Laboratorio

Android: NFC

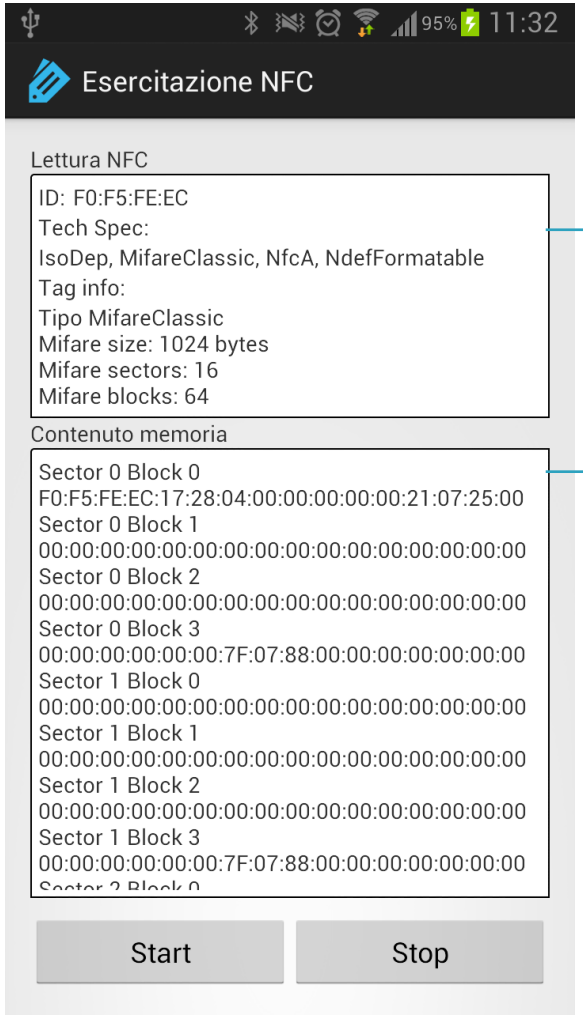
Corso di Smart City e Tecnologie Mobili

Università di Bologna

Dipartimento di Informatica — Scienza e Ingegneria

Luca Calderoni, Andrea Cirri, Dario Maio

Obiettivi

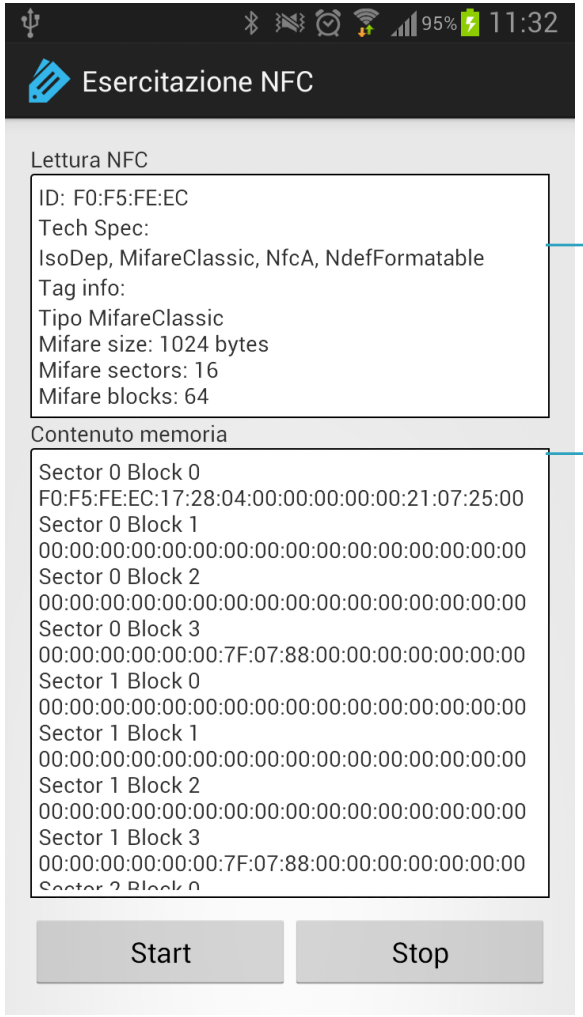


Questo box deve indicare il codice identificativo del Tag, oltre a una lista di tecnologie che lo caratterizzano tra quelle supportate dal sistema Android.

