Builder
ApK Builder: Hier funktioniert „Ant“.

Bei meinen Versuchen ging entweder mal „Ant“, dann mal „Gradle“ oder auch beides.
In dieser Dokumentation habe ich „Gradle“ nicht zum laufen gebracht. In früheren Testversionen von mir, war es auch mal umgekehrt.
„Ant“ scheint eine frühere Version zu sein und „Gradle“ der Nachfolger. „Ant“ scheint bei mir auch etwas schneller zu sein als „Gradle“. Was die Unterschiede sind und was es im Detail Bedeutet weiß ich nicht.
Wenn der Compiler einen Fehler bei Gradle hat, dann einfach mal auf Ant umstellen bzw. von Ant auf Gradle.

Chipset
Muss ARMv7a+Soft sein. Dieser muss mit dem AVD Manager bzw. mit den darin enthaltenen Device CPU/ABI übereinstimmen. In unserem Beispiel ist im AVD Manager (siehe auch Grafik weiter oben) zu sehen „ARM (armeabi-v7a)“. Beides stimmt überein.
Chipset muss auf jedenfall stimmen, sonst funktioniert gar nichts.



Hier sind nur die Standarteinstellungen ohne Veränderung, der Vollständigkeitshalber.

Hier kann man Title auswählen und ein eigenes Icon für die Anwendung. Dieses wird dann auf Android angezeigt.

Projekt compilieren und starten

Nicht wie üblich mit F9 sondern Strg+F1 (Start, LAMW Build Android Apk and run).

Sollte alles klappen, wird das Projekt nun erstellt und direkt an Android geschickt, installiert und gestartet. Evtl. wird man aufgefordert „Android emulators“ zu starten. Dieses Fenster muss immer beim ersten Start solange geöffnet bleiben bis die Android Anwendung im emulator gestartet ist bzw. der Emulator muss in Betrieb sein. Danach kann man das Fenster schließen.
Ggf. muss/kann man auch schon vorab über den AVD Manager die Emulation starten.

Die Emulation kann man während der Programmierung laufen lassen.

Apk manuell installieren auf der Emulation oder auf dem eigenen Smartphone

(Kurzer Hinweiß zu dem „Installation Tutorial for LazToApk“ Abschnitt „Step 08 Build your first android app“:
Mit LazToApk 0.9.0.41 und LazToApk 0.9.0.40 und ggf. höher ist der Abschnitt nicht mehr aktuell. Die Beschreibung trifft nur auf LazToApk 0.9.0.38 zu. Und muss ignoriert werden. Eine USB Übertragung über LazToApk ist nicht mehr möglich. (Stand 06/2020))

Wurde das Projekt erfolgreich compiliert, findet man die Apk Datei im „bin“-Verzeichnis des Projektverzeichnis.

Will man eigene Dateien dem Projekt hinzufügen z.B. eine Sqlite Datenbank, Bilder o.ä. dann gehören diese in das Projektverzeichnis „jni“

Emulator
Diese Apk Datei kann man per Drag-and-Drop in die emulation einfügen/installieren. Das geht dann automatisch.

Bei dem eigenen Smartphone:
Möglichkeit 1:
Anschließend eine kurze, grobe Schrittbeschreibung. Es gibt mehrere Wege und ist auch von Smartphone und Windows abhängig. Gute Smartphonekenntnisse sind von Vorteil.

PC

- Gerät mit dem PC verbinden (einfachste ist wahrscheinlich per USB. Aber auch Bluetooth, Wlan usw. ist möglich. Hauptsache es ist möglich eine Datei zu übertragen!). Beim anschließen mit USB auf dem Smartphone „als Speicher verwenden“ auswählen.
- Je nach Smartphone Gerät und Windowsversion kann man nun seine apk Datei über ein „externes Laufwerk“ dass das Smartphone darstellt, nun übertragen.
  (Evtl. auch über Software von Drittanbieter usw.)

Smartphone
Mit einem Datei Explorer die „heruntergeladene“ Datei nun aufrufen und installieren. Danach kann man auf dem Android Gerät das Programm normal starten.

