Tương tác với các thiết bị phần cứng của điện thoại.

1. Tương tác với Cảm biến (Sensors)

1.1. Giới thiệu về Sensor API và Các loại cảm biến phổ biến

Các loại cảm biến:

- Cảm biến gia tốc: đo tốc độ và hướng di chuyển.
- Cảm biến con quay hồi chuyển: đo xoay và thay đổi hướng.
- Cảm biến ánh sáng: đo cường độ ánh sáng.
- Cảm biến địa từ: đo hướng từ trường.

Ứng dụng của cảm biến: Tạo các ứng dụng đếm bước chân, điều khiển trò chơi, phát hiện cử động, đo độ nghiêng của thiết bị,…

1.2. Các bước sử dụng Sensor API

Sensor API trong Android cung cấp khả năng thu thập dữ liệu từ các cảm biến tích hợp trên thiết bị.

Các bước chính để sử dụng Sensor API gồm: khởi tạo SensorManager, đăng ký SensorEventListener, và hủy đăng ký khi không cần sử dụng.

Chi tiết các bước:

Bước 1: Khởi tạo SensorManager

Tạo đối tượng SensorManager: SensorManager là lớp quản lý tất cả các cảm biến trên thiết bị Android. Để sử dụng cảm biến, bạn cần khởi tạo một đối tượng SensorManager.

Cách khởi tạo:

```
SensorManager sensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
```

Chọn loại cảm biến cần sử dụng:

Android hỗ trợ nhiều loại cảm biến, như:

- Cảm biến gia tốc (Accelerometer): Đo gia tốc trên ba trục X, Y, Z.
- Cảm biến ánh sáng (Light): Đo độ sáng môi trường xung quanh.
- Cảm biến từ trường (Magnetic Field): Đo từ trường xung quanh.
- Cảm biến tiệm cận (Proximity): Đo khoảng cách của vật thể gần thiết bị.

Để lấy cảm biến cụ thể, sử dụng SensorManager.getDefaultSensor() và chỉ định loại cảm biến (Sensor.TYPE_ACCELEROMETER, Sensor.TYPE_LIGHT,...).

Ví dụ:

```
Sensor accelerometer =
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
Sensor lightSensor = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
```

Bước 2: Đăng ký SensorEventListener

Sau khi khởi tạo cảm biến, cần đăng ký một SensorEventListener để nhận dữ liệu từ cảm biến.

Tạo một SensorEventListener: SensorEventListener là giao diện để lắng nghe sự kiện từ cảm biến, bao gồm hai phương thức chính:

- onSensorChanged(SensorEvent event): Gọi mỗi khi có dữ liệu cảm biến mới.
- onAccuracyChanged(Sensor sensor, int accuracy): Gọi khi độ chính xác của cảm biến thay đổi.

Ví dụ:

```
SensorEventListener sensorEventListener = new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent event) {
        // Xử lý dữ liệu cảm biến tại đây
        if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
            float x = event.values[0];
            float y = event.values[1];
            float z = event.values[2];
            // Xử lý dữ liệu gia tốc
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // Xử lý khi độ chính xác của cảm biến thay đổi
    }
};
```

Đăng ký SensorEventListener với cảm biến: Sử dụng sensorManager.registerListener() để đăng ký SensorEventListener với cảm biến đã chọn.

Định nghĩa tần suất nhận dữ liệu từ cảm biến, với các hằng số như:

- SensorManager.SENSOR_DELAY_NORMAL: Lấy dữ liệu khoảng 200ms một lần.
- SensorManager.SENSOR_DELAY_UI: Tối ưu cho cập nhật giao diện người dùng.
- SensorManager.SENSOR_DELAY_GAME: Tần suất cao cho các ứng dụng cần phản hồi nhanh.
- SensorManager.SENSOR_DELAY_FASTEST: Tần suất cao nhất, phù hợp với các ứng dụng game hoặc đo lường chính xác.

Ví dụ:

```
sensorManager.registerListener(sensorEventListener, accelerometer,
SensorManager.SENSOR_DELAY_NORMAL);
```

Xử lý dữ liệu cảm biến trong onSensorChanged(): Khi nhận dữ liệu mới, onSensorChanged() sẽ được gọi với đối tượng SensorEvent, cung cấp:

- sensor: Loại cảm biến phát sinh sự kiện.
- values: Mảng chứa các giá trị đo được, tùy thuộc vào loại cảm biến. Ví dụ, cảm biến gia tốc trả về các giá trị trên ba trục X, Y, Z.

Ví dụ xử lý trong onSensorChanged():

```
@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_LIGHT) {
        float lightIntensity = event.values[0];
        // Xử lý độ sáng, ví dụ thay đổi giao diện theo mức ánh sáng
    }
}
```

Bước 3: Huỷ đăng ký khi không sử dụng

Để tiết kiệm tài nguyên hệ thống và giảm hao pin, bạn nên hủy đăng ký SensorEventListener khi không cần thu thập dữ liệu cảm biến nữa (ví dụ: khi ứng dụng ở chế độ nền hoặc khi đóng Activity).

