Mục lục

[Lời mở đầu 2](#_Toc478637664)

[Chương 1: Tìm hiểu về mã vạch (Barcode) 2](#_Toc478637665)

[1.1. Mã vạch là gì? 2](#_Toc478637666)

[1.2. Các loại mã vạch 2](#_Toc478637667)

[1.2.1. UPC (Universal Product Code) 2](#_Toc478637668)

[1.2.2. Code 39 3](#_Toc478637669)

[1.2.3. Code 128 4](#_Toc478637670)

[1.2.4. QR Code 4](#_Toc478637671)

[Chương 2: Lập trình ứng dụng android sử dụng thư viện Zxing để quét mã vạch. 5](#_Toc478637672)

[2.1. Bước 1: Import thư viện Zxing (Zxing-Embedded) vào trong project 5](#_Toc478637673)

[2.2. Bước 2: Thêm các button và editText để ghi lại thông tin 5](#_Toc478637674)

[2.3. Thêm các hàm xử lý sự kiện 10](#_Toc478637675)

[2.4. Kết quả chương trình sau khi quét thành công 13](#_Toc478637676)

# Lời mở đầu

Thông qua bài báo cáo này, em đã hiểu định nghĩa mã vạch, các loại mã vạch phổ biến hiện có, biết và sử dụng thư viện Zxing để tạo ứng dụng có thể đọc các loại mã vạch trên điện thoại.

# Chương 1: Tìm hiểu về mã vạch (Barcode)

## Mã vạch là gì?

Mã vạch (Barcode) theo định nghĩa là phương pháp lưu trữ và truyền tải thông tin bằng một loại ký hiệu gọi là ký mã vạch (Barcode symbology). Ký mã vạch hay được gọi tắt là mã vạch, là một ký hiệu tổ hợp các khoảng trắng và vạch thẳng để biểu diễn các mẫu ký tự, ký hiệu và các con số. Sự thay đổi trong độ rộng của vạch và khoảng trắng biểu diễn thông tin số hay chữ số dưới dạng mà máy có thể đọc được.

Mã số mã vạch được thu nhận bằng một máy quét mã vạch, là một máy thu nhận hình ảnh của mã vạch in trên các bề mặt và chuyển thông tin chứa trong mã vạch đến máy tính hay các thiết bị cần thông tin này. Nó thường có một nguồn sáng kèm theo thấu kính để hội tụ ánh sáng lên mã vạch, rồi thu ánh sáng phản xạ về một cảm quang chuyển hóa tín hiệu ánh sáng thành tín hiệu điện. Ngoài ra, nhiều máy quét mã vạch còn có thêm mạch điện tử xử lý tín hiệu thu được từ cảm quang để chuyển thành tín hiệu phù hợp cho kết nối với máy tính.

## Các loại mã vạch

### UPC (Universal Product Code)

UPC là một loại ký hiệu mã hóa số được ngành công nghiệp thực phẩm ứng dụng vào năm 1973. Ngành công nghiệp thực phẩm đã phát triển hệ thống này nhằm gắn mã số không trùng lặp cho từng sản phẩm. Người ta sử dụng UPC như “giấy phép bằng số” cho các sản phẩm riêng lẻ.

UPC gồm có 2 phần: phần mã vạch mà máy có thể đọc được và phần số mà con người có thể đọc được. Số của UPC gồm 12 ký số, không bao gồm ký tự. Đó là các mã số dùng để nhận diện mỗi một sản phẩm tiêu dùng riêng biệt.

Hình minh họa: 

Nhìn ký hiệu UPC như hình bên ta thấy tổng cộng gồm 12 ký số:  
**Ký số thứ 1**: Ở đây là số 0, gọi là ký số hệ thống số (number system digit) hoặc còn gọi là “Family code”. Nó nằm trong phạm vi của 7 con số định rõ ý nghĩa và chủng loại của sản phẩm như sau:

\* 5 -             Coupons: Phiếu lĩnh hàng hóa  
\* 4 -             Dành cho người bán lẽ sử dụng  
\* 3 -             Thuốc và các mặt hàng có liên quan đến  y tế.  
\* 2 -             Các món hàng nặng tự nhiên như thịt và nông sản.  
\* 0, 6, 7 -     Gán cho tất cả các mặt hàng khác như là một phần nhận diện của nhà sản xuất.