Deve inoltre riportare alcune informazioni di base sul tipo di Tag e la sua memoria (scomposizione in blocchi e settori)

Questo box deve mostrare il contenuto effettivo della memoria letta dal Tag. Il contenuto deve essere opportunamente diviso in blocchi e settori.

Aggiornamento dell'interfaccia




```
public void printId(byte[] data);
public void printTechList(String[] data);
public void printTagInfo(Tag tag);
```

```
public void printMemContent(final String data);
```

Per visualizzare a video i valori inferiti dal sensore si possono utilizzare i metodi sopra citati, già implementati nel progetto scheletro.
La stringa del metodo *printMemContent* deve essere già opportunamente formattata per la visualizzazione. Il metodo si limita a visualizzarla a video.

Near Field Communication

- **Near Field Communication** (NFC) è una tecnologia che fornisce connettività wireless (a radio frequenze) bidirezionale a corto raggio (fino a un massimo di 10 cm).
- Questa tecnologia è strettamente correlata alla meno recente tecnologia **Radio Frequency Identification** (RFID).
- Contrariamente ai più semplici dispositivi RFID, NFC permette una comunicazione bidirezionale: quando due apparecchi NFC (Initiator e Target) vengono avvicinati entro un raggio di 4 cm, viene creata una rete peer-to-peer tra i due ed entrambi possono inviare e ricevere informazioni.
- La tecnologia NFC opera alla frequenza di 13,56 MHz e può raggiungere una velocità di trasmissione massima di 424 Kb/s.

Radio Frequency Identification

Con **Radio Frequency Identification** (RFID) si intende l'utilizzo di segnale radio atto al trasferimento di dati per identificare automaticamente, ed eventualmente tracciare, alcuni piccoli ricevitori (tipicamente passivi) detti **tag**.

I tag contengono almeno un'antenna e un circuito integrato e possono essere dotati di componenti più complessi (co-processore per il calcolo crittografico, batteria, ecc).

Si distinguono varie tipologie di tag:

- Passivi: attivati unicamente dal segnale ricevuto dal reader
- Semi-passivi: dotati di una piccola batteria a supporto del segnale ricevuto dal reader, vengono comunque attivati dal reader
- Attivi: dotati di batteria, emettono periodicamente il segnale in autonomia, a prescindere dalla presenza del reader.

Due possibili modelli di comunicazione:

- Reader – Reader
- Reader – Tag

Radio Frequency Identification

I Reader possono essere attivi o passivi e lavorano a diverse frequenze:

- LF: 120 kHz – 150 kHz (10 cm)
- HF: 13,56 MHz (10 cm – 1 m) (standard utilizzato nelle SmartCard, stessa frequenza di NFC)
- UHF: 433 MHz (1 m – 100 m), 865 - 868 MHz, 902 - 928 MHz (1 m – 12 m)
- Microonde: 2450 MHz – 5800 MHz (1 m – 2 m), 3,1 GHz – 10 GHz (fino a 200 m)

Si distinguono tre diverse architetture applicate ai sistemi RFID:

- ARPT: Reader attivo e tag passivo, la comunicazione avviene sempre nel senso Reader → Tag.
- PRAT: Reader passivo e tag attivo, il reader è tipicamente fisso e funge da ricevitore dei segnali periodicamente inviati dai tag; la comunicazione avviene sempre nel senso Tag → Reader.
- ARAT: Sia il reader sia il tag sono attivi; è comunque il reader che tipicamente interroga il tag per ricevere il segnale in risposta anche se la comunicazione può tecnicamente avvenire in entrambi i sensi.