Huỷ đăng ký SensorEventListener: Sử dụng sensorManager.unregisterListener() để hủy đăng ký cảm biến, giúp giảm bớt tải CPU và tiết kiệm pin.

Ví dụ:

```
@Override
protected void onPause() {
    super.onPause();
    sensorManager.unregisterListener(sensorEventListener);
}
```

Giải thích về tiết kiệm tài nguyên: Cảm biến tiêu tốn năng lượng, đặc biệt là các cảm biến đòi hỏi tần suất cập nhật cao (như cảm biến gia tốc trong game). Do đó, chỉ nên kích hoạt cảm biến khi thực sự cần thiết và hủy khi không dùng để kéo dài thời gian sử dụng pin và giảm tải hệ thống.

Lưu ý: Trong một số ứng dụng đòi hỏi sự phản hồi liên tục từ cảm biến (như đo bước chân), cần đảm bảo tối ưu hóa chu kỳ sử dụng cảm biến để tránh hao pin.

2. Tương tác với Bluetooth

2.1. Giới thiệu về Bluetooth Classic và BLE

Bluetooth là công nghệ không dây phổ biến, giúp kết nối các thiết bị với nhau trong một khoảng cách ngắn. Công nghệ này có hai phiên bản chính:

- Bluetooth Classic: Phiên bản gốc, thích hợp cho truyền dữ liệu liên tục, tốc độ cao.
- Bluetooth Low Energy (BLE): Phiên bản cải tiến, tiêu tốn ít năng lượng hơn, thiết kế dành cho các thiết bị IoT và ứng dụng cần tiết kiệm pin.

So sánh 2 loại Bluetooth:

Thông tin so sánh	Bluetooth Classic	Bluetooth Low Energy (BLE)
Ưu điểm	Tốc độ truyền tải dữ liệu cao: Bluetooth Classic có tốc độ truyền tải dữ liệu tối đa cao hơn (khoảng 2-3 Mbps), lý tưởng cho các ứng dụng yêu cầu truyền tải âm thanh và video. Hỗ trợ các ứng dụng đòi hỏi kết nối liên tục: Kết nối ổn định hơn và phù hợp cho các ứng dụng yêu cầu kết nối liên tục, như tai nghe không dây, loa Bluetooth, chuột và bàn phím.	Tiêu thụ năng lượng thấp: BLE được thiết kế để hoạt động trong thời gian dài mà không tiêu tốn quá nhiều năng lượng, làm cho nó lý tưởng cho các thiết bị chạy bằng pin, như đồng hồ thông minh, cảm biến, thiết bị y tế. Khả năng kết nối linh hoạt và phạm vi hoạt động rộng: BLE hỗ trợ các kết nối ngắt quãng và chỉ hoạt động khi cần thiết, giúp tiết kiệm năng lượng và giảm độ trễ khi thiết bị cần truy cập dữ liệu theo chu kỳ. Tương thích cao với thiết bị IoT: BLE thường được dùng trong các thiết bị IoT và các hệ thống beacon để phát tín hiệu và giao tiếp mà không cần kết nối liên tục.
Nhược điểm	Tiêu thụ năng lượng cao: Bluetooth Classic tiêu tốn năng lượng nhiều hơn, khiến nó không lý tưởng cho các thiết bị nhỏ gọn hoặc thiết bị chạy bằng pin trong thời gian dài. Phạm vi kết nối giới hạn: Dù có thể hỗ trợ phạm vi từ 10 đến 100 mét, Bluetooth Classic thường giới hạn ở khoảng 10 mét trong môi trường thông thường. Tương thích kém với thiết bị IoT nhỏ gọn: Vì tiêu thụ năng lượng cao, nó ít phù hợp cho các thiết bị IoT nhỏ gọn, như cảm biến sức khỏe hoặc thiết bị đeo.	Tốc độ truyền tải dữ liệu thấp hơn: Với tốc độ truyền tối đa chỉ khoảng 1 Mbps, BLE không lý tưởng cho các ứng dụng yêu cầu truyền dữ liệu nhanh và liên tục, như phát âm thanh chất lượng cao. Khả năng xử lý và truyền dữ liệu hạn chế: BLE thường giới hạn trong truyền dữ liệu nhỏ gọn hoặc các thông báo trạng thái, không phù hợp cho các ứng dụng yêu cầu băng thông cao. Phạm vi ngắn hơn trong thực tế: Dù BLE có thể đạt phạm vi tối đa khoảng 100 mét, trong điều kiện thực tế, phạm vi này thường thấp hơn nhiều.
Ứng dụng chính	Âm thanh không dây: Tai nghe Bluetooth, loa không dây, và các thiết bị âm thanh chất lượng cao khác.	Thiết bị đeo và theo dõi sức khỏe: Thiết bị đeo (như Fitbit), cảm biến sức khỏe, và các thiết bị y tế vì khả năng tiết kiệm năng lượng tốt.

	Đồng bộ dữ liệu: Đồng bộ hóa dữ liệu giữa điện thoại và máy tính hoặc giữa các thiết bị cũ chưa hỗ trợ BLE.	Thiết bị IoT và beacon: Được sử dụng trong các ứng dụng định vị trong nhà, như thiết bị beacon và các thiết bị IoT sử dụng năng lượng thấp. Smart home: Các thiết bị thông minh như ổ khóa, đèn, và điều khiển từ xa thường sử dụng BLE do yêu cầu tiết kiệm năng lượng.