**Năm ký số thứ 2**:  Trong mẫu này, tượng trưng là 12345, ám chỉ mã người bán (Vendor Code), mã doanh nghiệp hay mã của nhà sản xuất (Manufacturer code). Ở Hoa kỳ, mã này được cấp bởi hiệp hội UCC (The Uniform Code Council) và mã được cấp cho người bán hoặc nhà sản xuất là độc nhất. Như vậy khi hàng hóa lưu thông trên thị trường bằng mã UPC thì chỉ cần biết được 5 ký số này là có thể biết được xuất xứ của hàng hóa.

**Năm ký số kế tiếp**:  Dành cho người bán gán cho sản phẩm của họ. Người bán tự tạo ra 5 ký số này theo ý riêng của mình để mã hóa cho sản phẩm .

Ký số cuối cùng: Ở đây là số 5, là ký số kiểm tra, xác nhận tính chính xác của toàn bộ số UPC

UPC được phát triển thành nhiều phiên bản (version) như UPC-A, UPC-B, UPC-C, UPC-D và UPC-E trong đó UPC-A được coi như phiên bản chuẩn của UPC, các phiên bản còn lại được phát triển theo những yêu cầu đặc biệt của ngành công nghiệp.

### Code 39

UPC có tính chất chuyên nghiệp và quốc tế nhưng khuyết điểm của nó là dung lượng có giới hạn và chỉ được mã hóa được số, không mã hóa được chữ.

Code 39 được phát triển sau UPC là ký hiệu chữ và số thông dụng nhất. Nó không có chiều dài cố định như UPC do đó có thể lưu trữ nhiều lượng thông tin hơn bên trong nó. Do tính linh hoạt như vậy, Code 39 được ưu chuộng rộng rãi trong bán lẻ và sản xuất. Bố ký tự này bao gồm tất cả các chữ hóa, các ký số từ 0 đến 9 và 7 ký tự đặc biệt khác.

Hình minh họa: 

### Code 128

Code 128 hay được biết đến với tên đầy đủ là “Full ASCII-Character set with Code128 ABC autoselection” được giới thiệu lần đầu vào năm 1981 như một loại mã hóa ký số ASCII với mật độ cao.

Code 128 là tập gồm các ký tự 0 đến 9, ký tự a-z (hoa và thường) và tất cả các ký tự biểu tượng chuẩn ASCII và cả mã điều khiển. Và được chia thành 3 loại A,B, C:

* Code 128A bao gồm các ký tự chuẩn ASCII, số, chữ hoa, chữ thường và mã điều khiển.
* Code 128B bao gồm các ký tự chuẩn ASCII, số, chữ (hoa và thường).
* Code 128C nén 2 ký số trong một ký tự mã hóa, cung cấp một dạng mã hóa nén tốt nhất.

Hình minh họa: 

### QR Code

QR Code được phát triển bởi công ty Denso Wave (Nhật Bản) vào năm 1994, viết tắt của Quick response code hay còn gọi là mã vạch ma trận (matrix-barcode) là dạng mã vạch hai chiều (2D) có thể được đọc bởi một máy đọc mã vạch hay điện thoại có chức năng chụp ảnh với ứng dụng đọc mã vạch.

Hình minh họa: 

# Chương 2: Lập trình ứng dụng android sử dụng thư viện Zxing để quét mã vạch.

## 2.1. Bước 1: Import thư viện Zxing (Zxing-Embedded) vào trong project

Đầu tiên ta tải thư viện Zxing-Embedded tại link (<https://github.com/journeyapps/zxing-android-embedded>), sau đó Import Module có tên zxing-android-embedded. Thêm vào file build.gradle như sau:

***Đoạn mã 2.1.*** Mô tả các câu lệnh thêm vào trong file build.gradle để chương trình có thể nhận thư viện Zxing

repositories {

jcenter()

}

dependencies {

compile 'com.journeyapps:zxing-android-embedded:3.5.0'

compile 'com.android.support:appcompat-v7:25.1.0' // Version 23+ is required

}

android {

buildToolsVersion '25.0.2' // Older versions may give compile errors

}

## 2.2. Bước 2: Thêm các button và editText để ghi lại thông tin

- Đầu tiên ta tạo một LinearLayout và định nghĩa theo chiều dọc (vertical).

- Sau đó thêm một LinearLayout và thêm một TextView để lưu định dạng của mã vạch mà ta đã quét được.

- Thêm tiếp một LinearLayout-Horizontal trong đó thêm một button để nhấn thực hiện quét và một EditText để lưu kết quả đã quét được hoặc cũng có thể tự điền kết quả vào đó.

