Near Field Communication mit Android

Ganz nah dran

Mit „Gingerbread" (Android Version 2.3) unterstützt Google nun auch Near Field Communication und verleiht damit dem Mobile Payment und Mobile Ticketing einen neuen Schub. Dieser Artikel stellt NFC vor und zeigt, wie Sie die Technik mit Android nutzen. *Dominik Gruntz*

NFC ist eine kontaktlose Schnittstellentechnologie, die die einfache und schnelle Kommunikation über rund 2cm zwischen RFID-Tags und einem speziell ausgerüsteten Mobiltelefon ermöglicht. Die Einsatzgebiete von NFC sind vielfältig: Daten, die auf einem NFC-Tag abgespeichert sind, können durch Berührung mit einem Mobiltelefon ausgelesen werden. Bei den ausgelesenen Daten kann es sich um eine URL, eine Telefonnummer, ein Bild, eine Geo-Koordinate oder eine Visitenkarte (vCard) handeln. Die Daten lassen sich dann direkt auf dem Gerät weiterverarbeiten, ein mühsames Abtippen oder Fotografieren von QR-Tags entfällt. Per NFC rufen Sie zum Beispiel allein über die Annäherung mit dem Mobiltelefon die Bedienungsanleitung für einen Geschirrspüler auf. Tags an den Regalen im Supermarkt informieren über Inhalte von Lebensmitteln, und mithilfe der ausgelesenen Produktnummer können nützliche Zusatzinformationen aus unabhängigen Quellen angezeigt werden, zum Beispiel eine persönliche Allergiewarnung. Auch das Selfscanning (wie zum Bei-

Kirsty Pargeter, 123RF

spiel passabene der schweizer Handelskette Coop [3]) ließe sich damit realisieren. NFC-Mobiltelefone wirken damit als Browser für das Internet der Dinge [1].

Die Anwendung von NFC ist vergleichbar mit dem Einlesen von 2D-Barcodes, außer dass die Handhabung viel einfacher ist. Es muss nicht speziell eine Applikation wie der Barcode-Scanner gestartet werden, und die Kamera muss auch nicht speziell auf ein Tag ausgerichtet werden. Es genügt, den NFC-Tag mit dem Mobiltelefon zu berühren. Dieses erkennt den Tag automatisch und startet eine Applikation zur Verarbeitung der übertragenen Daten. Mit einem NFC-Mobiltelefon kann im Prinzip auch ein Tag emuliert werden, welches von einem Lesegerät über die NFC-Schnittstelle ausgelesen oder beschrieben werden kann. Häufig handelt es sich dabei um Tags mit erweiterter Funktionalität, sogenannte Smartcards. Damit kann kontaktloses Bezahlen realisiert werden, wie es heute in vielen Ländern bereits in Form kontaktloser Kreditkarten genutzt wird.

Im Rahmen des Projekts touch'n'pay [7], an dem die Fachhochschule Nordwestschweiz mitgearbeitet hat, wurde zum Beispiel ein Hofladen so ausgerüstet, dass das Einkaufen mit NFC-fähigen Mobiltelefonen möglich wird. Die Regale erhielten dazu NFC-Produktetiketten. Wenn der Kunde sein Mobiltelefon an ein solches Produktetikett hält, wird das Produkt automatisch auf seinem elektronischen Kassenzettel auf dem Mobiltelefon eingetragen. Sobald alle gewünschten Produkte in der Einkaufstasche liegen, muss nur noch ein Checkout-Tag berührt oder das Zahlen-Menü auf dem Mobiltelefon gewählt werden, um den Bezahlvorgang auszulösen. Das Mobiltelefon wurde außerdem dazu verwendet, um auch außerhalb der Öffnungszeiten den Zugang zum Hofladen zu ermöglichen. Das Türschloss konnte dabei über NFC auf im Telefon abgespeicherte Daten zugreifen [2].