2.2. Các bước kết nối và truyền dữ liệu qua Bluetooth

Để kết nối và truyền dữ liệu qua Bluetooth trong Android, cần thực hiện các bước như yêu cầu quyền, bật Bluetooth, tìm kiếm thiết bị, ghép nối và truyền dữ liệu qua BluetoothSocket.

Bước 1: Yêu cầu quyền truy cập và bật Bluetooth

Yêu cầu quyền sử dụng Bluetooth trong AndroidManifest: Trước khi sử dụng Bluetooth, cần khai báo quyền trong tệp AndroidManifest.xml. Quyền cần thiết bao gồm:

- BLUETOOTH để sử dụng các tính năng Bluetooth.
- BLUETOOTH_ADMIN để quản lý Bluetooth, bao gồm bật/tắt và quét thiết bị.

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

Kiểm tra và bật Bluetooth:

- Sử dụng BluetoothAdapter để kiểm tra xem Bluetooth đã được bật chưa.
- Nếu Bluetooth chưa được bật, yêu cầu người dùng bật thông qua một Intent yêu cầu bật Bluetooth.

Ví dụ:

```
BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
if (bluetoothAdapter == null) {
    // Thiết bị không hỗ trợ Bluetooth
    Toast.makeText(this, "Thiết bị của bạn không hỗ trợ Bluetooth",
Toast.LENGTH_LONG).show();
} else if (!bluetoothAdapter.isEnabled()) {
    // Yêu cầu bật Bluetooth
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
```

Bước 2: Tìm kiếm thiết bị gần đó

Bắt đầu tìm kiếm các thiết bị Bluetooth gần đó:

- Để tìm kiếm các thiết bị Bluetooth xung quanh, sử dụng BluetoothAdapter.startDiscovery().
- Khi quá trình tìm kiếm bắt đầu, các thiết bị được phát hiện sẽ được gửi về qua một BroadcastReceiver.

- Cần đăng ký một BroadcastReceiver để lắng nghe sự kiện BluetoothDevice.ACTION_FOUND khi tìm thấy một thiết bị.

Ví dụ:

```
// Đăng ký BroadcastReceiver để nhận thông báo khi phát hiện thiết bị
BroadcastReceiver receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            // Nhận thông tin thiết bị Bluetooth
            BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            String deviceName = device.getName();
            String deviceAddress = device.getAddress();
            // Thêm thiết bị vào danh sách hiển thị
        }
    }
};

// Bắt đầu tìm kiếm thiết bị
bluetoothAdapter.startDiscovery();
```

Quản lý quyền tìm kiếm thiết bị: Đối với Android 6.0 trở lên, cần yêu cầu quyền vị trí (ACCESS_FINE_LOCATION) trước khi tìm kiếm các thiết bị Bluetooth xung quanh.

Ngừng tìm kiếm khi tìm thấy thiết bị cần kết nối: Để tiết kiệm pin và tài nguyên, nên gọi bluetoothAdapter.cancelDiscovery() để dừng quá trình tìm kiếm sau khi tìm thấy thiết bị hoặc khi không cần tiếp tục quét.

Bước 3: Ghép nối và truyền dữ liệu

Thực hiện ghép nối với thiết bị:

- Để ghép nối với một thiết bị, sử dụng phương thức device.createBond() nếu thiết bị chưa được ghép nối.
- Khi đã ghép nối, thiết bị sẽ có thể kết nối qua BluetoothSocket.

Ví dụ:

```
BluetoothDevice device = bluetoothAdapter.getRemoteDevice(deviceAddress);
device.createBond();
```

Kết nối và tạo BluetoothSocket:

- Sau khi ghép nối, tạo một kết nối với thiết bị thông qua BluetoothSocket.

- Sử dụng UUID (Universally Unique Identifier) để tạo socket kết nối với thiết bị. UUID này phải phù hợp giữa hai thiết bị.

Ví dụ:

```
BluetoothSocket bluetoothSocket = null;
try {
    bluetoothSocket = device.createRfcommSocketToServiceRecord(MY_UUID);
    bluetoothSocket.connect();
} catch (IOException e) {
    e.printStackTrace();
}
```

Gửi và nhận dữ liệu qua BluetoothSocket: Sau khi kết nối thành công, có thể gửi và nhận dữ liệu qua BluetoothSocket bằng cách sử dụng InputStream và OutputStream.

- Gửi dữ liệu: Sử dụng OutputStream để gửi dữ liệu qua Bluetooth.

Ví dụ:

```
OutputStream outputStream = bluetoothSocket.getOutputStream();
outputStream.write("Hello, Bluetooth!".getBytes());
```

- Nhận dữ liệu: Sử dụng InputStream để đọc dữ liệu được gửi từ thiết bị kết nối.