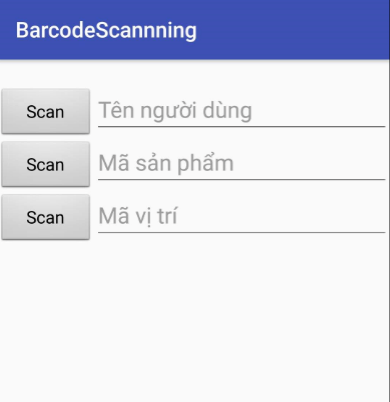
- Thêm tiếp 2 LinearLayout-Horizontal và làm tương tự như trên.

***Đoạn mã 2.2.*** File xml sau khi thêm các button và editText

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/activity\_main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context="com.example.cong.barcodescannning.MainActivity">  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">

<TextView  
 android:id="@+id/scan\_format"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/scan\_button"  
 android:layout\_centerHorizontal="true"  
 android:layout\_weight="1"  
 android:textIsSelectable="true" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/scan\_button\_name"  
 style="@android:style/Widget.Button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Scan" />  
  
 <EditText  
 android:id="@+id/scan\_content\_name"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/scan\_format"  
 android:layout\_weight="1"  
 android:hint="Tên người dùng"  
 android:inputType="textPersonName" />  
  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/scan\_button\_id"  
 style="@android:style/Widget.Button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Scan" />  
  
 <EditText  
 android:id="@+id/scan\_content\_id"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/scan\_format"  
  
 android:layout\_weight="1"  
 android:hint="Mã sản phẩm"  
 android:inputType="text" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/scan\_button\_locate"  
 style="@android:style/Widget.Button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Scan" />  
  
 <EditText  
 android:id="@+id/scan\_content\_locate"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/scan\_format"  
  
 android:layout\_weight="1"  
 android:hint="Mã vị trí"  
 android:inputType="text" />  
 </LinearLayout>  
  
</LinearLayout>

Hình ảnh giao diện sau khi tạo xong file xml:



- Trong file java chính, ta tạo các Button, TextView, EditText

***Đoạn mã 2.3.*** Mô tả việc định nghĩa các Button, TextView, EditText

private Button scanBtnName, scanBtnId, scanBtnLocate; //(1)  
private TextView formatTxt; //(2)  
private EditText contentTxtName,contentTxtId, contentTxtLocate; //(3)  
int check=0; //(4)

Trong đoạn mã 2.3:

* (1) định nghĩa tên các nút bấm tương ứng.
* (2) định nghĩa tên của TextView để ghi lại kết quả format sau khi quét.
* (3) định nghĩa tên của EditText tương ứng, để ghi lại nội dung sau khi quét.
* (4) biến kiểm tra nút bấm nào đã được bấm để ghi lại nội dung vào EditText.

- Sau đó trong hàm onCreate gán các giá trị vừa mới định nghĩa với file xml

***Đoạn mã 2.4.*** Hàm onCreate sau khi gán giá trị các button và editText với file xml

public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main); //(1)  
  
 scanBtnName = (Button)findViewById(R.id.scan\_button\_name); //(2)  
 scanBtnId = (Button)findViewById(R.id.scan\_button\_id);//(3)   
 scanBtnLocate = (Button)findViewById(R.id.scan\_button\_locate);  
 formatTxt = (TextView)findViewById(R.id.scan\_format);//(4)  
 contentTxtName = (EditText) findViewById(R.id.scan\_content\_name); //(5)  
 contentTxtId = (EditText) findViewById(R.id.scan\_content\_id); //(6)  
 contentTxtLocate = (EditText) findViewById(R.id.scan\_content\_locate); //(7)  
  
 scanBtnName.setOnClickListener(onScanName); //(8)  
 scanBtnId.setOnClickListener(onScanId); //(9)  
 scanBtnLocate.setOnClickListener(onScanLocate); //(10)  
}

Trong đoạn mã 2.4:

* (1) đưa ra nội dung xuất hiện trên màn hình chính là activity\_main.xml
* (2), (3), (4), (5), (6), (7) Gọi đến hàm findViewById gán các đối tượng tương ứng với id đã được khai báo trong xml.
* (8), (9), (10) gán các đối tượng tương ứng vào các hàm xử lý sự kiện.

## 2.3. Thêm các hàm xử lý sự kiện

- Đầu tiên ta import 2 lớp IntentIntegrator và IntentResult đã có sẵn trong thư viện Zxing để có thể thực hiện quét và ghi kết quả sau khi đọc mã vạch.