Technologie

Einen guten Überblick über die NFC-Technologie findet man im kürzlich erschienenen Buch von Josef Langer und Michael Roland [4]. Dieser Artikel geht deshalb nur auf die wichtigsten Aspekte ein. NFC unterscheidet folgende Betriebsarten:

- Reader/Writer-Modus: In dieser Betriebsart wird das NFC-Mobiltelefon zum Leser und kann passive NFC-Tags auslesen und mit Daten beschreiben.
- Card-Emulation-Modus: In dieser Betriebsart ist das NFC-Mobiltelefon passiv und emuliert ein Tag, typischerweise eine Smartcard. Ein RFID-Leser, zum Beispiel ein Kassensystem oder ein Türschloss, greift auf das im NFC-Mobiltelefon emulierte Smartcard-Tag zu.
- Peer-to-peer-Modus: Die Peer-to-peer-Betriebsart ermöglicht es, Informationen zwischen zwei (aktiven) Geräten auszutauschen. Es kann sich dabei um einen Gutschein handeln, oder um gegenseitige Identifikationen, damit größere Datenmengen danach einfach über Bluetooth oder über das Internet ausgetauscht werden können.

Wie die Smartcard enthält ein NFC-Gerät auch ein sogenanntes Secure-Element (SE), auf welchem Daten und Programme gesi-

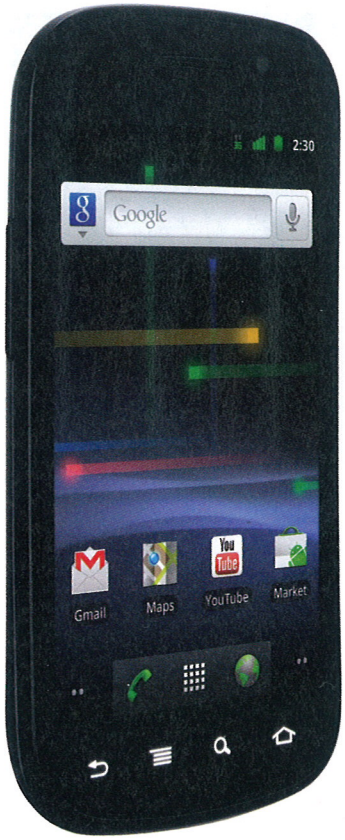

Abb. 1: Das Nexus S von Google ist das erste Android-Smartphone mit Gingerbread und einem NFC-Chip.

TABELLE 1: Mögliche TNF-Werte

TNF_EMPTY = 0	Dieses Format zeigt an, dass der Record leer ist, das heißt, er hat weder Typ, Kennung noch Nutzdaten. Die entsprechenden Getter-Methoden auf dem Record geben leere Arrays zurück.
TNF_WELL_KNOWN = 1	Dieses Format zeig an, dass der Typ als NFC Forum Well-known Type codiert ist (entsprechend der Record Type Definition). Beispiel: *T* steht für einen Text-Record, *U* für eine URI oder *Sp* für einen Smart-Poster-Record. Die wichtigsten Typen sind auch in den Konstanten *NdefRecord.RTD_TEXT*, *RTD_URI* und *RTD_SMART_POSTER* abgelegt.
TNF_MIME_MEDIA = 2	Records mit diesem TNF enthalten eine Typ-String im MIME Format nach RFC 2046, zum Beispiel *"image/png"* für ein Bild oder *"text/x-vCard"* für eine Visitenkarte.
TNF_ABSOLUTE_URI = 3	Wenn diese Konstante gesetzt ist, dann enthält das Typfeld eine absolute URI nach RFC 3986. Dieser URI definiert dann das Format der Nutzdaten.
TNF_EXTERNAL_TYPE = 4	Dieses Format gibt an, dass das Typfeld ein NFC Forum External Type ist. Firmen können damit anwendungsspezifische Record-Typen definieren. Ein Beispiel eines solchen Typs ist *urn:nfc:ext:fhnw.ch:shop*, wobei im Typ-Feld nur der relative URN *fhnw.ch:shop* eingetragen wird.
TNF_UNKNOWN = 5	Wenn dieses Flag gesetzt ist, dann enthält der Record Daten in einem unbekannten Format. Der Typ-String ist in diesem Fall leer.
TNF_UNCHANGED = 6	Ein TNF mit Wert 6 gibt an, dass die Nutzdaten auf mehrere Records verteilt sind (analog zu chunked encoding von HTTP-Payloads). Der Typ der Nutzdaten wird auf dem ersten Record spezifiziert, auf den nachfolgenden Records ist *TNF=6* gesetzt und das Typfeld ist leer.

