

PROGRAMACIÓN DEL HARDWARE EN ANDROID

Carlos Maldonado

5IV7

1. Introducción

La programación de la cámara en dispositivos Android ha evolucionado significativamente desde los primeros días del sistema operativo. Inicialmente, el acceso a la cámara era complejo y propenso a errores, pero con el tiempo, Google ha introducido APIs más robustas y fáciles de usar. Hoy en día, los desarrolladores tienen múltiples opciones para integrar funcionalidades de cámara en sus aplicaciones, desde la API Camera original hasta la moderna CameraX.

La importancia de implementar correctamente las funciones de cámara radica en que los usuarios esperan experiencias fluidas y de alta calidad, similares a las aplicaciones de cámara nativas. Una mala implementación puede resultar en aplicaciones lentas, que consumen mucha batería o que incluso fallan en diferentes dispositivos.

2. Arquitectura del Sistema de Cámara en Android

El sistema de cámara en Android está compuesto por varias capas:

2.1 Capa de Hardware

- **Controladores de cámara:** Software de bajo nivel que comunica directamente con el hardware
- **Sensores de imagen:** CMOS o CCD que capturan la luz
- **Procesador de señales de imagen (ISP):** Procesa los datos brutos del sensor

2.2 Capa de HAL (Hardware Abstraction Layer)

- Proporciona una interfaz estandarizada entre el framework de Android y los controladores de hardware
- Define métodos estándar que los fabricantes deben implementar

2.3 Framework de Android

- **Camera API (`android.hardware.Camera`):** API original (deprecada en API 21)
- **Camera2 API (`android.hardware.camera2`):** Introducida en Android 5.0 (API 21)
- **CameraX:** Biblioteca Jetpack lanzada en 2019

2.4 Aplicaciones

- Utilizan las APIs proporcionadas por el framework
- Pueden implementar procesamiento adicional de imagen/video

3. Permisos y Configuración Inicial

3.1 Permisos en AndroidManifest.xml

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />

<!-- Para guardar imágenes -->
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    android:maxSdkVersion="28" />
<uses-permission
    android:name="android.permission.READ_EXTERNAL_STORAGE" />

<!-- Para Android 10+ -->
<uses-permission
    android:name="android.permission.ACCESS_MEDIA_LOCATION" />
```

3.2 Solicitud de permisos en tiempo de ejecución

En Android 6.0+ (API 23), los permisos peligrosos deben solicitarse en tiempo de ejecución:

```
private static final int REQUEST_CAMERA_PERMISSION = 100;

private void checkCameraPermission() {
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
        != PackageManager.PERMISSION_GRANTED) {

        ActivityCompat.requestPermissions(this,
            new String[]{Manifest.permission.CAMERA},
            REQUEST_CAMERA_PERMISSION);
    } else {
        // Permiso ya concedido, inicializar cámara
    }
}
```

```
    initCamera();
}
}

@Override
public void onRequestPermissionsResult(int requestCode,
        @NonNull String[] permissions,
        @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == REQUEST_CAMERA_PERMISSION) {
        if (grantResults.length > 0 &&
            grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            initCamera();
        } else {
            // Mostrar mensaje explicando por qué se necesita el permiso
        }
    }
}
```

4. APIs de Cámara Disponibles

4.1 Camera API (Deprecada)

- Disponible desde Android 1.0
- Sencilla pero limitada
- No compatible con características avanzadas
- Problemas de compatibilidad entre dispositivos

4.2 Camera2 API

- Introducida en Android 5.0 (API 21)
- Control granular sobre el hardware
- Soporte para características avanzadas:
 - Control manual de exposición
 - RAW capture
 - Burst mode
 - Procesamiento posterior

Limitaciones:

- Código más complejo
- Diferencias de implementación entre fabricantes
- Requiere manejo explícito del estado de la cámara

4.3 CameraX (Recomendada)

- Parte de Android Jetpack
- Compatible con Android 5.0+ (API 21+)
- Usa Camera2 internamente
- Ventajas:
 - Menos código boilerplate
 - Compatibilidad entre dispositivos
 - Ciclo de vida automático
 - Extensiones opcionales (HDR, Night mode, etc.)

5. CameraX: La API Moderna

5.1 Arquitectura de CameraX

CameraX sigue un modelo basado en casos de uso (use cases):

1. **Preview:** Vista previa en tiempo real
2. **Image Analysis:** Análisis de frames para ML o procesamiento
3. **Image Capture:** Captura de imágenes de alta calidad
4. **Video Capture:** Captura de video (agregado posteriormente)

5.2 Componentes principales

- **ProcessCameraProvider:** Punto de entrada principal
- **CameraSelector:** Selecciona cámara frontal/trasera
- **PreviewView:** Surface para mostrar vista previa
- **ImageCapture:** Para capturar fotos
- **ImageAnalysis:** Para analizar frames

5.3 Ciclo de vida

CameraX se integra automáticamente con el ciclo de vida de Android, liberando recursos cuando la cámara no está en uso.

6. Ejemplo Práctico con CameraX

A continuación se presenta un ejemplo completo de una aplicación de cámara básica utilizando CameraX:

6.1 Configuración de dependencias

En el archivo build.gradle del módulo:

```
dependencies {
    def camerax_version = "1.3.0"

    // Implementación básica de CameraX
    implementation "androidx.camera:camera-core:${camerax_version}"
    implementation "androidx.camera:camera-camera2:${camerax_version}"
    implementation "androidx.camera:camera-lifecycle:${camerax_version}"
    implementation "androidx.camera:camera-view:${camerax_version}"

    // Para captura de imágenes
    implementation "androidx.camera:camera-extensions:${camerax_version}"
}
```

6.2 Layout XML (activity_main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.camera.view.PreviewView
        android:id="@+id/previewView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="32dp"
        android:orientation="horizontal"
```

```
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent">

    <ImageButton
        android:id="@+id/btnSwitchCamera"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:layout_marginEnd="32dp"
        android:background="@android:color/transparent"
        android:src="@android:drawable/ic_menu_camera" />

    <ImageButton
        android:id="@+id/btnCapture"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:background="@android:color/transparent"
        android:src="@drawable/ic_camera" />

    <ImageButton
        android:id="@+id/btnFlash"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:layout_marginStart="32dp"
        android:background="@android:color/transparent"
        android:src="@android:drawable/ic_menu_edit" />

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

6.3 Clase MainActivity.java

```
package com.example.cameraapp;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.camera.core.CameraSelector;
import androidx.camera.core.ImageCapture;
import androidx.camera.core.ImageCaptureException;
import androidx.camera.core.Preview;
import androidx.camera.lifecycle.ProcessCameraProvider;
import androidx.camera.view.PreviewView;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.lifecycle.LifecycleOwner;

import android.Manifest;
import android.content.ContentValues;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.view.View;
import android.widget.ImageButton;
import android.widget.Toast;

import com.google.common.util.concurrent.ListenableFuture;

import java.text.SimpleDateFormat;
import java.util.Locale;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class MainActivity extends AppCompatActivity {
    private static final String TAG = "CameraApp";
```

```
private static final int REQUEST_CAMERA_PERMISSION = 101;

private ExecutorService cameraExecutor;
private PreviewView previewView;
private ImageButton btnCapture, btnSwitchCamera, btnFlash;
private ImageCapture imageCapture;
private int cameraFacing = CameraSelector.LENS_FACING_BACK;

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

    // Inicializar vistas
    previewView = findViewById(R.id.previewView);
    btnCapture = findViewById(R.id.btnCapture);
    btnSwitchCamera = findViewById(R.id.btnSwitchCamera);
    btnFlash = findViewById(R.id.btnFlash);

    // Solicitar permisos
    if (allPermissionsGranted()) {
        startCamera();
    } else {
        ActivityCompat.requestPermissions(this,
            new String[]{Manifest.permission.CAMERA},
            REQUEST_CAMERA_PERMISSION);
    }

    // Configurar botones
    btnCapture.setOnClickListener(v -> takePhoto());
    btnSwitchCamera.setOnClickListener(v -> switchCamera());