Noch ein paar Worte hierzu:

- Viele Android Geräte haben von Haus aus keinen Datei Explorer. Dieser muss nachträglich installiert werden über den Play Store z.B. Total Commander. Damit kann man dann zu dem Pfad gehen wo man seine Datei abgelegt hat. Mit einem Klick auf die apk Datei wird diese installiert und im „Programmmenü“ wie alle andere Anwendungen angezeigt.
  - Android prüft von Haus aus eine apk Datei und gibt evtl Warnmeldungen aus dass diese Datei nicht von Google ist, zeigt Meldungen zu Berechtigungen usw. an.
- Evtl. muss man, bevor man eine Anwendung installieren kann den „Entwicklermodus“ des Android Gerätes freischalten. Standardmäßig ist dieser ausgeschaltet. Wie man diesen aktiviert, kann über die Anleitung des Gerätes erfahren. Evtl. ist dies aber auch nicht nötig, erst mal ohne versuchen. Einfache Anwendungen benötigen keinen Entwicklermodus.

Möglichkeit 2:
Webspacebesitzer können auch den Umweg über die eigene Webseite machen. Die Apk Datei auf den Server des Webspaces hochladen mit dem PC. Ggf. die Apk Datei in etwas sinnvolles umbenennen.

Auf dem Smartphone Online gehen und diese Apk Datei aufrufen bzw. herunterladen und installieren. Firefox gibt einen Hilfestellung dazu. Herunterladen, installieren, fertig!
Mit Chrome oder andere Browser vermutlich ähnlich.

Wie oben schon beschrieben, muss man dies zulassen in den Einstellungen. Auch meldet sich Android protect und will die Datei auf eigene Server laden und überprüfen lassen. Das ist in dem Fall alles nicht erforderlich, weil es ja unsere eigene Datei ist und daher -vertraulich- :)

Android

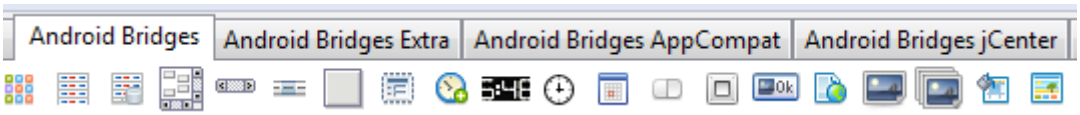
Auf dem Gerät selber kann man die App starten in dem man in den „Programmmenü“ geht. Wie bei den Smartphones üblich auf den mittigen Button drücken und den Projektnamen anklicken.
Entweder hat Lazarus das schon automatisch gemacht, oder man macht dies manuell.

Je nach Android Version und Gerät kann die Bedienung auch abweichen. Neuere Geräte haben keinen mittigen Button mehr und man muss von unten nach oben wischen u.ä. Auch kann sich die App je nach Androidversion anders Verhalten als in der Emulation. Das muss man vorher alles ausprobieren.

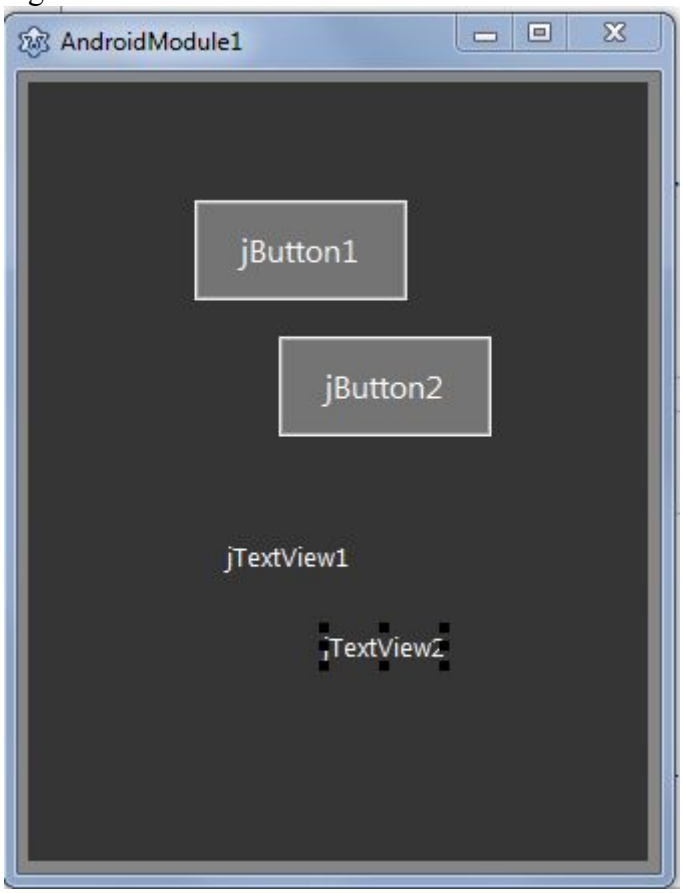
4) Lazarus: Hello World Android!