***Đoạn mã 2.5.*** Import 2 lớp IntentIntegrator và IntentResult

import com.google.zxing.integration.android.IntentIntegrator;  
import com.google.zxing.integration.android.IntentResult;

***Đoạn mã 2.6.*** Hàm xử lý sự kiện onScanName

private View.OnClickListener onScanName=new View.OnClickListener(){ //(1)  
 public void onClick(View v){  
 if(v.getId()==R.id.scan\_button\_name){ //(2)  
 IntentIntegrator scanIntegrator= new IntentIntegrator(MainActivity.this); //(3)  
 scanIntegrator.setBeepEnabled(false); //(4)  
 check=1; //(5)  
 scanIntegrator.initiateScan(); //(6)  
 }  
 }  
};

Trong đoạn mã 2.6:

* (1) tạo đối tượng View.OnClickListener để “lắng nghe” sự kiện OnClickListener có được nhấn hay không, nếu được nhấn sẽ thực hiện hàm onClick().
* (2) kiểm tra lại xem nút bấm đó là nút nào.
* (3) tạo đối tượng IntentIntegrator mới. Khi nhấn nút scan thì màn hình sẽ chuyển từ màn hình chính sang màn hình quét.
* (4) đặt giá trị false để khi quét không phát ra tiếng beep.
* (5) gán giá trị check = 1 để biết đây là nút ScanName
* (6) sau khi chuyển sang màn hình quét thì ta bắt đầu thực hiện quét.

- Sau đó thêm tiếp 2 hàm xử lý sự kiện tương tự như onScanName

***Đoạn mã 2.7.*** Hàm xử lý sự kiện onScanId

private View.OnClickListener onScanId=new View.OnClickListener(){  
 public void onClick(View v){  
 if(v.getId()==R.id.scan\_button\_id){  
 IntentIntegrator scanIntegrator= new IntentIntegrator(MainActivity.this);  
 scanIntegrator.setBeepEnabled(false);  
 check=2;  
 scanIntegrator.initiateScan();  
 }  
 }  
};

***Đoạn mã 2.8.*** Hàm xử lý sự kiện onScanLocate

private View.OnClickListener onScanLocate=new View.OnClickListener(){  
 public void onClick(View v){  
 if(v.getId()==R.id.scan\_button\_locate){  
 IntentIntegrator scanIntegrator= new IntentIntegrator(MainActivity.this);  
 scanIntegrator.setBeepEnabled(false);  
 check=3;  
 scanIntegrator.initiateScan();  
 }  
 }  
};

***Đoạn mã 2.9.*** Hàm onActivityResult() để trả về kết quả sau khi quét

public void onActivityResult(int requestCode,int resultCode, Intent intent){  
 IntentResult scaningResult = IntentIntegrator.parseActivityResult(requestCode, resultCode, intent); //(1)  
 if(scaningResult != null){ //(2)  
 String scanContent = scaningResult.getContents();  
 String scanFormat = scaningResult.getFormatName();  
 formatTxt.setText("FORMAT: " + scanFormat);  
 if(check==1) {  
 contentTxtName.setText(scanContent); //(3)  
 }  
 else if(check==2) {  
 contentTxtId.setText(scanContent); //(4)  
 }  
 else if(check==3) {  
 contentTxtLocate.setText(scanContent); //(5)  
 }  
 }  
 else  
 Toast.makeText(getApplicationContext(),"No data received",Toast.LENGTH\_SHORT).show(); //(6)  
}

Trong đoạn mã 2.9:

* (1) hàm parseActivityResult được gọi trong activity onActivityResult sẽ trả về kết quả là định dạng dữ liệu được quét và nội dung của nó, nếu người dùng hủy việc quét thì nó sẽ trả về giá trị null cho tất cả các trường.
* (2) nếu quét thành công nó sẽ trả về kết quả là định dạng dữ liệu và nội dung được ghi vào editText.
* (6) nếu quét không thành công thì chương trình sẽ hiện lên thông báo với nội dung là “No data received”.
* (3) nếu người dùng ấn vào nút scan ở dòng Tên người dùng thì nó sẽ trả giá trị vào đúng dòng đấy.
* (4), (5) tương tự như (3), nó sẽ trả giá trị vào đúng editText ở dòng đó.

## 2.4. Kết quả chương trình sau khi quét thành công

Hình ảnh giao diện sau khi quét thành công