Ví dụ:

```
InputStream inputStream = bluetoothSocket.getInputStream();
byte[] buffer = new byte[1024];
int bytes;
while ((bytes = inputStream.read(buffer)) != -1) {
    String receivedData = new String(buffer, 0, bytes);
    // Xử lý dữ liệu nhận được
}
```

Đóng kết nối khi hoàn thành: Để giải phóng tài nguyên sau khi hoàn thành việc truyền dữ liệu, hãy đóng BluetoothSocket.

Ví dụ:

```
try {
    bluetoothSocket.close();
} catch (IOException e) {
    e.printStackTrace();
}
```

3. NFC (Near Field Communication)

3.1. Giới thiệu về NFC

3.1.1. NFC là gì?

NFC (Near Field Communication) là công nghệ giao tiếp không dây phạm vi ngắn, cho phép hai thiết bị trao đổi dữ liệu khi đặt gần nhau (thường là trong khoảng cách 4 cm). NFC phát triển từ công nghệ RFID (Radio Frequency Identification) và ngày nay được tích hợp trong nhiều thiết bị như điện thoại thông minh, thẻ tín dụng, và các thiết bị đeo thông minh.

Nguyên lý hoạt động của NFC

- Dựa trên cảm ứng từ trường: NFC hoạt động dựa vào cảm ứng từ trường giữa hai anten NFC, giúp truyền dữ liệu khi các thiết bị nằm trong phạm vi cực ngắn.
- Tần số hoạt động: NFC hoạt động ở tần số 13.56 MHz và tốc độ truyền dữ liệu lên đến 424 kbps, đủ để truyền các dữ liệu nhỏ như thông tin thanh toán hoặc một số dữ liệu trạng thái.
- Chế độ hoạt động: NFC có thể làm việc ở chế độ chủ động (thiết bị phát tín hiệu) hoặc thụ động (thiết bị nhận tín hiệu), tùy vào cách thiết bị giao tiếp với nhau.

3.1.2. Các chế độ hoạt động của NFC và ứng dụng của nó.

NFC có ba chế độ hoạt động chính, mỗi chế độ có ứng dụng và phương thức sử dụng khác nhau:

a) Reader/Writer (Đọc/Ghi)

Chức năng: Thiết bị NFC (như điện thoại) hoạt động như một máy quét để đọc hoặc ghi dữ liệu lên các thẻ NFC thụ động. Chế độ này cho phép đọc dữ liệu từ thẻ và ghi nội dung mới vào thẻ.

Ứng dụng: Đọc thông tin từ thẻ NFC được cài đặt trên poster quảng cáo, thẻ nhân viên, hoặc thẻ vé sự kiện.

Ví dụ: Khi bạn chạm điện thoại vào thẻ quảng cáo có tích hợp NFC, điện thoại có thể mở một trang web hoặc hiển thị thông tin sản phẩm được lưu trong thẻ.

b) Peer-to-Peer (P2P)

Chức năng: Cho phép hai thiết bị NFC (như hai điện thoại) kết nối và chia sẻ dữ liệu với nhau. Cả hai thiết bị sẽ lần lượt chuyển đổi giữa chế độ chủ động và thụ động để gửi và nhận dữ liệu.

Ứng dụng: Chia sẻ dữ liệu nhanh chóng giữa hai thiết bị, chẳng hạn như danh bạ, hình ảnh, hoặc đường dẫn (URL) mà không cần kết nối Wi-Fi hay Bluetooth.

Ví dụ: Android Beam (trên các thiết bị Android cũ) cho phép người dùng chạm hai điện thoại NFC lại với nhau để gửi ảnh hoặc tài liệu đơn giản qua NFC.

c) Card Emulation (Giả lập thẻ)

Chức năng: Thiết bị NFC (như điện thoại) hoạt động như một thẻ NFC hoặc thẻ tín dụng, giúp thực hiện các giao dịch và thanh toán không tiếp xúc. Chế độ này sử dụng công nghệ Host-based Card Emulation (HCE) để giả lập thiết bị thành một thẻ NFC.

Ứng dụng: Sử dụng điện thoại như một thẻ ngân hàng hoặc thẻ giao thông công cộng để thanh toán hoặc kiểm tra vé điện tử.

Ví dụ: Các ứng dụng thanh toán di động như Google Pay và Samsung Pay cho phép bạn dùng điện thoại như một thẻ tín dụng tại các máy POS hỗ trợ NFC.

3.2. Các bước tích hợp NFC trong ứng dụng

Để tích hợp NFC vào ứng dụng Android, cần thực hiện các bước từ kiểm tra hỗ trợ của thiết bị, phát hiện và xử lý thẻ NFC, cho đến việc đọc và ghi dữ liệu vào thẻ. Dưới đây là hướng dẫn chi tiết cho từng bước.

Bước 1: Kiểm tra hỗ trợ NFC

Trước khi tích hợp NFC, ứng dụng cần kiểm tra xem thiết bị có hỗ trợ NFC không và nếu cần, hướng dẫn người dùng bật tính năng này.

