QR code recognition

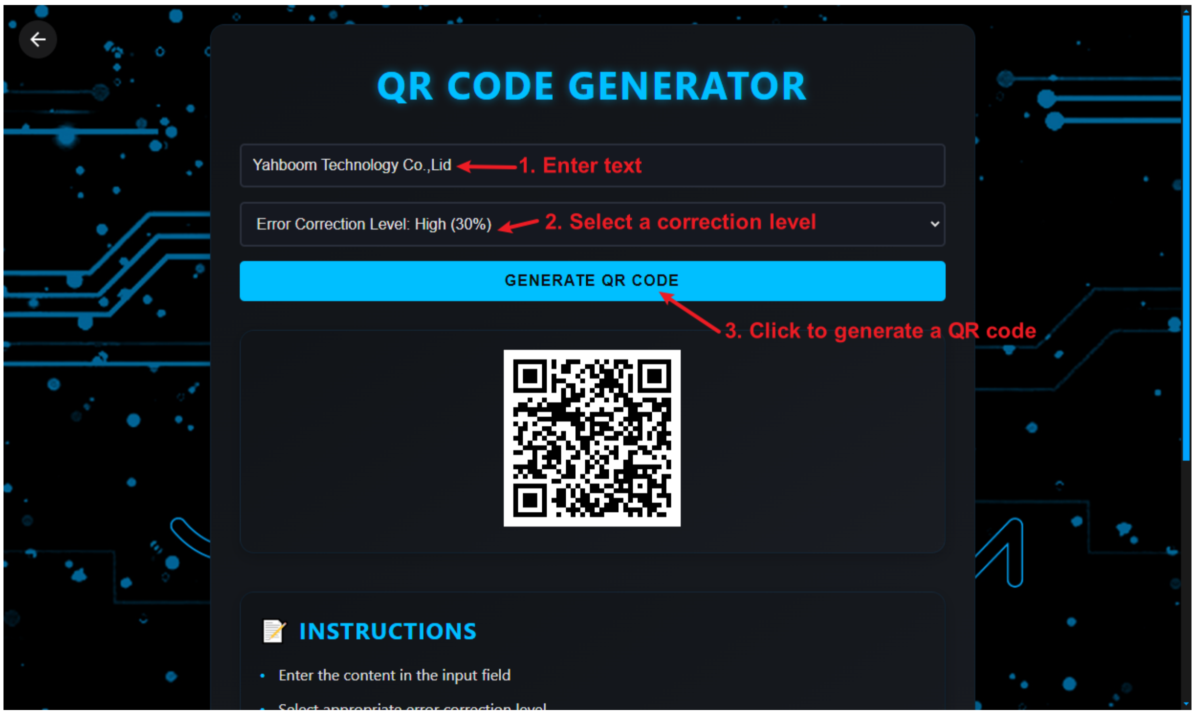
Introduction to the results of routine experiments

In this section, we will identify the QR code

> **What is the difference between a QR code and a barcode?**
>
> The main differences between QR codes and barcodes lie in the encoding method and functionality: barcodes are one-dimensional, have a small capacity, are mainly used for product identification, and require special equipment to scan; while QR codes are flat and two-dimensional, have a larger storage capacity, a wider range of applications, are fault-tolerant, and can be easily identified by smartphones, which makes QR codes more widely used in modern digital life.

First, we use the QR code generator to generate a QR code image (you can use the QR code generation tool provided in our attachment, or you can search online to generate a barcode)

Enter English or digital content in the input box and click the Generate QR Code button to generate it.


QR CODE GENERATOR
Yahboom Technology Co.,Lid
1. Enter text
Error Correction Level: High (30%)
2. Select a correction level
GENERATE QR CODE
3. Click to generate a QR code
INSTRUCTIONS
Enter the content in the input field
Select appropriate error correction level