Nun kann mit Lazarus Android Anwendungen erstellt werden. Dabei kann man fast wie gewohnt mit den Komponenten und den Objektinspektor arbeiten und auch mit den Ereignissen. In manchen Bereichen muss man etwas umdenken. Aber viele Befehle sind ähnlich oder identisch.

Nun können in Lazarus alle Komponenten auf die Form gezogen werden, die mit „Android…“ beginnen. Alle anderen sollten nicht verwendet werden, wenn sie eine visuelle Elemente haben oder sich auf visuelle Elemente beziehen. „Uses math“ o.ä. sollte aber durchaus möglich sein.



Für die erste Beispielanwendung zwei Buttons und zwei Textfelder auf die Form ziehen:



Und compilieren und starten (Strg + F1 oder Start – „Lamw Build Android...“). Das sieht dann so aus:

Nach dem compilieren wird aber alles übereinander gelegt. Das ist kein Fehler, sondern bei jedem Element muss zusätzlich zwingend auch die Position mit angegeben werden.

Positionsangabe, zweites Beispiel

Jedes Element benötigt eine Horizontale und eine Vertikale Position Angabe!
Fehlt eine Angabe, wird Android die fehlende Angabe des Elements standardmäßig immer auf die Linke Bildseite setzen und/oder auf die obere Bildseite.

Die Positionsangabe ist **immer relativ** und nicht absolut!

Bei der Positionsangabe ist zu beachten, dass es zwei Möglichkeiten gibt:

a) Die Position bezieht sich auf das „Elternelement“ bzw. dem übergeordneten Element. In unserem Beispiel wäre es die „Form“. (PosrelativetoParent)

b) Die Position bezieht sich auf ein anderen Element „Anchor“. Das andere Element muss auch angegeben werden (in „anchor“), dann kann die Position relativ dazu gesetzt werden (PosRelativeToAnchor). Wichtig hierbei zu beachten ist, dass alles ausgewählt werden kann aber unüberlegt kann es nur zum Chaos führen.
Es muss immer eine horizontale und eine vertikale Position zum Elternelement und/oder zum Anchor geben. Und zwar logisch sinnvoll.

Lazarus selbst richtet die Elemente entsprechend der Angabe aus, dann sieht es aus wie auf Android.
! Gibt man aber keine oder eine falsche Positionsangabe an, dann setzt Lazarus die Elemente absolut so auf die Form wie man sie darauf gezogen hat – auch wenn das „Falsch ist“ und bei Android anders interpretiert wird. Hier muss man selbst aufpassen! Lazarus gibt hier keine Warnung o.ä. heraus. Am besten probiert man sich in einer Testanwendung etwas aus um ein Gefühl dafür zu bekommen.
Anfangs ist es erstmal ungewohnt, dass es keine Absolute Positionsangaben gibt wie bei den üblichen Anwendungen unter Lazarus. Mit den relative Positionsangabe ist es möglich die Anwendung auf allen

Smartphonegrößen „schön“ darzustellen. Die unterschiedlichen Displaygrößen auf Smartphones sollte beachtet werden.

Als Merkhilfe:
Bei der Positionsangabe müssen 2 Häkchen gemacht werden! Fehlt eines, dann ist es unvollständig.