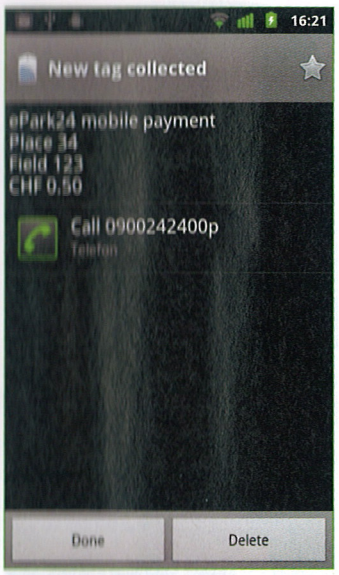

Abb. 2: Android hat einen NFC-Tag erkannt (Parkgebühr auf einem Parkplatz) und zeigt einen passenden Hinweis an.

chert abgelegt werden können. Das SE ist entweder im Gerät fest verbaut, oder es ist Teil der SIM-Karte des Mobilfunkanbieters. Bei einem Wechsel des Gerätes ist es von Vorteil, wenn die Daten und Programme mit der SIM-Karte auf das neue Gerät übernommen werden können.

Die Hardware

Android unterstützt in der aktuellen Version 2.3 „Gingerbread" nur den Reader-Modus, das heißt Daten, die auf einem NFC-Tag abgespeichert sind, können ausgelesen werden. Das Schreiben von Tags, die Emulation von Karten oder der Austausch von Daten zwischen zwei NFC-Geräten ist über das bereitgestellte API jedoch noch nicht möglich.

Das erste Smartphone, das Gingerbread offiziell unterstützt, ist das Nexus S von Samsung. In diesem Mobiltelefon ist der PN544 NFC Controller von NXP eingebaut [5]. Dieser Controller unterstützt alle Betriebsarten, es ist also nur eine Frage der Software, alle Funktionen über das Android-API anzusprechen. Das Schreiben von Tags ist bereits jetzt mit ein paar Tricks und mithilfe von Reflection möglich, das heißt, diese Funktionalität ist in den Bibliotheken vorhanden. Knackpunkt ist aktuell der Zugriff auf das Secure-Element. Es ist zurzeit (Stand Januar 2011) nicht klar, ob dieses im Smartphone integriert ist (SmartMX Security Chip) oder ob via UICC (Universal Integrated Circuit Card, eine erweiterte SIM-Karte) auf ein SE auf der SIM-Karte zugegriffen wird. In beiden Fällen ist für das Speichern von Programmen jedoch ein Schlüssel nötig. Die Frage, wie dieser an die Entwickler verteilt wird, bleibt hingegen unbeantwortet.

LISTING 1: android.nfc.action

```
<intent-filter>
   <action android:name="android.nfc.action.TAG_DISCOVERED"/>
   <category android:name="android.intent.category.DEFAULT"/>
</intent-filter>
```

LISTING 2: Einlesen der Tag-Daten

```
NdefMessage[] readMessages(Intent intent) {
   String action = intent.getAction();
   if( !NfcAdapter.ACTION_TAG_DISCOVERED.equals(action))
      throw new RuntimeException("unknown intent");
   Parcelable[] rawMsgs =
      intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);
   NdefMessage[] msgs;
   if (rawMsgs != null && rawMsgs.length > 0) {
      msgs = new NdefMessage[rawMsgs.length];
      for (int i = 0; i < rawMsgs.length; i++) {
         msgs[i] = (NdefMessage) rawMsgs[i];
      }
   } else { // No message read => return unknown tag type
     byte[] empty = new byte[] {};
     NdefRecord record = new NdefRecord(NdefRecord.TNF_UNKNOWN,
                                 empty, empty, empty);
     NdefMessage msg = new NdefMessage(new NdefRecord[] {record});
     msgs = new NdefMessage[] {msg};
   }
   return msgs;
}
```