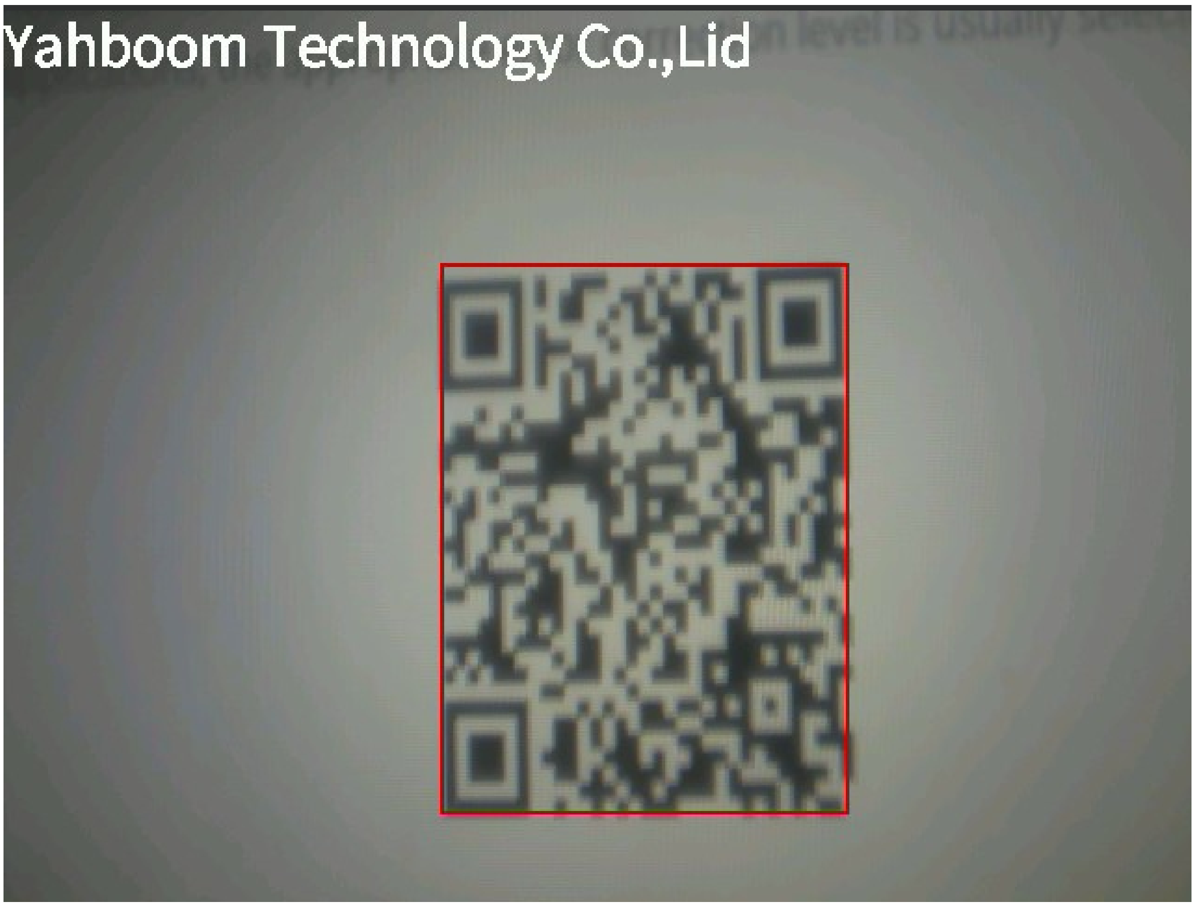
> **The error correction level of the QR code** is divided into four levels: L, M, Q, and H, with error correction capabilities of 7%, 15%, 25%, and 30% respectively. When the QR code is partially damaged or blocked, a higher error correction level can ensure that the QR code can still be correctly identified and decoded. However, it should be noted that a higher error correction level will increase the density and complexity of the QR code, making the pattern denser. In actual applications, the appropriate error correction level is usually selected according to the usage environment.