Dazu gibt es im Objektinspektor zwei Punkte. Zur Vereinfachung mit deutscher Übersetzung:

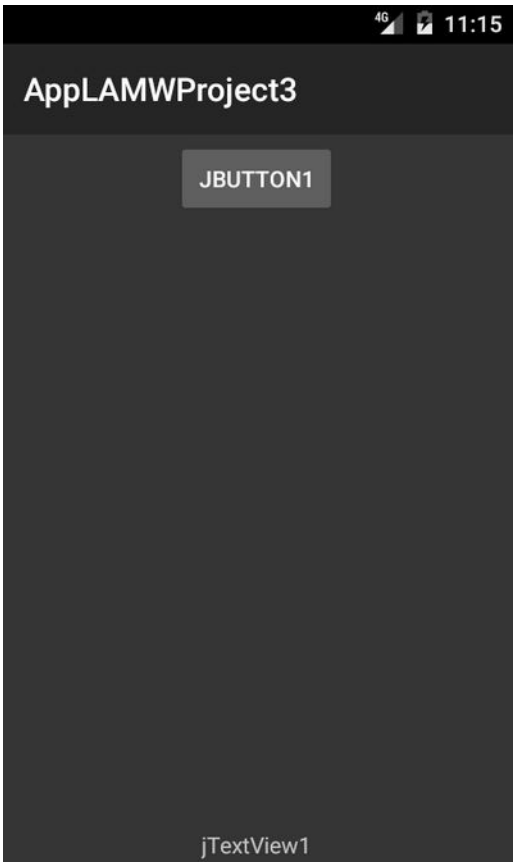
```
Anchor              Fixpunkt / Anker
PosRelativeToAnchor Position relativ zum Fixpunkt
     Above          Darüber, oberhalb
     AlignBaseline  Ausrichten Grundlinie
     AlignBottom    Ausrichten Unten
     AlignEnd       Ausrichten am Schluss, Ziel
     AlignLeft      Ausrichten Links
     AlignRight     Ausrichten Rechts
     AlignStart     Ausrichten am Anfang
     AlignTop       Ausrichten oben
     Below          Unten
     ToEndOf        zu ende von
     ToLeftOf       links von
     ToRightOf      rechts von
     ToStartOf      am anfang..

PosRelativeToParent Position relativ
     rpBottom       Unten
     CenterHorizontal  Mitte Horizontal
     CenterinParent   Mitte Eltern
     CenterVertical   Mitte Vertikal
     End            ende
     Left           Links
     Right          Recjts
     Start          Anfang
     Top            oben
```

Ein weiteres Beispiel zur Positionierung:

Auf einem leeren Form zwei Elemente legen: Ein Button, ein Textfeld.

Der Button soll oben mittig positioniert werden, dass Textfeld unten mittig.

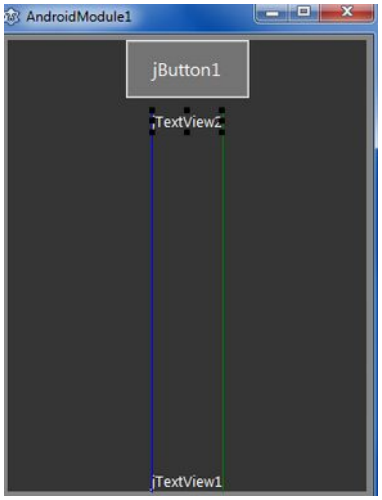
Die Eingabe lautet wie folgt:
- Beim Button jbutton1: PosRelativeToParrent → rpCenterHorizontal + rpTop
- Beim Textfeld jtextview1: PosRelativeToParent → rpCenterHoriztonal + rpBottom



Um ein weiteres Textfeld mittig unter dem Button zu positionieren: Erst ein Textfeld auf die Form ziehen und dann folgende Angabe machen im Objektinspektor:

Textfeld jtextview2:
Anchor: jButton1 !
PosRelativeToAnchor → raBelow
PosRelativeToParent → rpCenterHorizontal