Kiểm tra phần cứng hỗ trợ NFC: Sử dụng NfcAdapter để kiểm tra xem thiết bị có hỗ trợ NFC. NfcAdapter là lớp quản lý giao tiếp NFC trên thiết bị Android. Cách kiểm tra:

```
NfcAdapter nfcAdapter = NfcAdapter.getDefaultAdapter(this);
if (nfcAdapter == null) {
    // Thiết bị không hỗ trợ NFC
    Toast.makeText(this, "Thiết bị của bạn không hỗ trợ NFC",
Toast.LENGTH_LONG).show();
} else if (!nfcAdapter.isEnabled()) {
    // NFC chưa được bật
    Toast.makeText(this, "Vui lòng bật NFC", Toast.LENGTH_LONG).show();
    // Có thể hướng dẫn người dùng mở cài đặt để bật NFC
    startActivity(new Intent(Settings.ACTION_NFC_SETTINGS));
} else {
    // Thiết bị có hỗ trợ và NFC đã bật
}
```

Yêu cầu quyền sử dụng NFC trong AndroidManifest: Để sử dụng NFC trong ứng dụng, cần khai báo quyền NFC trong tệp AndroidManifest.xml.

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

Bước 2: Phát hiện và xử lý thẻ NFC

Sau khi xác nhận thiết bị có hỗ trợ NFC, bước tiếp theo là phát hiện và xử lý thẻ NFC khi thẻ được đưa lại gần thiết bị.

Đăng ký NfcAdapter để phát hiện thẻ NFC: Sử dụng NfcAdapter và thiết lập PendingIntent để ứng dụng có thể nhận diện thẻ khi thiết bị tiếp xúc với một thẻ NFC.

Ví dụ:

```
NfcAdapter nfcAdapter = NfcAdapter.getDefaultAdapter(this);
PendingIntent pendingIntent = PendingIntent.getActivity(
    this, 0, new Intent(this,
getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);
nfcAdapter.enableForegroundDispatch(this, pendingIntent, null, null);
```

Xử lý sự kiện khi thẻ NFC được tiếp xúc: Ghi đè phương thức onNewIntent(Intent intent) trong Activity để xử lý sự kiện khi phát hiện thẻ NFC.

Phương thức này sẽ nhận thông báo khi một thẻ NFC mới được phát hiện và cho phép thực hiện đọc hoặc ghi dữ liệu.

Ví dụ:

```
@Override
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(intent.getAction())) {
        // Xử lý thẻ NFC được phát hiện
        Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
        // Thực hiện đọc hoặc ghi dữ liệu từ/đến thẻ
    }
}
```

Bước 3: Đọc và ghi dữ liệu lên thẻ NFC

Khi phát hiện một thẻ NFC, ứng dụng có thể đọc hoặc ghi dữ liệu lên thẻ. Tùy thuộc vào loại thẻ (chẳng hạn như NDEF), ứng dụng sẽ sử dụng các giao thức và phương thức cụ thể để thao tác với dữ liệu.

Đọc dữ liệu từ thẻ NFC: Thẻ NFC thường sử dụng định dạng NDEF (NFC Data Exchange Format) để lưu trữ dữ liệu. Để đọc dữ liệu, đầu tiên cần kiểm tra nếu thẻ có hỗ trợ NDEF không.

Sử dụng lớp Ndef để đọc các bản ghi (record) từ thẻ.

Ví dụ:

```
Ndef ndef = Ndef.get(tag);
if (ndef != null) {
```

```
    try {
        ndef.connect();
        NdefMessage ndefMessage = ndef.getNdefMessage();
        if (ndefMessage != null) {
            for (NdefRecord record : ndefMessage.getRecords()) {
                // Đọc từng bản ghi, ví dụ đọc Text từ bản ghi
                String payload = new String(record.getPayload());
                // Xử lý dữ liệu từ thẻ
            }
        }
    } catch (IOException | FormatException e) {
        e.printStackTrace();
    } finally {
        try {
            ndef.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Ghi dữ liệu vào thẻ NFC: Để ghi dữ liệu, cũng cần kiểm tra nếu thẻ hỗ trợ NDEF và có thể ghi dữ liệu.

Tạo một NdefMessage mới và ghi vào thẻ qua lớp Ndef.

Ví dụ:

```
NdefRecord ndefRecord = NdefRecord.createTextRecord("en", "Nội dung cần ghi lên thẻ");
NdefMessage ndefMessage = new NdefMessage(new NdefRecord[]{ndefRecord});

Ndef ndef = Ndef.get(tag);
if (ndef != null) {
    try {
        ndef.connect();
        if (ndef.isWritable()) {
            ndef.writeNdefMessage(ndefMessage);
            Toast.makeText(this, "Đã ghi dữ liệu lên thẻ", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "Thẻ không cho phép ghi", Toast.LENGTH_SHORT).show();
        }
    } catch (IOException | FormatException e) {
```

```
            e.printStackTrace();
        } finally {
            try {
                ndef.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

Hướng dẫn thực hành

Bài 1. Ứng dụng đếm bước chân.

Sử dụng cảm biến bước chân hoặc cảm biến gia tốc xây dựng ứng dụng đếm bước chân.