    // Crear executor para operaciones de cámara
    cameraExecutor = Executors.newSingleThreadExecutor();
}

private boolean allPermissionsGranted() {
    return ContextCompat.checkSelfPermission(
```

```
        this,                         Manifest.permission.CAMERA) ==  
PackageManager.PERMISSION_GRANTED;  
    }  
  
    @Override  
    public void onRequestPermissionsResult(int requestCode,  
                                           @NonNull String[] permissions,  
                                           @NonNull int[] grantResults) {  
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
        if (requestCode == REQUEST_CAMERA_PERMISSION) {  
            if (grantResults.length > 0 &&  
                grantResults[0] == PackageManager.PERMISSION_GRANTED) {  
                startCamera();  
            } else {  
                Toast.makeText(this,  
                           "Permiso de cámara denegado",  
                           Toast.LENGTH_SHORT).show();  
                finish();  
            }  
        }  
    }  
}  
  
private void startCamera() {  
    ListenableFuture<ProcessCameraProvider> cameraProviderFuture =  
        ProcessCameraProvider.getInstance(this);  
  
    cameraProviderFuture.addListener(() -> {  
        try {  
            ProcessCameraProvider cameraProvider = cameraProviderFuture.get();  
  
            // Configurar Preview  
            Preview preview = new Preview.Builder().build();  
            preview.setSurfaceProvider(previewView.getSurfaceProvider());  
  
            // Configurar ImageCapture  
            imageCapture = new ImageCapture.Builder()  
                .setCaptureMode(ImageCapture.CAPTURE_MODE_MINIMIZE_LATENCY)  
                .build();  
        } catch (ExecutionException | InterruptedException e) {  
            e.printStackTrace();  
        }  
    }, ContextCompat.getMainExecutor(this));  
}
```

```
// Seleccionar cámara
CameraSelector cameraSelector = new CameraSelector.Builder()
    .requireLensFacing(cameraFacing)
    .build();

// Limpiar casos de uso previos
cameraProvider.unbindAll();

// Vincular casos de uso al ciclo de vida
cameraProvider.bindToLifecycle(
    (LifecycleOwner) this,
    cameraSelector,
    preview,
    imageCapture);

} catch (ExecutionException | InterruptedException e) {
    Log.e(TAG, "Error al inicializar cámara", e);
}
}, ContextCompat.getMainExecutor(this));
}

private void takePhoto() {
    if (imageCapture == null) {
        Toast.makeText(this, "Cámara no inicializada", Toast.LENGTH_SHORT).show();
        return;
    }

    // Configurar metadata de la imagen
    String name = new SimpleDateFormat("yyyy-MM-dd-HH-mm-ss-SSS",
        Locale.getDefault()).format(System.currentTimeMillis());

    ContentValues contentValues = new ContentValues();
    contentValues.put(MediaStore.MediaColumns.DISPLAY_NAME, name);
    contentValues.put(MediaStore.MediaColumns.MIME_TYPE, "image/jpeg");

    if (Build.VERSION.SDK_INT > Build.VERSION_CODES.P) {
```

```
        contentValues.put(MediaStore.Images.Media.RELATIVE_PATH,
"Pictures/CameraApp");
    }

    // Crear opciones de salida
    ImageCapture.OutputFileOptions outputFileOptions =
        new ImageCapture.OutputFileOptions.Builder(
            getContentResolver(),
            MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
            contentValues
        ).build();

    // Tomar foto
    imageCapture.takePicture(
        outputFileOptions,
        ContextCompat.getMainExecutor(this),
        new ImageCapture.OnImageSavedCallback() {
            @Override
            public void onImageSaved(
                @NonNull ImageCapture.OutputFileResults outputFileResults) {
                String msg = "Foto guardada: " + outputFileResults.getSavedUri();
                Toast.makeText(MainActivity.this, msg, Toast.LENGTH_SHORT).show();
                Log.d(TAG, msg);
            }
        }

