http://www.qrcode.com/en/

QR code: Quick Response Code

- Invented in 1994 by a Japanese company Denso Wave for point of sales and also to track vehicle parts during manufacture (Denso Wave was a subsidiary of Toyota)
- It's license-free
- It's a 2D barcode
- QR codes can be read in any orientation
- The official specification for QR code is ISO/IEC 18004

QR code is capable of encoding the same amount of data in approximately one-tenth the space of a traditional barcode. Can even use Micro QR code to achieve an even smaller printout size.

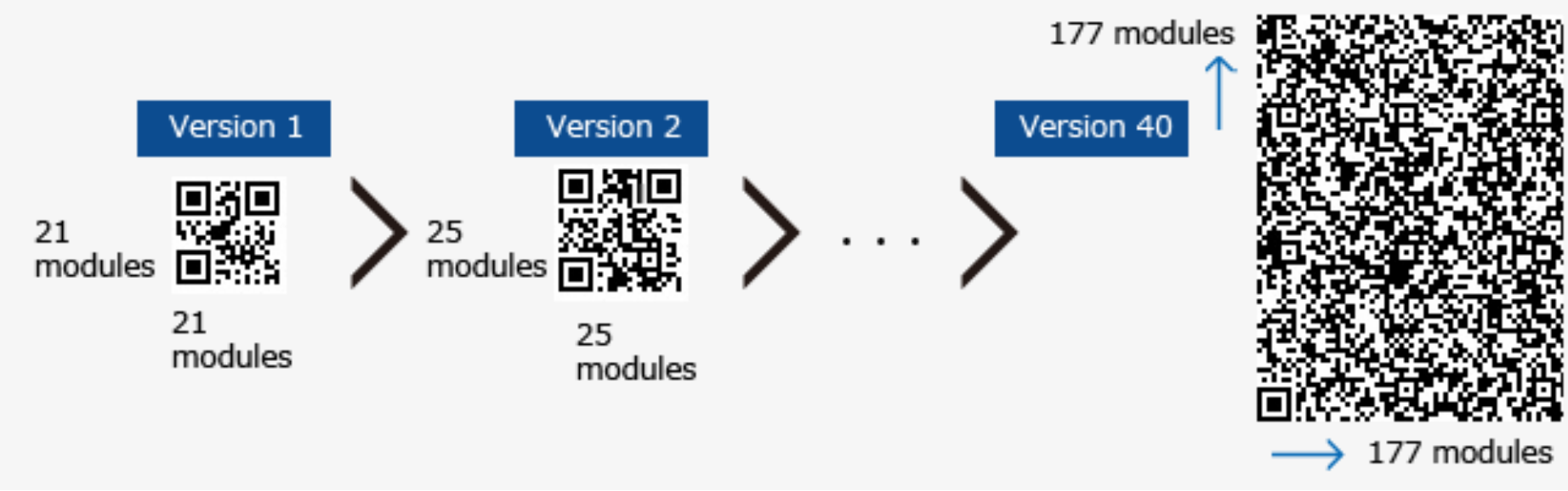
4 standardized encoding modes:

- Numeric
- Alphanumeric
- binary
- Kanji, katakana, hiragana

QR コードは漢字・かなを効率良く
表現することができます。

Storage:

- version (1, …, 40) indicates the dimension of symbol
- Each higher version number comprises 4 additional modules per side
- Example: 40-L means version 40, error correction level L
- Examples: version-1 21x21
  version-2 25x25
  version-40 177x177