Die Software

Aktuell ist mit dem Nexus S lediglich das Auslesen von NFC-Tags möglich. Die NFC Data Exchange Format (NDEF) Spezifikation [6] legt die Formate für den Austausch von Informationen zwischen NFC-Geräten und NFC-Tags fest. Anwendungsdaten sind (zusammen mit Metainformationen) in einem oder mehreren NDEF-Einträgen abgelegt. Ein oder mehrere NDEF-Einträge bilden eine NDEF-Meldung. Die NDEF-Einträge ermöglichen die Repräsentation von Datenpaketen in diversen Datenformaten. Ein Tag enthält in einer NDEF-Meldung zum Beispiel eine URI, einen Text und ein Icon. Wie Daten in den Einträgen repräsentiert und auf den Geräten interpretiert werden, legt die RTD-Spezifikation (NFC Record Type Definition) fest.

In Android sind die nötigen Datentypen im Paket `andorid.nfc` definiert. Die Klasse `NdefMessage` repräsentiert eine NDEF-Meldung, welche einen oder mehrere NDEF-Einträge enthält. In diesen Einträgen sind die Nutzdaten abgelegt. Ein NDEF-Eintrag besteht aus einem Header und einem Datenteil. Im Header sind neben Flags Informationen zum Typ der Nutzdaten, die Länge der Nutzdaten sowie optional eine eindeutige Kennung des Eintrags abgelegt. Android bildet einen NDEF-Eintrag mit der Klasse `NdefRecord` ab.

Für die Beschreibung des Typs eines Records sind verschiedene Formate vorgesehen. Der Wert des Feldes Type Name Format (TNF) definiert das Format des Typ-Strings. Welche TNF-Werte in der NDEF-Spezifikation (und als Konstanten in der Klasse `NdefRecord`) definiert sind, zeigt Tabelle 1.

Neben den beiden Klassen `NdefMessage` und `NdefRecord` enthält das Paket `android.nfc` auch noch die Klasse `NfcAdapter`. Mit der Methode `NfcAdapter.getDefaultAdapter` greift eine App auf den NFC-Controller zu. Aktuell lässt sich auf dem Controller nur mit der Methode `isEnabled` prüfen, ob Near Field Communication in den Einstellungen für Drahtlosnetzwerke aktiviert ist.

NFC-Tags lesen

Sobald Android ein NFC-Tag erkennt, löst es einen entsprechenden Intent aus, der die eingelesenen Daten enthält. Eine Aktivität, die den passenden Intent-Filter definiert hat, startet dann automatisch. Falls mehrere Applikationen auf NFC-Tags reagieren können, so muss der Benutzer wählen, welche Aktivität er verwenden will (zu den Intents im Detail siehe auch den Artikel auf Seite 60). In Listing 1 ist angegeben, welchen Filter eine Aktivität deklarieren muss, um auf NFC-Tags reagieren zu können.

Auf dem Intent, den die Activity erhält, kann die NDEF-Meldung im Binärformat ausgelesen werden. Die Daten, die auf dem Tag gespeichert sind, werden als Byte-Arrays ausgelesen und lassen sich in entsprechende Instanzen des Typs `NdefMessage` konvertieren, wie Listing 2 zeigt.

NDEF-Meldungen verarbeiten

Leider bietet Android 2.3 zurzeit keine weiteren Klassen an, mit denen sich NDEF-Einträge verarbeiten lassen. Sie müssen die Daten somit von Hand parsen.

Betrachten wir als Beispiel das Auslesen eines Text-Eintrags, also eines Eintrags mit `TNF = "Well-Known"` und `Typ = "T"`. Das Format dieser Einträge ist in der Text Record Type Definition des NFC-Forums spezifiziert. Neben dem eigentlichen Text enthält ein solcher Record noch Metainformationen bezüglich der Zeichencodierung und der Sprache des Textes. Mit der Sprachangabe ist es möglich, einen Text (in separaten NDEF-Einträgen) in mehreren Sprachen bereitzustellen. Damit unterscheidet sich dieser NDEF-Datentyp „Text“ vom MIME Typ `"text/plain"`.