NFC e RFID

- I dispositivi NFC possono comunicare fra loro, oppure essere usati come lettori di Tag RFID.
- Operando a 13,56 MHz, i sistemi basati su lettori NFC e Tag RFID si collocano tipicamente nella categoria ARPT con un raggio d'azione limitato a circa 10 cm.
- Poiché molti Smartphone sono oggi dotati di sensore NFC, le applicazioni che utilizzano questa tecnologia (fra le quali quelle che si basano sulla lettura di Tag RFID passivi) stanno riscontrando notevole successo e avranno probabilmente una sempre maggiore diffusione.
- L'ostacolo che ha sempre caratterizzato le applicazioni che coinvolgono gli RFID su larga scala è individuabile nel costo dei reader e nella scomodità di doversi dotare di un apparecchio dedicato per la lettura dei Tag. La presenza nativa di un reader RFID sugli smartphone sta abbattendo questo scoglio. In pochi anni ciascun individuo che viva in una zona moderatamente sviluppata avrà a disposizione un lettore RFID sul proprio telefono e dunque potrà disporne pressoché ovunque e in ogni momento.

Le tecnologie RFID su Android

Il sistema operativo Android supporta nativamente la formattazione NDEF per Tag RFID. Alternativamente è possibile gestire le seguenti tecnologie:

Class	Description
TagTechnology	The interface that all tag technology classes must implement.
NfcA	Provides access to NFC-A (ISO 14443-3A) properties and I/O operations.
NfcB	Provides access to NFC-B (ISO 14443-3B) properties and I/O operations.
NfcF	Provides access to NFC-F (JIS 6319-4) properties and I/O operations.
NfcV	Provides access to NFC-V (ISO 15693) properties and I/O operations.
IsoDep	Provides access to ISO-DEP (ISO 14443-4) properties and I/O operations.
Ndef	Provides access to NDEF data and operations on NFC tags that have been formatted as NDEF.
NdefFormatable	Provides a format operations for tags that may be NDEF formattable.
MifareClassic	Provides access to MIFARE Classic properties and I/O operations, if this Android device supports MIFARE.
MifareUltralight	Provides access to MIFARE Ultralight properties and I/O operations, if this Android device supports MIFARE.

Mifare Classic

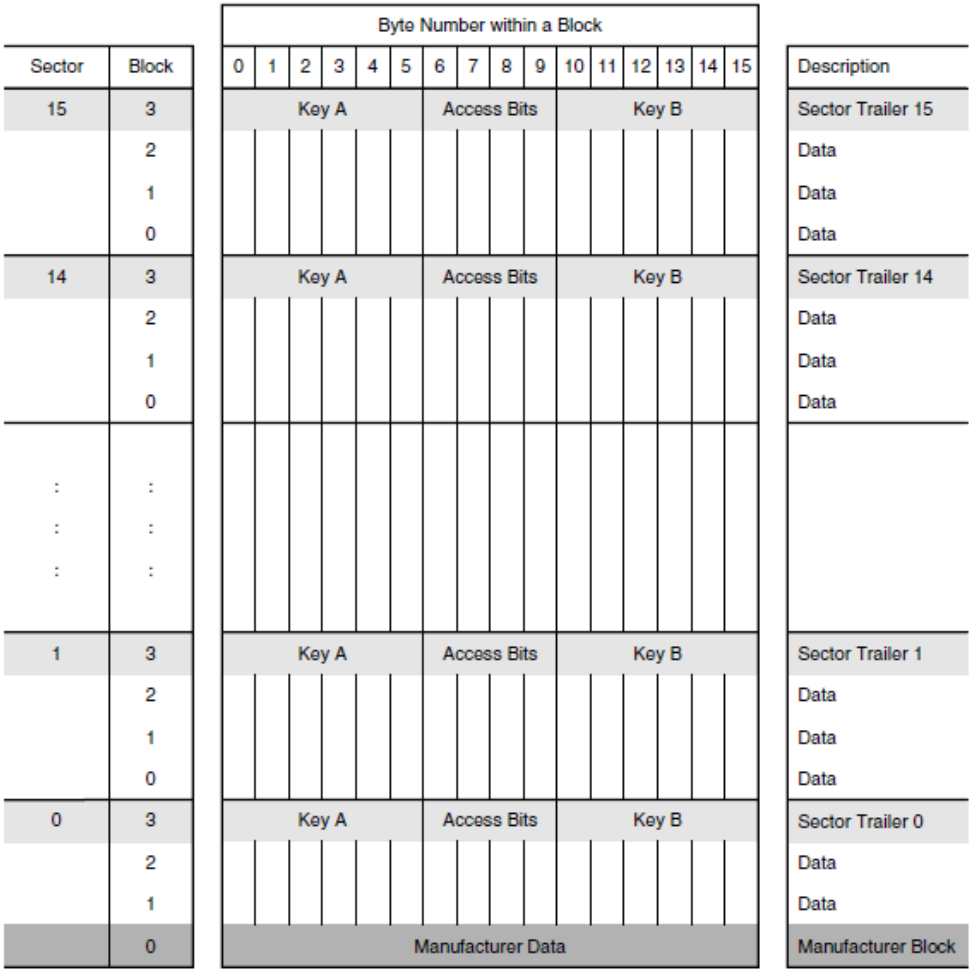
Esistono 4 tipologie di chip Mifare Classic:

- MIFARE Classic Mini: 320 bytes (SIZE_MINI), 5 settori da 4 blocchi ciascuno
- MIFARE Classic 1K: 1024 bytes (SIZE_1K), 16 settori da 4 blocchi ciascuno
- MIFARE Classic 2K: 2048 bytes (SIZE_2K), 32 settori da 4 blocchi ciascuno
- MIFARE Classic 4K: 4096 bytes (SIZE_4K), 32 settori da 4 blocchi ciascuno e 8 settori da 16 blocchi

Ogni blocco contiene 16 byte.
Ogni settore è sottoposto a controllo dell'accesso tramite due chiavi di 6 byte ciascuna salvate nell'ultimo blocco di ogni settore. I 4 byte centrali indicano le politiche di accesso. Il valore di default delle chiavi è FFFFFFFFFFFF. Quando si cerca di leggerle la card restituisce il valore 000000000000.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Key A						Access Bits				Key B (optional)					

Mifare Classic: struttura della memoria



Sector	Block	Byte Number within a Block 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	Description
15	3	Key A | Access Bits | Key B	Sector Trailer 15
	2		Data
	1		Data
	0		Data
14	3	Key A | Access Bits | Key B	Sector Trailer 14
	2		Data
	1		Data
	0		Data
:	:		
:	:		
:	:		
1	3	Key A | Access Bits | Key B	Sector Trailer 1
	2		Data
	1		Data
	0		Data
0	3	Key A | Access Bits | Key B	Sector Trailer 0
	2		Data
	1		Data
	0	Manufacturer Data	Manufacturer Block

Mifare Classic: lettura in Android

Una volta letto un Tag Mifare Classic si può procedere alla lettura della memoria come segue:

```
tagChip.connect();
int sectors = tagChip.getSectorCount();

//Ciclo sui settori
for (int currentSector = 0; currentSector < sectors; currentSector++) {
      boolean auth = tagChip.authenticateSectorWithKeyA(currentSector, MifareClassic.KEY_DEFAULT);
      //Procede alla lettura del settore solo se l'autenticazione è andata a buon fine
      if(auth){
            int blockOfSector = tagChip.getBlockCountInSector(currentSector);
            //Ciclo sui blocchi del settore
            for (int currentBlock = 0; currentBlock < blockOfSector; currentBlock++) {
                  //Lettura settore sec e Blocco i
                  int firstBlockofSector =  tagChip.sectorToBlock(currentSector);
                  tagChip.readBlock(firstBlockofSector + currentBlock);
            }
      }
}

tagChip.close();
```

Mifare UltraLight

Esistono 2 tipologie di chip Mifare UltraLight:

- MIFARE UltraLight: 64 bytes (TYPE_ULTRALIGHT), 16 pagine da 4 byte ciascuna; le prime 4 pagine sono dedicate al produttore e contengono i *locking bits*. Le restanti pagine sono dedicate ai dati.