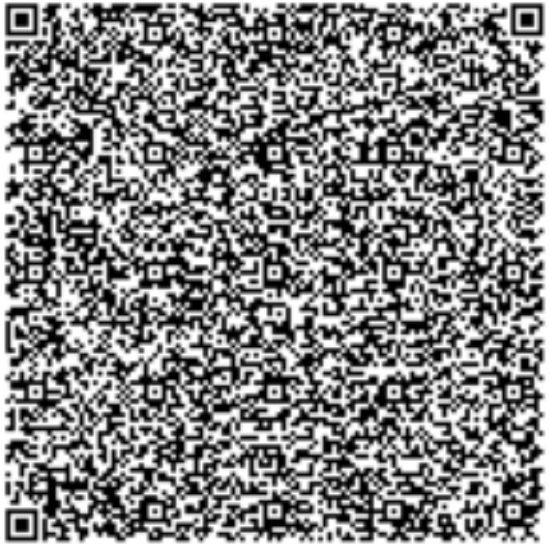


http://www.qrcode.com/en/about/version.html

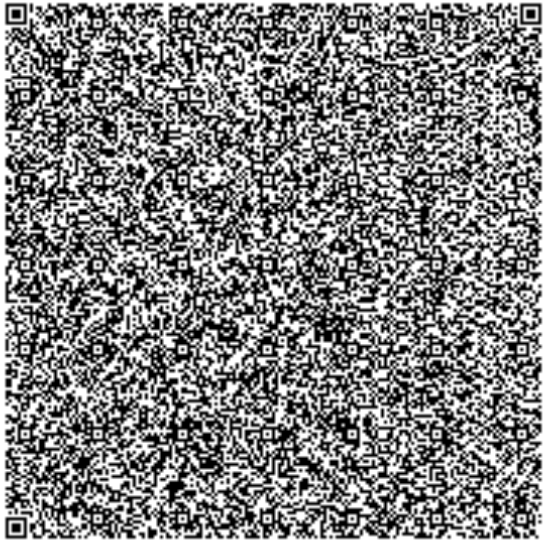
Version 1

Version 3

Version 10



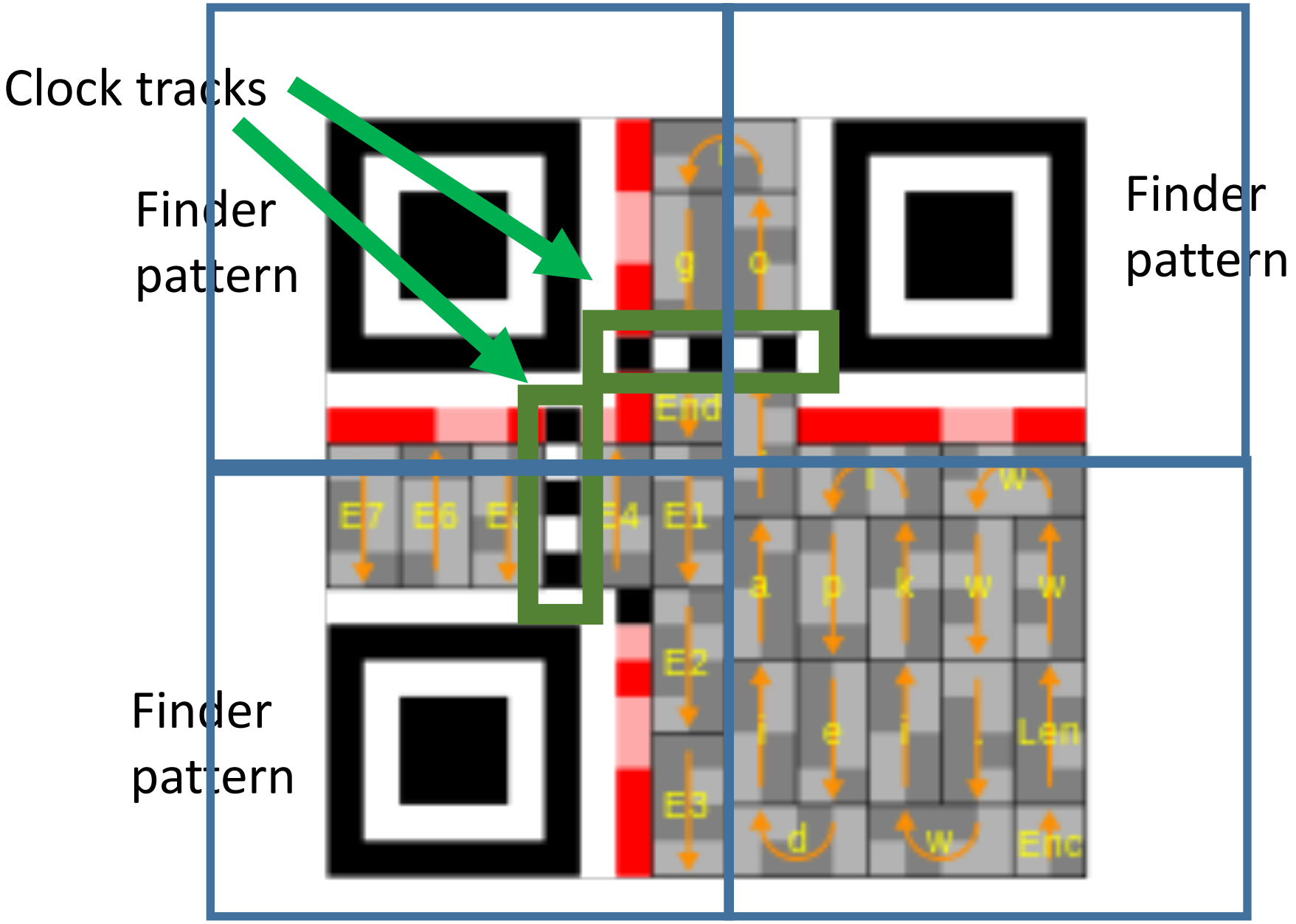
Version 25



Version 40

Example of maximum character storage capacity (40-L)

Data Mode	Maximum # of characters
Numeric	7089
Alphanumeric	4296
Binary	2953
Kanji/kana	1817


Clock tracks
Finder pattern
Finder pattern
Finder pattern
End
E7
E6
E4
E1
E2
E3
a
p
k
W
W
i
e
i
Len
d
W
Enc

Processing:

- Locates the 3 distinctive squares at the corners of the QR code image
- Small dots carry the data and are converted to binary numbers. Error-correcting algorithm is applied in the conversion process.