Der Payload einer Textmeldung startet mit einem Statusbyte. Bit 7 des Statusbytes gibt die Zeichencodierung des Textes an. 0 steht für UTF-8 und 1 steht für UTF-16. Bit 6 ist für zukünftige Erweiterungen reserviert und Bit 0 bis 5 geben die Länge des Sprachcodes an. Listing 3 zeigt, wie eine App die Nutzdaten aus einem NDEF-Eintrag vom Typ `RTD_TEXT` ausliest. Das Resultat legen Sie dann in einer Instanz der Klasse `TextRecord` ab. Beachten Sie, dass die Methode `parse` den Fall, dass der Text auf mehrere NDEF-Einträge verteilt ist (chunked encoding), nicht berücksichtigt.

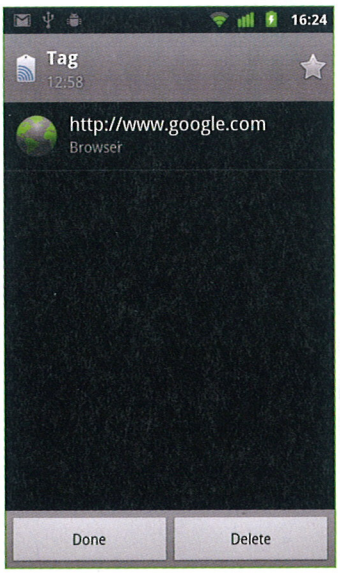

Abb. 3: Die Tag-Verwaltung zeigt die bislang „aufgesammelten“ NFC-Tags wie eine Bookmark-Liste an.

LISTING 3: Extraktion eines Text-Eintrags aus einem NDEF-Eintrag

```
public class TextRecord {
   private final String languageCode;
   private final String text;

   private TextRecord(String languageCode, String text) {
      if(languageCode == null || text == null)
         throw new IllegalArgumentException();
      this.languageCode = languageCode;
      this.text = text;
   }

   public String getText() { return this.text; }
   public String getLanguageCode() { return languageCode; }

   public static TextRecord parse(NdefRecord record) {
      if (record.getTnf() != NdefRecord.TNF_WELL_KNOWN
      || ! Arrays.equals(record.getType(), NdefRecord.RTD_TEXT))
         throw new IllegalArgumentException();
      try {
         byte[] payload = record.getPayload();
         byte status = payload[0];
         String enc = ((status & 0x80) == 0) ? "UTF-8" : "UTF-16";
         int len = status & 0x3f;

         String languageCode = new String(payload, 1, len, "US-ASCII");
         String text = new String(payload, len+1, payload.length-len-1, enc);
         return new TextRecord(languageCode, text);
      } catch (UnsupportedEncodingException e) { // malformed tag.
         throw new IllegalArgumentException(e);
      }
   }
}
```

Erste Anwendungen

Das Nexus S bringt zur Verwaltung von NFC-Tags eine gesonderte Tags-Applikation mit. Halten Sie das Gerät an einen Sensor und erkennt es den Tag (vorausgesetzt, NFC ist in den Einstellungen für Drahtlosnetzwerke aktiviert), dann erscheint der Inhalt des Tags auf dem Bildschirm (Abbildung 2) und der Benutzer kann per Fingerklick eine Aktion ausführen. Die eingelesenen Tags verwaltet Android analog zu einer Bookmark-Liste im Browser (Abbildung 3).

Damit der Android-Market eine App nur dann anzeigt, wenn das Gerät NFC unterstützt, muss das Manifest den folgenden `uses-feature`-Abschnitt enthalten:

```
<uses-feature android:name="android.hardware.
nfc" android:required="true">
```

Im Android-Market sind bereits erste Applikationen enthalten die NFC verwenden (eine davon ist allerdings nur in den USA verfügbar). Taglet ist eine Applikation die es erlaubt, im Internet Informationen zu einem Tag zu hinterlegen. Diese Information zeigt die App dann an, wenn man das Tag mit einem anderen Gerät ausliest. Auf diese Weise lassen sich Tags annotieren. Eine zweite Anwendung ist Enable Table. Diese Applikation erlaubt es, Besuchern eines Restaurants Rabatt-Gutscheine abzugeben. Aus der Beschreibung im Market und im Internet geht jedoch nicht eindeutig hervor, wie diese Gutscheine dann eingelöst werden können.

