



NFC

LA TECNOLOGIA CHE HA CAMBIATO IL MONDO

CONTENUTI

Introduzione

Funzionamento

Utilizzi

Novità

Integrazione Android

Integrazione iOS

Conclusioni



INTRODUZIONE

Near Field Communication è una tecnologia di ricetrasmissione che fornisce connettività senza fili (RF) bidirezionale a distanza (contactless) a corto raggio.

PEER TO PEER

Comunicazione tra due o più dispositivi

Non esiste un server centrale

Scambio diretto di dati

Passivo/Attivo



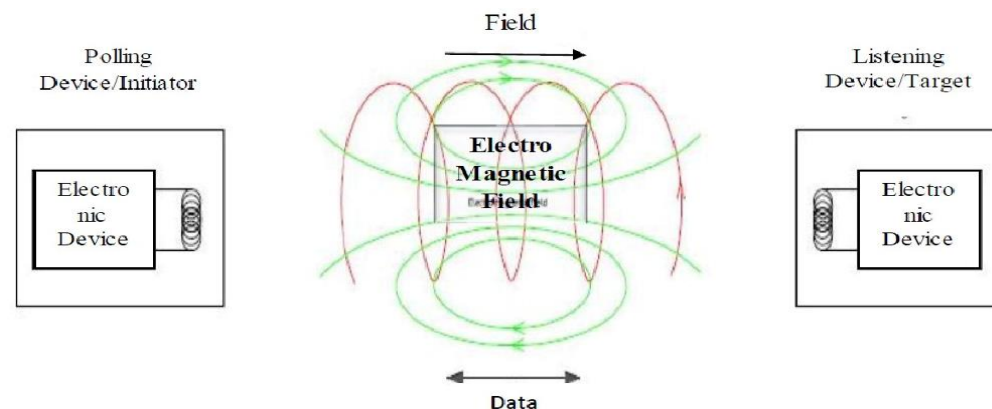
TRASMISSIONE

Onde elettromagnetiche

Comunicazione di prossimità

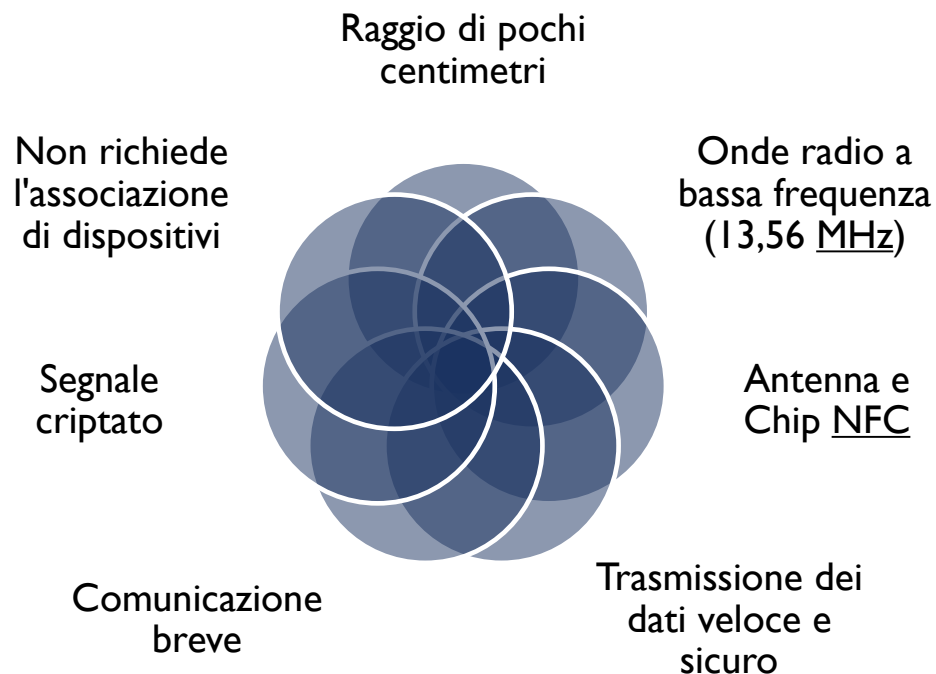
Velocità di 424 Kbit/s

Raggio di pochi centimetri

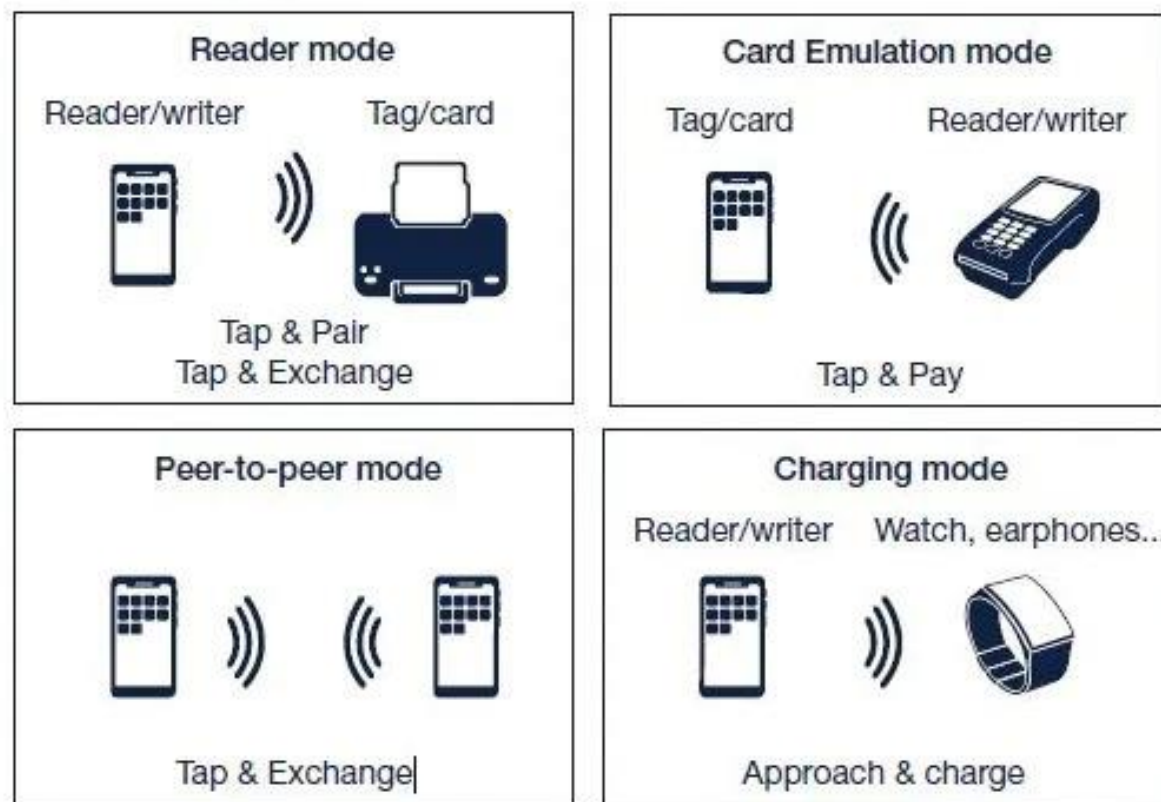


FUNZIONAMENTO

CARATTERISTICHE



TIPOLOGIE



CONFRONTO

NFC, Bluetooth e Wi-Fi Direct sono tre tecnologie di comunicazione wireless che hanno diverse caratteristiche e utilizzi.