Error Correction

- Codewords are 8 bits long
- Use Reed-Solomon error correction
- 4 error correction levels:
  - Level L can tolerate up to 7% errors
  - Level M can tolerate up to 15% errors
  - Level Q can tolerate up to 25% errors
  - Level H can tolerate up to 30% errors

More storage capacity

Less storage capacity

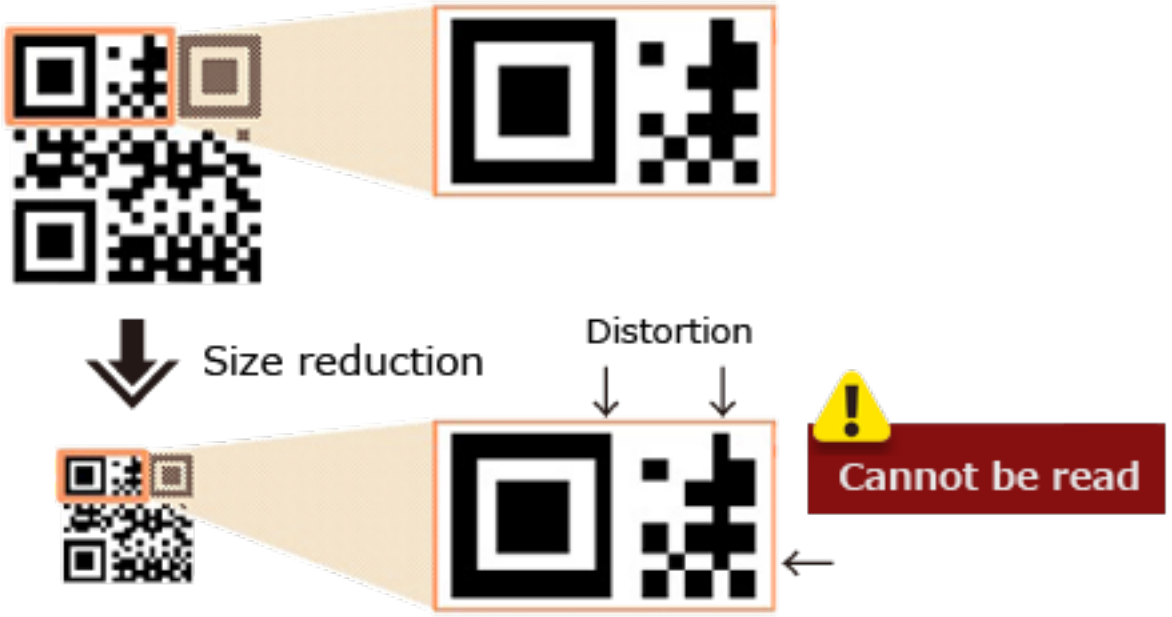
Points to note when generating a QR code:

- Module size
  - A module is a small black square
  - Depends on Resolution of printer and reading device

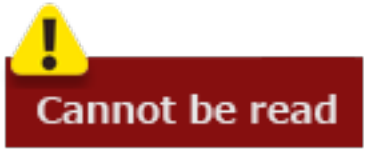
- A margin is required around a QR code.