Raum für Verbesserung

Die im Paket `android.nfc` definierten Klassen sind sehr spartanisch gehalten. Eigene Klassen für die in den NDEF- und RTD-Spezifikationen definierten Typen wären wünschenswert. Es ist nicht einsehbar, warum jeder Programmierer die Tags erneut parsen soll. Aber auch in diesem Punkt darf damit gerechnet werden, dass Android in einer zukünftigen Version nachbessert.

Wird ein Tag erkannt, so zeigt Android alle Applikationen an, die generell auf Tags reagieren. Auf dem für diesen Artikel benutzten Nexus S sind das neben der Tags-Applikation die Taglet-Aktivität und die Touch'n'pay-Aktivität. Es wäre sinnvoll, im Intent-Filter den Datentyp zu spezifizieren, den eine Aktivität versteht. Erkennt Android ein Tag vom Typ `urn:nfc:ext:fhnw.ch:shop`, dann sollte es sofort die Shop-Anwendung öffnen, ohne den Benutzer zu fragen, mit welcher Aktivität er den eingelesenen Tag behandeln möchte.

Neben diesen Mängeln stehen auch noch viele Fragen offen, insbesondere im Kontext des Secure-Elements. Es ist nicht klar, wo genau das SE abgelegt ist (SIM Karte, Chip oder beides) und ob beziehungsweise wie man als Programmierer auf das SE zugreifen kann und die dazu nötigen kryptographischen Schlüssel erhält. Vermutlich sind diese offenen Fragen mit ein Grund dafür, dass das API vorerst nur einen Teil der Funktionalität des NFC-Controllers anbietet.

Fazit

Die NFC-APIs und das Nexus S sind ein erster Schritt in Richtung Mobile Payment und Mobile Ticketing. Mit der vorliegenden Android-Version lässt sich diese Vision jedoch noch nicht realisieren, da wichtige Funktionen fehlen. Der im Nexus S eingebaute NFC-Controller unterstützt jedoch die volle Funktionalität, daher ist es nur eine Frage der Zeit, bis Google auch die Bibliotheken auffrischt.

NFC ist eine Technologie, deren Verbreitung und Akzeptanz durch die Initiative von Google nun rasch zunehmen dürfte. Ob 2011 wirklich als „das NFC-Jahr" in die Geschichte eingehen wird, hängt neben Adroid, Google und den Smartphone-Herstellern vermutlich auch davon ab, ob das für den Sommer erwartete iPhone 5 tatsächlich einen NFC-Chip mitbringen wird. (mhi) •••

INFOS

[1] Friedemann Mattern, Christian Flörkemeier, Vom Internet der Computer zum Internet der Dinge, Informatik Spektrum 33 (2) 2010, S. 107-121.

[2] Ingo Bauersachs, Dominik Gruntz; touch'n pay: ein NFC-Feldversuch, IMVS Fokus-Report, p. 37-42: *http://www.fhnw.ch/technik/imvs/publikationen/artikel-2010/touchn-pay-ein-nfc-feldversuch*

[3] Coop Supermarkt, passabene für das praktische Einkaufen: *http://tinyurl.com/passabene*

[4] Josef Langer, Michael Roland, Anwendungen und Technik von Near Field Communication (NFC), Springer Verlag, Sept 2010.

[5] NXP NFC Controller PN544: *http://www.nxp.com/acrobat_download2/literature/9397/75016890.pdf*

[6] NFC Forum, NFC Data Exchange Format (NDEF), Rev. 1.0. Technical Specification, Jul. 2006: *http://www.nfc-forum.org/specs/spec_list/*

[7] Touch'n pay: *http://www.touchnpay.ch*