BLUETOOTH

Comunicazione con raggio di 10m

Comunicazione fino a 25Mbps

Richiede l'associazione

Comunicazione intercettabile



Wireless devices, streaming rich content, like video and audio

Devices that connect with both. The center of your wireless world

Sensor devices, sending small bits of data, using very little energy

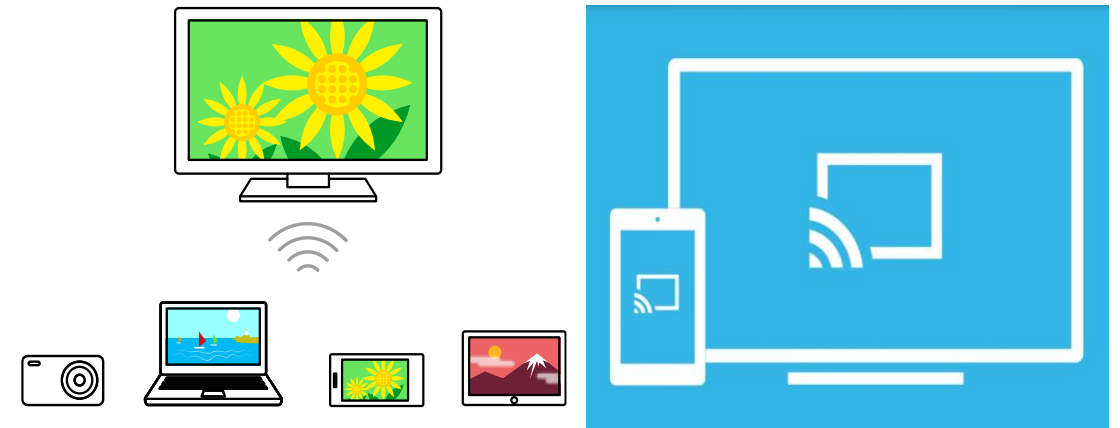
WI-FI DIRECT

Comunicazione con raggio di 100m

Comunicazione fino a 250Mbps

Richiede l'associazione

Comunicazione intercettabile



UTILIZZI



Pagamenti mobili



Condivisione di
dati sicura



Accesso
elettronico



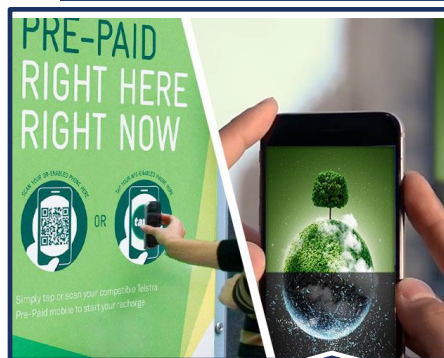
Smart home



Biglietti elettronici



Talking Store



Poster NFC



Gestione
impostazioni



Anti-contraffazione



Pet Tag

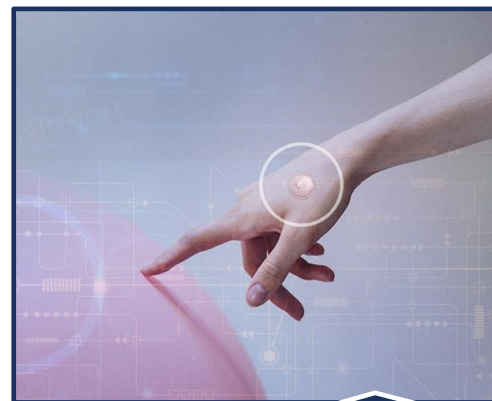
IL FUTURO



Unghie



Vestiti intelligenti



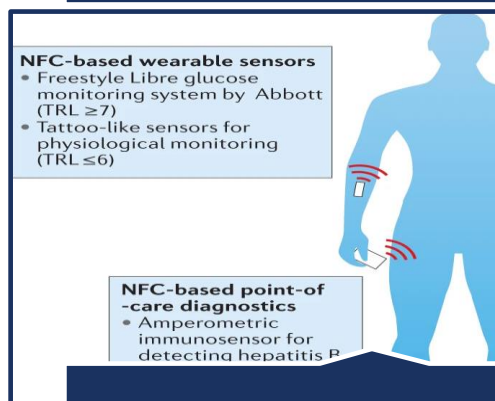
Tatuaggi digitali



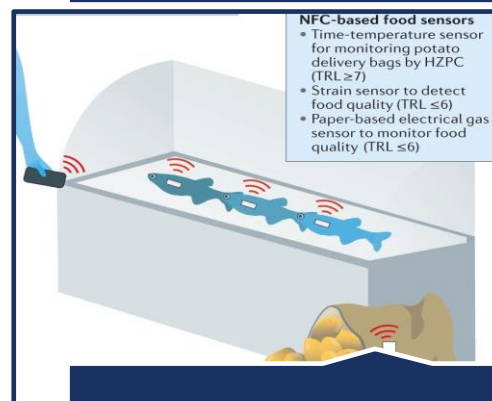
Smart Ring



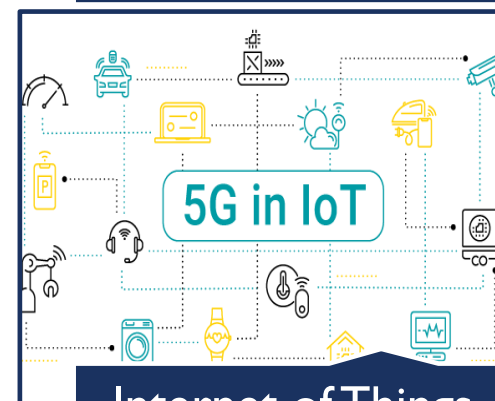
Tag impiantato



Health-Care



Etichette intelligenti



Internet of Things
5G

INTEGRAZIONE IN ANDROID

Passo 1: Aggiungi i permessi NFC al progetto nel Manifest.

```
<uses-permission android:name="android.permission.NFC" />
```