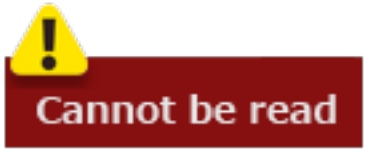
- The area of a QR code is determined by its version, the module size, and the margin size

Codes that are invalid and so cannot be read as QR code:

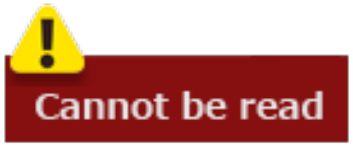


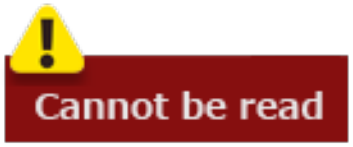
http://www.qrcode.com/en/howto/trouble.html





http://www.qrcode.com/en/howto/trouble.html





Ways of using QR Code

- Traceability (eg. of raw materials of a product)
- Picking
- Inventory Management
- Inspection
- Process Management
- Production Management
- Data Entry
- Dispensing
- Admission Control

Other types of 2D codes

Aztec Code (supported by Android)

Datamatrix (supported by Android)

Microsoft Tag

Can Generate QR Code from the following site:

http://www.qr-code-generator.com/#info_dynamisch

```
package sg.edu.nus.ngtk.myweekendqrcode;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {

    public Button button_qr_code_from_file;
    public Button button_qr_code_from_camera;

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

```
    button_qr_code_from_file =
findViewById(R.id.button_qr_code_from_file);

    button_qr_code_from_file.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent myIntent = new Intent(getApplicationContext(),
ActivityQRFromFile.class);
            startActivity(myIntent);
        }
    });
```

```
        button_qr_code_from_camera =
findViewById(R.id.button_qr_code_from_camera);

        button_qr_code_from_camera.setOnClickListener(new
View.OnClickListener(){
            @Override
            public void onClick(View view) {
                Intent myIntent = new Intent(getApplicationContext(),
ActivityQRFromCamera.class);
                startActivity(myIntent);
            }
        });

    }

}
```

Reading QR Code From File

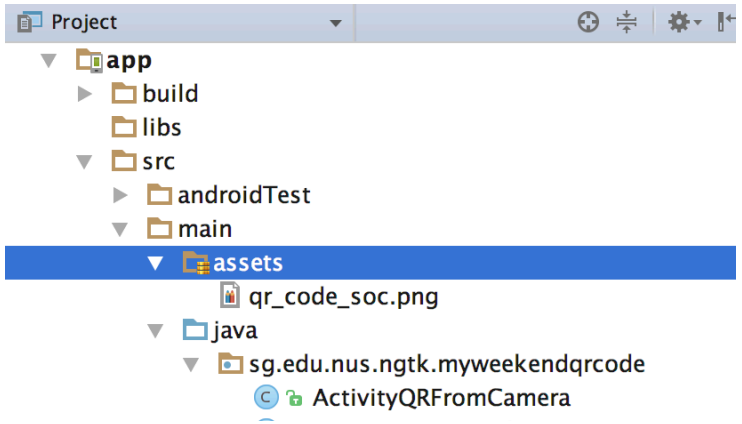
In build.gradle, add the statement in red below:

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.4.0'

compile 'com.google.android.gms:play-services-vision:7.8.0’

}
```

ActivityQRFromFile.java

```
package sg.edu.nus.ngtk.myweekendqrcode;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.util.Log;
import android.util.SparseArray;
import android.widget.Toast;

import com.google.android.gms.vision.Frame;
import com.google.android.gms.vision.barcode.Barcode;
import com.google.android.gms.vision.barcode.BarcodeDetector;

import java.io.IOException;

public class ActivityQRFromFile extends Activity {
```

```
@Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_qrfrom_file);
     Bitmap myQRCode = null;
     try {
        myQRCode =
BitmapFactory.decodeStream(getAssets().open("qr_code_soc.png"));
        Toast.makeText(this, "successfully open qr_code_soc.png",
              Toast.LENGTH_LONG).show();
     } catch (IOException e) {
        Toast.makeText(this, "cannot open QRCodeExample.png",
Toast.LENGTH_LONG).show();
     }
```



```
        BarcodeDetector barcodeDetector =
                new BarcodeDetector.Builder(this)
                .setBarcodeFormats(Barcode.QR_CODE).build();
        Frame myFrame = new
Frame.Builder().setBitmap(myQRCode).build();
        SparseArray<Barcode> barcodes =
barcodeDetector.detect(myFrame);

        if (barcodes.size() != 0) {
            Toast.makeText(this, "My QR Code Data is " +
barcodes.valueAt(0).displayValue,
                    Toast.LENGTH_LONG).show();
            Log.d("My QR Code Data ", barcodes.valueAt(0).displayValue);
        } else {
            Toast.makeText(this, "MyQRCode = " +
myQRCode.describeContents(), Toast.LENGTH_LONG).show();
        }
    }
}
```