Hướng dẫn:

1. Tạo ứng dụng.

2. Khai báo quyền sử dụng cảm biến.

Khai báo quyền sử dụng cảm biến bước chân trong tệp AndroidManifest.xml:

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

3. Thiết lập giao diện người dùng

Trong tệp activity_main.xml, tạo giao diện đơn giản để hiển thị số bước chân:

```
    <TextView
        android:id="@+id/stepCountView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Số bước chân đã đi: 0"
        android:textSize="24sp"
        android:layout_gravity="center"/>
```

4. Cài đặt Activity chính

Trong MainActivity.java, sử dụng SensorManager và Sensor để đếm bước chân.

Bước 1: Khởi tạo SensorManager và cảm biến.

```
public class MainActivity extends AppCompatActivity implements
SensorEventListener {
    private SensorManager sensorManager;
    private Sensor stepSensor;
    private TextView stepCountView;
```

```
    private int stepCount = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Gán view hiển thị số bước
        stepCountView = findViewById(R.id.stepCountView);

        // Khởi tạo SensorManager và cảm biến bước chân
        sensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
        if (sensorManager.getDefaultSensor(Sensor.TYPE_STEP_DETECTOR) != null)
{
            // Nếu thiết bị hỗ trợ cảm biến bước chân
            stepSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_STEP_DETECTOR);
        } else if (sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) !=
null) {
            // Dùng cảm biến gia tốc nếu không có cảm biến bước chân
            stepSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        }
    }
```

Bước 2: Đăng ký SensorEventListener để lắng nghe sự kiện bước chân

Trong onResume() và onPause(), đăng ký và hủy đăng ký SensorEventListener để nhận dữ liệu cảm biến:

```
@Override
protected void onResume() {
    super.onResume();
    if (stepSensor != null) {
        sensorManager.registerListener(this, stepSensor,
SensorManager.SENSOR_DELAY_UI);
    }
}

@Override
protected void onPause() {
    super.onPause();
    sensorManager.unregisterListener(this);
}
```

Bước 3: Xử lý sự kiện cảm biến trong onSensorChanged

Ví dụ dùng cảm biến bước chân (TYPE_STEP_DETECTOR), mỗi sự kiện cảm biến tương ứng với một bước chân.

```
@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_STEP_DETECTOR) {
        // Tăng số bước mỗi khi nhận sự kiện bước chân
        stepCount++;
        stepCountView.setText("Số bước chân đã đi: " + stepCount);
    }

}

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {

}
```

Lưu ý: Do máy ảo không hỗ trợ các cảm biến bước chân nên để thuận tiện nhất nên sử dụng máy thật để thực thi và kiểm tra ứng dụng.

Bài 2. Kết nối Bluetooth.

Hướng dẫn.

1. Tạo ứng dụng.

2. Khai báo quyền sử dụng Bluetooth.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

3. Thiếp lập giao diện người dùng.

```
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Danh sách các thiết bị đã kết nối"
        android:textSize="25sp"
        android:textAlignment="center"
        android:layout_marginBottom="10dp" />

    <ProgressBar
        android:id="@+id/progressBar"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:visibility="gone"
        style="@android:style/Widget.DeviceDefault.Light.ProgressBar.Large"
        android:layout_marginBottom="20dp" />

    <TextView
        android:id="@+id/noDeviceText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Không tìm thấy thiết bị nào"
        android:textSize="16sp"
        android:textAlignment="center"
        android:visibility="gone"
        android:layout_marginBottom="20dp" />

    <ListView
        android:id="@+id/deviceList"
        android:layout_width="match_parent"
        android:layout_height="500dp"
        android:divider="@android:color/darker_gray"
        android:dividerHeight="1dp"
        android:background="@android:color/white"
        android:layout_marginBottom="20dp" />

    <Button
        android:id="@+id/scanButton"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:text="Quét thiết bị"
        android:backgroundTint="#4CAF50"
        android:textColor="@android:color/white"
        android:textSize="18sp" />
```

4. Cài đặt Activity chính.

Tóm tắt quá trình xử lý:

1. Khởi tạo và kiểm tra Bluetooth:

- bluetoothAdapter = BluetoothAdapter.getDefaultAdapter(); — Kiểm tra xem thiết bị có hỗ trợ Bluetooth không.

- Nếu không hỗ trợ, ứng dụng sẽ hiển thị thông báo và kết thúc.

2. Kiểm tra và yêu cầu quyền Bluetooth:

- Phương thức checkAndRequestPermissions() được gọi để kiểm tra và yêu cầu quyền Bluetooth cần thiết (ví dụ: quyền quét thiết bị, kết nối Bluetooth).

3. Hiển thị thiết bị đã ghép nối (Paired Devices):

- Phương thức showPairedDevices() được gọi để lấy danh sách các thiết bị đã ghép nối và hiển thị chúng trong ListView.

4. Quét thiết bị Bluetooth:

- Khi người dùng nhấn nút quét, phương thức startDiscovery() được gọi để bắt đầu quét các thiết bị Bluetooth xung quanh.

- Quá trình quét kéo dài 10 giây, sau đó sẽ dừng lại và thông báo nếu không có thiết bị nào được tìm thấy.

- BluetoothDevice.ACTION_FOUND được sử dụng để nhận thông tin thiết bị khi tìm thấy.

5. Kết nối đến thiết bị Bluetooth:

- Khi người dùng chọn một thiết bị từ danh sách, phương thức connectToDevice() sẽ tạo một kết nối Bluetooth đến thiết bị đó.