Passo 2: Aggiungi l'azione NFC all'intent-filter dell'attività nel file Manifest.

```
<activity
```

```
...
```

```
<intent-filter>
```

```
<action android:name="android.nfc.action.NDEF_DISCOVERED" />
```

```
<category android:name="android.intent.category.DEFAULT" />
```

```
<data android:mimeType="text/plain" />
```

```
</intent-filter>
```

```
<intent-filter>
```

```
<action android:name="android.nfc.action.TECH_DISCOVERED" />
```

```
<category android:name="android.intent.category.DEFAULT" />
```

```
</intent-filter>
```

```
<intent-filter>
```

```
<action android:name="android.nfc.action.TAG_DISCOVERED" />
```

```
<category android:name="android.intent.category.DEFAULT" />
```

```
</intent-filter>
```

```
</activity>
```

Questi intent-filter consentono:

- Il primo si attiva quando l'utente avvicina un dispositivo NFC che contiene dati di tipo plain text.
- Il secondo si attiva quando l'utente avvicina un dispositivo NFC che corrisponde a una tecnologia specifica.
- Il terzo si attiva quando l'utente avvicina un dispositivo NFC di qualsiasi tipo.

Passo 3: Crea l'interfaccia utente. Creeremo un'interfaccia utente semplice con un solo pulsante per attivare la tecnologia NFC e gestiremo l'attivazione della tecnologia NFC

```
...  
class MainActivity : AppCompatActivity() {  
    private lateinit var nfcAdapter: NfcAdapter  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        val nfcManager = getSystemService(NFC_SERVICE) as NfcManager  
        nfcAdapter = nfcManager.defaultAdapter  
        setContent {  
            NFCApp()  
        }  
    }  
    override fun onResume() {  
        super.onResume()  
        nfcAdapter.enableForegroundDispatch(this, null, null, null)  
    }  
    override fun onPause() {  
        super.onPause()  
        nfcAdapter.disableForegroundDispatch(this)  
    }  
}  
@Preview  
@Composable  
fun NFCApp() {  
    Surface(color = MaterialTheme.colors.background) {  
        val tagContent = remember { mutableStateOf("") }  
        Column(modifier = Modifier.padding(16.dp)) {  
            Text(text = "Contenuto del tag NFC:")  
            Text(text = tagContent.value, modifier = Modifier.padding(bottom = 16.dp))  
            Button(onClick = { /* TODO */ }) {  
                Text("Attiva NFC")  
            }  
        }  
    }  
}
```

- La variabile `nfcAdapter` viene inizializzata nel metodo `onCreate` utilizzando il `NfcManager`, in esso viene richiesto il servizio al SO Android.
- Il metodo `onResume` abilita il dispatch NFC col metodo `enableForegroundDispatch`
- Il metodo `onPause` lo disabilita con il metodo `disableForegroundDispatch` quando l'attività entra in pausa.
- La funzione `NFCApp` definisce l'interfaccia utente, la funzione definisce un'interfaccia utente con un pulsante Attiva NFC e una casella di testo dove verrà letto il contenuto del tag NFC.

Passo 4: Aggiorna la funzione onClick del pulsante:

```
val tag = intent.getParcelableExtra<Tag>(NfcAdapter.EXTRA_TAG)
if (tag != null) {
    val ndef = Ndef.get(tag)
    ndef.connect()
    val message = ndef.ndefMessage
    val payload = message.records[0].payload
    val text = String(payload)
    ndef.close()
    tagContent.value = text
}
```

- Il codice controlla se l'intent passato contiene un oggetto di tipo Tag usando il metodo `getParcelableExtra` della classe `Intent`.
- Se esiste un tag, viene ottenuto l'oggetto `Ndef` associato ad esso usando il metodo statico `get` della classe `Ndef`.
- Successivamente, viene stabilita una connessione con il tag utilizzando il metodo `connect`.
- Viene ottenuto il messaggio NFC Data Exchange Format associato al tag tramite il metodo `ndefMessage`.
- Il payload del primo record del messaggio viene quindi recuperato dall'oggetto `message` utilizzando la proprietà `records` e assegnato alla variabile `payload`. Il payload è un array di byte che rappresenta i dati contenuti nel record.
- Il payload viene convertito in una stringa e assegnato alla variabile `text`.
- La connessione con il tag viene quindi chiusa utilizzando il metodo `close`.
- Il valore della stringa `text` viene assegnato alla variabile `tagContent.value`.

INTEGRAZIONE IN IOS

Passo 1: Aggiungi il supporto NFC

Per aggiungere il supporto NFC al progetto, si seguono i seguenti passaggi:

1. Seleziona il file "Info.plist" del tuo progetto nella navigazione.
2. Fai clic su aggiungi nella sezione Information Property List.
3. Digitare Privacy - NFC Scan Usage Description come chiave e una descrizione del permesso come valore, es: Leggere i tag NFC.
4. Aggiungi il framework CoreNFC al tuo progetto.
Per farlo si seleziona il progetto nella navigazione, e nella scheda General nella sezione Frameworks, Libraries, and Embedded Content. Si aggiunge CoreNFC.framework.