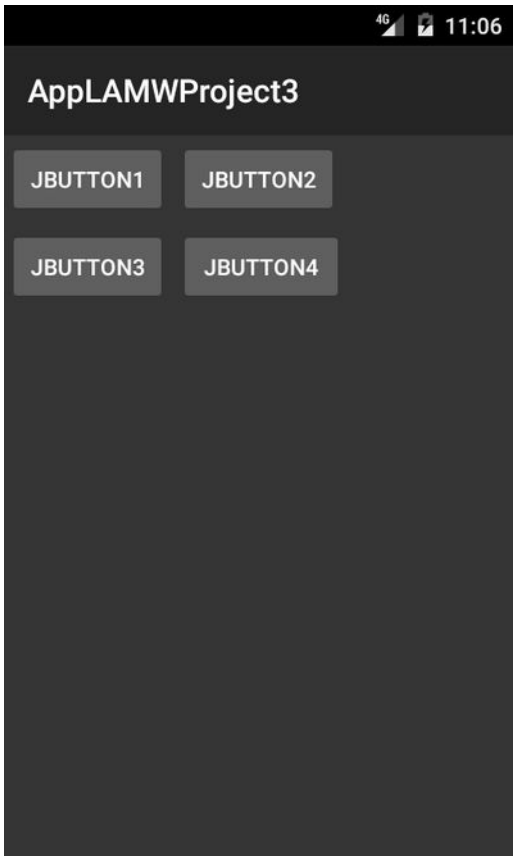
Dann sieht es so aus:



Bevor man also eine Android Anwendung schreibt, sollte man vorab schon eine Vorstellung haben wie die Anwendung aussehen soll. Welche Texte, Buttons und Elemente soll die App haben und welche positionelle Zusammenhänge soll es geben?
Diese „lästige“ Arbeit kann einen später sonst viel Zeit und nerven kosten.

Eine weitere Idee ist, sich eine eigene Routine schreiben für die Positionierung der Elemente (siehe auch nächstes Beispiel!).

Ein drittes Beispiel, Position und Margin (Element-Abstand)



(Möchtest du erstmal selbst versuchen die Elemente so wie in der Grafik anzuordnen?)

Um gezeigtes Beispiel so anzuordnen:

jButton1:
Anchor: -
PosRelativToParrent: rpLeft, rpTop

jButton2:
Anchor: jButton1
PosRelativToParent: raToEndof
PosRelativToAnchor: rpTop

jButton3:

Anchor: jButton1
PosRelativToParent: rpLeft
PosRelativToAnchor: raBelow

jButton4:
Anchor: jButton2
PosRelativToAnchor: raBelow, raAlignLeft
MarginLeft: 0

Margin ist der Abstand zu einem Element. Standardmäßig ist er bei 5. In diesem Fall, möchte man die vier Buttons Tabellenförmig anordnen, muss der linke Abstand herausgenommen werden.

Mithilfe von *Margin*Left, Right usw. kann man auch Elemente auch genauer positionieren. Macht man dies nicht im Objektinspektor sondern im Quelltext, muss auch ein „Update“ erfolgen damit das Element neu positioniert wird z.B.:

```
jTextView1.MarginTop  :=  100;
jTextView1.MarginLeft :=  20;
jTextView1.UpdateLayout;
```

Fehlt das .UpdateLayout wird das Element nicht oder nicht Neu-Positioniert.

Kleines Schlusswort

Android ist ein anderes System als Windows. Deshalb müssen die Android spezifischen Sachen beachtet werden u.a. Zeichensätze usw.

Ein paar Fehlermeldungen...

Fehlermeldung: Kann gtk2 nicht finden verwendet von […] von Package LCL
Man hat eine nicht-Android Komponente/Unit/Datei/… zu erstellen und dem Projekt hinzugefügt. Das kann nicht klappen mit Android.
Lösung: Die Komponente muss wieder vollständig entfernt werden:
1) Die Komponente auf der Form entfernen 2) Im Projektinspektor auch die zusätzlichen Dateien entfernen z.B. LCL (oder was auch hinzugefügt wurde!) 3) Im Uses Teil
Danach sollte alles wieder laufen.