Reading QR Code From Camera

In build.gradle, add the statement in red below:

```
dependencies {

    .....
    .....

    compile 'com.google.android.gms:play-services-vision:7.8.0'
    compile 'com.android.support:appcompat-v7:26.+'
}
```

In AndroidManifest.xml, add the following

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

```
<meta-data
android:name="com.google.android.gms.vision.DEPENDENCIES"
   android:value="barcode" />
```

To use Camera to scan QR Code, we will need to ask for permission to use camera. Permission to use camera is a dangerous permission. So we need to do the proper set up to ask the users.

- Add the following permission statement in AndroidManifest.xml

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

<uses-feature android:name="android.hardware.camera" />

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

In the activity that requires the permission, add the codes regarding permissions as in the lecture on Location, to request for permission during run time.

The completed program together with the QR code camera scanning is as shown in the next few slides:

ActivityQRFromCamera.java

```
public class ActivityQRFromCamera extends Activity {

    private static final int REQUEST_CAMERA_PERMISSION = 100;

    private boolean permissionToUseCameraAccepted = false;

    private String[] permissions = {Manifest.permission.CAMERA};


    SurfaceView cameraView;
    TextView tvCodeInfo;

    BarcodeDetector barcodeDetector;
    CameraSource cameraSource;
```

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


      if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.CAMERA)
              != PackageManager.PERMISSION_GRANTED) {

          ActivityCompat.requestPermissions(this, permissions,
REQUEST_CAMERA_PERMISSION);

          //ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.CAMERA}, REQUEST_CAMERA_PERMISSION);
      }
```

```
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.CAMERA)
                == PackageManager.PERMISSION_DENIED) {
            ActivityCompat.requestPermissions(this, permissions,
REQUEST_CAMERA_PERMISSION);
        }

        cameraView = (SurfaceView) findViewById(R.id.camera_view);
        tvCodeInfo = (TextView) findViewById(R.id.QRCode_Info);

        barcodeDetector =
                new BarcodeDetector.Builder(this)
                        .setBarcodeFormats(Barcode.QR_CODE)
                        .build();
```

```
        cameraSource = new CameraSource
                .Builder(this, barcodeDetector)
                .setRequestedPreviewSize(640, 480)
                .build();

        cameraView.getHolder().addCallback(new
SurfaceHolder.Callback() {
            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                try {
                    cameraSource.start(cameraView.getHolder());
                } catch (IOException ie) {
                    Log.e("CAMERA SOURCE", ie.getMessage());
                }
            }
```

```
        @Override
        public void surfaceChanged(SurfaceHolder holder, int format,
int width, int height) {
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            cameraSource.stop();
        }
    });

    barcodeDetector.setProcessor(new
Detector.Processor<Barcode>() {
        @Override
        public void release() {
        }
```

```
@Override
        public void receiveDetections(Detector.Detections<Barcode>
detections) {
            final SparseArray<Barcode> barcodes =
detections.getDetectedItems();
            if (barcodes.size() != 0) {
                tvCodeInfo.post(new Runnable() {    // Use the post
method of the TextView
                    public void run() {
                        tvCodeInfo.setText(    // Update the TextView
                                barcodes.valueAt(0).displayValue
                        );
                    }
                });
            }
        }
    });
  }
```

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

        switch (requestCode) {
            case REQUEST_CAMERA_PERMISSION:
                if (grantResults.length > 1) {
                    permissionToUseCameraAccepted = grantResults[1] ==
PackageManager.PERMISSION_GRANTED;
                }
                break;
        }
        if (!permissionToUseCameraAccepted) finish();

    }
}
```

activity_activity_qrfrom_camera.xml

```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.example.ngtk.myqrcode.ActivityQRFromCamera">

    <SurfaceView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="8"
        android:id="@+id/camera_view"
        />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:id="@+id/QRCode_Info"
        android:textSize="20sp"
        android:text="@string/QRCode_info"
        />
</LinearLayout>
```