- Một socket Bluetooth được tạo để giao tiếp với thiết bị thông qua RFCOMM.

- Dữ liệu được gửi từ Android tới thiết bị (ví dụ: "Hello from Android") và nhận dữ liệu từ thiết bị. Các thông điệp nhận được sẽ được hiển thị thông qua Toast.

6. Xử lý quyền Bluetooth:

- Nếu ứng dụng chưa có quyền Bluetooth cần thiết, nó sẽ yêu cầu quyền từ người dùng thông qua requestBluetoothPermissions().

7. Quản lý kết nối và tài nguyên:

- Khi ứng dụng không còn sử dụng Bluetooth nữa (hoặc khi người dùng thoát ứng dụng), phương thức onDestroy() được gọi để hủy đăng ký BroadcastReceiver và đóng kết nối Bluetooth (nếu có).

8. Quản lý các kết nối Bluetooth:

- Phương thức BluetoothSocket.connect() dùng để thực hiện kết nối đến thiết bị và BluetoothSocket.close() để đóng kết nối khi xong.

Chi tiết mã nguồn xử lý.

```
public class MainActivity extends AppCompatActivity {

    private BluetoothAdapter bluetoothAdapter;  // Bluetooth Adapter để kiểm
tra và thao tác với Bluetooth
    private BluetoothSocket bluetoothSocket;    // Kết nối Bluetooth đến thiết
bị
```

```
    private ArrayAdapter<String> deviceArrayAdapter;  // Adapter để hiển thị
danh sách thiết bị
    private ArrayList<BluetoothDevice> deviceList = new ArrayList<>();  // Danh
sách lưu trữ các thiết bị Bluetooth
    private ProgressBar progressBar;  // ProgressBar hiển thị khi đang quét
thiết bị
    private TextView noDeviceText;    // TextView thông báo không có thiết bị

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Khởi tạo Bluetooth Adapter
        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        ListView deviceListView = findViewById(R.id.deviceList);  // ListView
hiển thị danh sách thiết bị
        Button scanButton = findViewById(R.id.scanButton);        // Nút quét
thiết bị
        progressBar = findViewById(R.id.progressBar);  // ProgressBar hiển thị
khi quét
        noDeviceText = findViewById(R.id.noDeviceText);  // Thông báo không có
thiết bị

        // Kiểm tra và yêu cầu quyền Bluetooth
        checkAndRequestPermissions();

        // Kiểm tra nếu thiết bị không hỗ trợ Bluetooth
        if (bluetoothAdapter == null) {
            Toast.makeText(this, "Thiết bị không hỗ trợ Bluetooth",
Toast.LENGTH_SHORT).show();
            finish();
            return;
        }

        // Khởi tạo ArrayAdapter để hiển thị danh sách thiết bị
        deviceArrayAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1);
        deviceListView.setAdapter(deviceArrayAdapter);

        // Hiển thị các thiết bị đã ghép nối
        showPairedDevices();

        // Khi nhấn nút quét, bắt đầu quét thiết bị Bluetooth
```

```
        scanButton.setOnClickListener(v -> {
            progressBar.setVisibility(View.VISIBLE);  // Hiển thị ProgressBar
            noDeviceText.setVisibility(View.GONE);  // Ẩn thông báo không có
thiết bị
            startDiscovery();  // Bắt đầu quét thiết bị
        });

        // Xử lý khi người dùng chọn một thiết bị trong danh sách
        deviceListView.setOnItemClickListener((adapterView, view, position, id)
-> {
            BluetoothDevice device = deviceList.get(position);
            connectToDevice(device);  // Kết nối tới thiết bị đã chọn
        });
    }

    // Kiểm tra và yêu cầu quyền Bluetooth
    private void checkAndRequestPermissions() {
    }

    // Bắt đầu quét thiết bị Bluetooth
    private void startDiscovery() {
        // Kiểm tra quyền đã được cấp hay chưa
        if (!hasBluetoothPermissions()) {
            Toast.makeText(this, "Ứng dụng cần quyền để quét thiết bị
Bluetooth.", Toast.LENGTH_SHORT).show();
            requestBluetoothPermissions();
            return;
        }

        // Nếu quyền đã được cấp, bắt đầu quét thiết bị
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH_SCAN) != PackageManager.PERMISSION_GRANTED) {
            return;
        }

        // Nếu đang quét, hủy quét và bắt đầu lại
        if (bluetoothAdapter.isDiscovering()) {
            bluetoothAdapter.cancelDiscovery();
        }

        bluetoothAdapter.startDiscovery();  // Bắt đầu quét
        registerReceiver(receiver, new
IntentFilter(BluetoothDevice.ACTION_FOUND));  // Đăng ký Receiver để nhận thiết
bị
```

```
        // Hủy quét sau 10 giây nếu không có thiết bị nào được tìm thấy
        new Handler().postDelayed(() -> {
            bluetoothAdapter.cancelDiscovery();
            progressBar.setVisibility(View.GONE);  // Ẩn ProgressBar
            if (deviceList.isEmpty()) {
                noDeviceText.setVisibility(View.VISIBLE);  // Hiển thị thông
báo không có thiết bị
            }
        }, 10000); // 10 giây
    }

    // Hiển thị các thiết bị đã ghép nối
    private void showPairedDevices() {
        if (!hasBluetoothPermissions()) {
            requestBluetoothPermissions();
            return;
        }

        // Kiểm tra quyền đã được cấp cho phép kết nối Bluetooth
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH_CONNECT) != PackageManager.PERMISSION_GRANTED) {
            return;
        }

        // Lấy danh sách các thiết bị đã ghép nối
        Set<BluetoothDevice> pairedDevices =
bluetoothAdapter.getBondedDevices();
        if (pairedDevices.size() > 0) {