Passo 2: Creazione l'interfaccia utente.

Interfaccia utente con un pulsante che attiva la lettura dei tag NFC e una visualizzazione del contenuto del tag.

Nel file ContentView.swift si inserisce il seguente codice:

```
...
struct ContentView: View {
    @State private var tagContent = ""
    var body: some View {
        VStack {
            Text("Contenuto del tag NFC:")
            Text(tagContent)
                .padding(.bottom, 16)
            Button(action: {
                // Codice per la lettura del tag NFC.
            }) {
                Text("Attiva NFC")
            }
        }
        .padding()
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

In questo codice, abbiamo utilizzato lo stato mutabile di SwiftUI (@State) per aggiornare l'interfaccia utente con il contenuto del tag NFC. Struttura simile a quella mostrata in Android

Passo 3: Aggiungi il codice per la lettura del tag NFC.

Per leggere il contenuto del tag NFC, si deve implementare il protocollo `NFCTagReaderSessionDelegate`.

...

```
extension ContentView: NFCTagReaderSessionDelegate {
    func tagReaderSession(_ session: NFCTagReaderSession, didDetect tags: [NFCTag]) {
        guard let tag = tags.first else {
            session.invalidate(errorMessage: "Nessun tag rilevato")
            return
        }
        session.connect(to: tag) { (error: Error?) in
            if error != nil {
                session.invalidate(errorMessage: "Errore di connessione al tag NFC")
                return
            }
            if case let NFCTag.miFare(miFareTag) = tag {
                let apdu = NFCISO7816APDU(instructionClass: 0x00, instructionCode: 0xB0, p1Parameter: 0x00, p2Parameter: 0x00, data: Data(), expectedResponseLength: 16)
                miFareTag.sendCommand(apdu: apdu) { (responseData, sw1, sw2, error) in
                    if let error = error {
                        session.invalidate(errorMessage: "Errore nella lettura del tag NFC: \(error.localizedDescription)")
                        return
                    }
                    if sw1 == 0x90 && sw2 == 0x00 {
                        let tagContent = String(data: responseData ?? Data(), encoding: .utf8) ?? "Nessun contenuto trovato"
                        DispatchQueue.main.async {
                            self.tagContent = tagContent
                        }
                    } else {
                        session.invalidate(errorMessage: "Errore nella lettura del tag NFC")
                    }
                }
            } else {
                session.invalidate(errorMessage: "Il tag NFC non è supportato")
            }
        }
    }
}
```

In questa estensione, si è implementata la funzione `tagReaderSession` che viene chiamata quando viene rilevato un tag NFC.

Questa funzione controlla se il tag è un tag MiFare, quindi crea un'Application Protocol Data Unit per leggere il contenuto del tag e lo invia al tag con la funzione `miFareTag.sendCommand`.

Se la lettura è avvenuta con successo, aggiorniamo l'interfaccia utente con il contenuto del tag.

...

Passo 4: Attiva la lettura del tag NFC

Attivazione della lettura del tag NFC quando si fa clic sul Bottone. Cambiamo il bottone in:

```
Button(action: {  
    let session = NFCTagReaderSession(pollingOption: .iso14443, delegate: self)  
    session.alertMessage = "Tieni il tag NFC vicino all'iPhone per leggerlo."  
    session.begin()  
}) {  
    Text("Attiva NFC")  
}
```

Si è creata una nuova sessione `NFCTagReaderSession` ed è stata avviata con la funzione `begin()`.
E' stato anche impostato il messaggio di avviso della sessione con la proprietà `alertMessage`.

CONCLUSIONI

I tag NFC svolgeranno un ruolo inevitabile nei futuri dispositivi intelligenti per funzioni più integrate, nei trasporti, nell'industria aeronautica, nelle spedizioni e nell'industria manifatturiera per l'automazione di particolari attività. L'integrazione della tecnologia NFC nel moderno processo di comunicazione dei dati e di transazione garantisce convenienza, risparmio di tempo, efficienza energetica e soprattutto una maggiore sicurezza.





Grazie per l'attenzione

ANDREA BEDEI

ANDREA.BEDEI2@STUDIO.UNIBO.IT

PRESENTAZIONE NFC PER
PROGRAMMAZIONE DI SISTEMI MOBILE