- MIFARE UltraLight C: 192 bytes (TYPE_ULTRALIGHT_C), 48 pagine da 4 bye ciascuna; le prime 4 pagine sono dedicate al produttore e contengono i locking bits. Le seguenti 36 pagine sono dedicate ai dati. Seguono 4 pagine per locking bits aggiuntivi. Le ultime 4 pagine (illeggibili) contengono la chiave di autenticazione.

Mifare UltraLight: lettura in Android

Una volta letto un Tag Mifare UltraLight si può procedere alla lettura della memoria come segue:

```
mifare.connect();

//Legge le prime 4 pagine (la prima è dedicata al produttore)
//In totale vengono quindi letti 16 byte
//L'indice del metodo readPages indica l'offset. (es. readPages(3) legge dalla quarta all'ottava
//pagina)
byte[] payload = mifare.readPages(0);
mifare.close();
```

Android: NFC intent filters

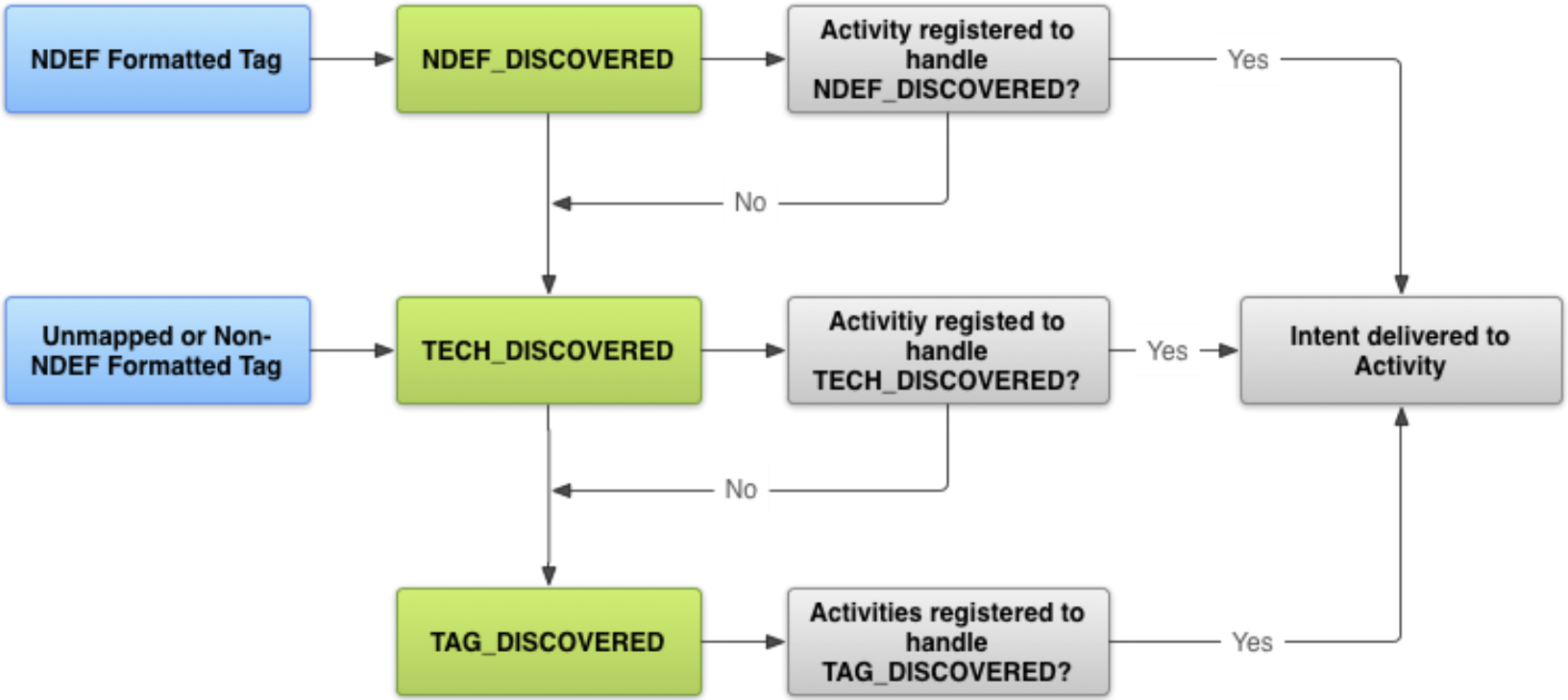
L'applicazione Android deve registrarsi ad almeno uno dei tre eventi che coinvolgono l'NFC; occorre specificare alcuni *intent filters* nel Manifest dell'applicazione.

```
//Viene scatenato un intent a prescindere dal tipo di Tag che entra nel raggio d'azione dello
//SmartPhone
<intent-filter>
    <action android:name="android.nfc.action.TAG_DISCOVERED" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>

//Viene scatenato un intent solo se il Tag appartiene a una delle tecnologie contenute ne file
//nfc_filter
<intent-filter>
    <action android:name="android.nfc.action.TECH_DISCOVERED" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
<meta-data android:name="android.nfc.action.TECH_DISCOVERED" android:resource="@xml/nfc_filter"/>

//Viene scatenato un intent solo se il Tag che entra nel raggio d'azione è formattato NDEF
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
    <category android:name="android.intent.category.DEFAULT"/>
</intent-filter>
```