        @Override
        public void onError(@NonNull ImageCaptureException exception) {
            Log.e(TAG, "Error al guardar foto: " + exception.getMessage(), exception);
            Toast.makeText(MainActivity.this,
                "Error al guardar foto",
                Toast.LENGTH_SHORT).show();
        }
    );
}

private void switchCamera() {
    cameraFacing = (cameraFacing == CameraSelector.LENS_FACING_BACK)
```

```
? CameraSelector.LENS_FACING_FRONT  
: CameraSelector.LENS_FACING_BACK;  
startCamera();  
}  
  
@Override
```

```
protected void onDestroy() {  
    super.onDestroy();  
    if (cameraExecutor != null) {  
        cameraExecutor.shutdown();  
    }  
}
```

7. Consideraciones de Rendimiento

7.1 Uso de memoria

- Las imágenes de cámara son grandes (usualmente 8-12 MP)
- Utilizar resoluciones apropiadas para el caso de uso
- Liberar recursos cuando no sean necesarios

7.2 Consumo de batería

- La cámara es uno de los componentes que más energía consume
- Liberar la cámara cuando la aplicación esté en segundo plano
- Utilizar modos de baja potencia cuando sea posible

7.3 Compatibilidad entre dispositivos

- Probar en múltiples dispositivos y versiones de Android
- Manejar excepciones específicas de fabricantes
- Utilizar CameraX para mayor compatibilidad

7.4 Procesamiento en segundo plano

- Utilizar ExecutorService para operaciones intensivas
- No bloquear el hilo principal
- Mostrar indicadores de progreso para operaciones largas

8. Buenas Prácticas

8.1 Manejo de errores

- Verificar disponibilidad de hardware
- Manejar excepciones de permisos
- Proporcionar retroalimentación al usuario

8.2 Experiencia de usuario

- Mantener la interfaz responsiva
- Proporcionar retroalimentación visual (animaciones)
- Manejar cambios de orientación correctamente

8.3 Seguridad y privacidad

- Solicitar permisos solo cuando sean necesarios
- Explicar por qué se necesitan los permisos
- No almacenar imágenes sin consentimiento

8.4 Optimizaciones

- Utilizar resoluciones apropiadas
- Implementar caché para imágenes procesadas
- Reutilizar objetos cuando sea posible

9. Conclusión

La programación de la cámara en Android ha evolucionado hacia APIs más simples y potentes. CameraX representa el estado del arte actual, proporcionando una abstracción de alto nivel sobre Camera2 mientras mantiene compatibilidad con versiones anteriores de Android.

El ejemplo presentado muestra cómo implementar una aplicación básica de cámara con las funciones esenciales. Para aplicaciones más avanzadas, los desarrolladores pueden explorar características adicionales como:

- Modo nocturno automático
- Detección de rostros y sonrisas
- Filtros y efectos en tiempo real
- Captura de video con estabilización
- Procesamiento con ML Kit

Bibliografía

- **Vogella GmbH — Android Camera API: Tutorial** — Explica dos formas de usar la cámara en Android: (1) invocando la app de cámara mediante un *intent*, o (2) usando directamente la API Camera para integrar la cámara en tu app. Contiene ejemplos en Java. [Vogella](#)
- **Google Developers — Documentación de cámara en Android (Camera API)** — Describe permisos necesarios, cómo abrir la cámara con Camera.open(), cómo ajustar parámetros (foco automático, tamaño, etc.), y advertencias de seguridad / uso. [Android Developers+1](#)
- **AbhiAndroid — “Camera Tutorial With Example In Android Studio”** — Un tutorial paso a paso sobre cómo capturar fotos desde una app Android, usando tanto el “intent” de cámara como la API directa. [Abhi Android](#)
- **TheDeveloperBlog.com — Android Camera Tutorial** — Describe las opciones de usar cámara mediante *intent* o mediante API, y comenta los componentes básicos (Camera, SurfaceView, MediaRecorder) necesarios para video/foto. [El Blog del Desarrollador+1](#)
- **OpenCV (con Java / C++ en Android)** — Aunque es más avanzada, permite procesar video/imágenes capturadas, aplicar filtros, detección, etc. Un proyecto con OpenCV + Android puede ilustrar usos más allá de solo tomar fotos. [Riunet+1](#)