            for (BluetoothDevice device : pairedDevices) {
                // Thêm thiết bị vào danh sách và hiển thị lên giao diện
                deviceArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
                deviceList.add(device);
            }
        } else {
            noDeviceText.setVisibility(View.VISIBLE);  // Hiển thị thông báo
nếu không có thiết bị
        }
    }

    // Receiver nhận sự kiện khi có thiết bị Bluetooth mới
    private final BroadcastReceiver receiver = new BroadcastReceiver() {
        @Override
```

```
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                if (device != null && !deviceList.contains(device)) {
                    // Kiểm tra quyền đã được cấp trước khi xử lý thiết bị
                    if (ActivityCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.BLUETOOTH_CONNECT) != PackageManager.PERMISSION_GRANTED) {
                        return;
                    }
                    String deviceName = device.getName() != null ?
device.getName() : "Không xác định được thiết bị";
                    String deviceAddress = device.getAddress();
                    // Thêm thiết bị vào danh sách và cập nhật giao diện
                    deviceArrayAdapter.add(deviceName + "\n" + deviceAddress);
                    deviceList.add(device);
                    deviceArrayAdapter.notifyDataSetChanged();  // Cập nhật
danh sách hiển thị
                }
            }
        }
    };

    // Kết nối đến thiết bị Bluetooth đã chọn
    private void connectToDevice(BluetoothDevice device) {
        new Thread(() -> {
            try {
                // Kiểm tra quyền đã được cấp
                if (!hasBluetoothPermissions()) {
                    requestBluetoothPermissions();
                    return;
                }

                // Kiểm tra quyền kết nối Bluetooth
                if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH_CONNECT) != PackageManager.PERMISSION_GRANTED) {
                    return;
                }

                // Tạo socket kết nối đến thiết bị
                bluetoothSocket =
device.createRfcommSocketToServiceRecord(UUID.fromString("00001101-0000-1000-
8000-00805F9B34FB"));
```

```
                bluetoothAdapter.cancelDiscovery();  // Hủy quá trình quét
                bluetoothSocket.connect();  // Kết nối đến thiết bị

                // Gửi và nhận dữ liệu qua kết nối Bluetooth
                OutputStream outputStream = bluetoothSocket.getOutputStream();
                InputStream inputStream = bluetoothSocket.getInputStream();
                outputStream.write("Hello có nghĩa Xin chào".getBytes());  //
Gửi dữ liệu

                byte[] buffer = new byte[1024];
                int bytes;
                while ((bytes = inputStream.read(buffer)) > 0) {
                    String receivedMessage = new String(buffer, 0, bytes);
                    // Hiển thị thông điệp nhận được từ thiết bị
                    runOnUiThread(() -> Toast.makeText(this, "Đã nhận thông
điệp: " + receivedMessage, Toast.LENGTH_SHORT).show());
                }

                bluetoothSocket.close();  // Đóng kết nối
            } catch (IOException e) {
                e.printStackTrace();
                runOnUiThread(() -> Toast.makeText(this, "Kết nối thất bại",
Toast.LENGTH_SHORT).show());
            }
        }).start();
    }

    // Kiểm tra quyền Bluetooth
    private boolean hasBluetoothPermissions() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
            return ActivityCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH_SCAN) == PackageManager.PERMISSION_GRANTED &&
                    ActivityCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH_CONNECT) == PackageManager.PERMISSION_GRANTED;
        } else {
            return ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED;
        }
    }

    // Yêu cầu quyền Bluetooth
    private void requestBluetoothPermissions() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
            ActivityCompat.requestPermissions(this, new String[]{
```

```
                    Manifest.permission.BLUETOOTH_CONNECT,
                    Manifest.permission.BLUETOOTH_SCAN
            }, 1);
        } else {
            ActivityCompat.requestPermissions(this, new String[]{
                    Manifest.permission.ACCESS_FINE_LOCATION
            }, 1);
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);

        if (requestCode == 1) {
            for (int grantResult : grantResults) {
                if (grantResult != PackageManager.PERMISSION_GRANTED) {
                    Toast.makeText(this, "Quyền bị từ chối. Ứng dụng sẽ không
hoạt động đầy đủ.", Toast.LENGTH_SHORT).show();
                    return;
                }
            }
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        // Đóng kết nối và hủy đăng ký receiver khi ứng dụng bị hủy
        if (receiver != null) {
            unregisterReceiver(receiver);
        }
        try {
            if (bluetoothSocket != null) {
                bluetoothSocket.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```