Android: Discovery di un TAG



NFC Adapter

Per ricevere gli intent scatenati all'avvicinamento di un Tag è necessario istanziare un NFC Adapter. Se lo Smartphone non dispone di sensore NFC verrà notificato.

```
nfcAdapter = NfcAdapter.getDefaultAdapter(this);

//Controlla che lo SmartPhone disponga di sensore NFC
if(nfcAdapter == null){
      new AlertDialog.Builder(this).setTitle("Attenzione").setMessage("Modulo NFC non presente nel
      dispositivo.").setPositiveButton("OK", null).create().show();
      finish();
      return;
}

//Questo è l'intent che viene scatenato all'avvicinamento di un Tag per mettere l'activity in
//foreground se non lo è già (questa procedura va abbinata al seguente comando
//enableForegroundDispatch)
pendingIntent = PendingIntent.getActivity(this, 0, new Intent(this,
getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);
//Se il sensore NFC è disabilitato mostra la schermata dei settings per abilitarlo
if(!nfcAdapter.isEnabled()) {
      showWirelessSettingsDialog();
}

//Registra il pending intent in modo che se l'activity è già in foreground venga inviato a
//quell'activity il solo intent relativo al Tag, senza passare dal meccanismo di risoluzione degli
//intent del sistema Android.
nfcAdapter.enableForegroundDispatch(this, pendingIntent, null, null);
```

Risoluzione degli Intent

Per la risoluzione degli Intent è obbligatorio implementare un ulteriore metodo che si rende necessario per le applicazioni che sono eseguite in modalità singleTop o che vengono risvegliate con un intent del tipo FLAG_ACTIVITY_SINGLE_TOP.

```
@Override
public void onNewIntent(Intent intent) {
    setIntent(intent);
    resolveIntent(intent);
}
```

Il metodo resolveIntent dovrà accertare quale tipo di Tag ha risvegliato l'applicazione. Una volta fatto ciò sarà possibile istanziare un nuovo oggetto Tag e successivamente farne il cast alla opportuna tecnologia (ad esemipo MifareClassic).

```
if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(action)
    || NfcAdapter.ACTION_TECH_DISCOVERED.equals(action)
    || NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)) {
      byte[] id = intent.getByteArrayExtra(NfcAdapter.EXTRA_ID);
      Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
      ...
```

Consigli per l'implementazione

Per visualizzare le porzioni di memoria lette dal Tag, è opportuno utilizzare il formato esadecimale per ottenere una formattazione del tipo:

XX:XX:XX: …

A tale scopo può essere di aiuto la libreria *commons-codec* di Apache.

```
String s = new String(Hex.encodeHex(byte[] data));
```

Quando l'applicazione entra in pausa non è più necessario instradare gli intent direttamente a essa come quando è in foreground.

```
if (nfcAdapter != null) {
      nfcAdapter.disableForegroundDispatch(this);
}
```