We enlarge the generated QR code

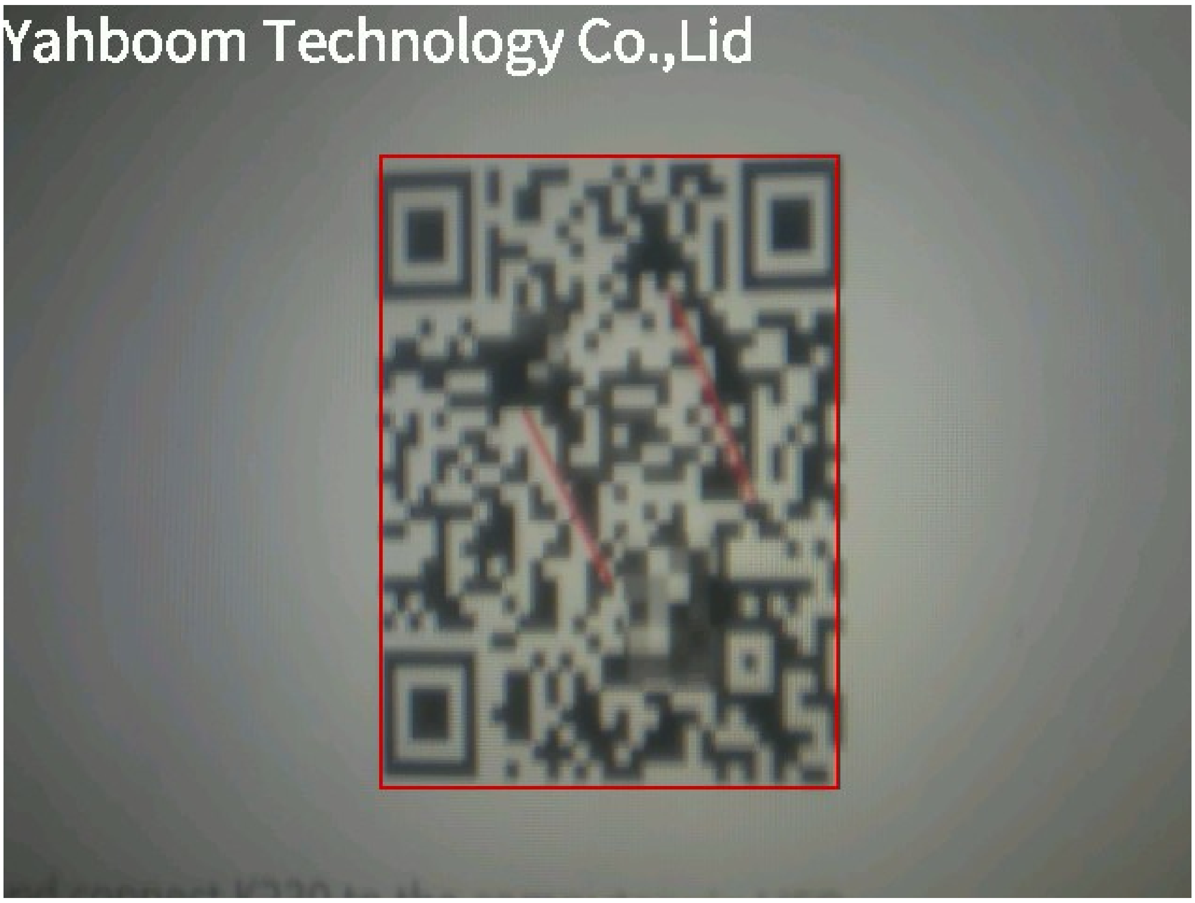
We use CanMV IDE to open the sample code and connect K230 to the computer via USB.

Click the Run button in the lower left corner of CanMV IDE to run the example

Point the camera at the QR code just generated, and you can see that the content of the barcode has been successfully recognized.



We selected a high error correction level (30%) when generating the QR code, so we covered the QR code with some noise to see if it can be recognized normally.



It can be seen that the QR code with a high error correction level (30%) has a relatively good anti-interference ability.

The source code of this section is located in [Source Code/06.Codes/02.find_qrcodes.py]

Code Explanation

The peripherals we will use in this section are mainly camera modules

The detection and recognition of QR codes is implemented by the find_qrcodes() method in K230, which belongs to the image module.

Complete code

```
import time
import math
import os
import gc
from media.sensor import Sensor
from media.display import Display
from media.media import MediaManager

def init_camera():
    """
    Initialize camera sensor with specified settings
    初始化摄像头传感器及其参数设置
    """
    sensor = Sensor()
    sensor.reset()  # Reset sensor to default settings | 重置传感器到默认设置
    sensor.set_framesize(width=640, height=480)  # Set resolution | 设置分辨率
    sensor.set_pixformat(Sensor.RGB565)  # Set pixel format | 设置像素格式
    return sensor

def init_display(sensor_obj):
    """
    Initialize display device
    初始化显示设备
    """
    Display.init(Display.ST7701, to_ide=True)  # Initialize ST7701 display | 初始
化ST7701显示器
    # Alternative virtual display initialization | 备用虚拟显示器初始化
    #Display.init(Display.VIRT, sensor_obj.width(), sensor_obj.height())

def process_qrcode(image, qr_result):
    """
    Process detected QR code and draw information on image
    处理检测到的二维码并在图像上绘制信息

    Args:
        image: Current frame image | 当前帧图像
        qr_result: QR code detection result | 二维码检测结果
    """
    if len(qr_result) > 0:
        # Draw rectangle around QR code | 在二维码周围画矩形
        image.draw_rectangle(qr_result[0].rect(), thickness=2)
        # Display QR code content | 显示二维码内容
        image.draw_string_advanced(0, 0, 30, qr_result[0].payload(),
                                   color=(255, 255, 255))
        print(qr_result[0].payload())
```

```python
def main():
    """
    Main function to run QR code detection loop
    运行二维码检测的主函数
    """
    # Initialize system components | 初始化系统组件
    sensor = init_camera()
    init_display(sensor)
    MediaManager.init()
    sensor.run()

    # Initialize FPS clock | 初始化FPS计时器
    clock = time.clock()

    try:
        while True:
            clock.tick()  # Update FPS clock | 更新FPS计时器

            # Capture frame | 捕获图像帧
            img = sensor.snapshot()

            # Detect QR codes | 检测二维码
            qr_codes = img.find_qrcodes()

            # Process detection results | 处理检测结果
            process_qrcode(img, qr_codes)

            # Display result | 显示结果
            Display.show_image(img)
            print(clock.fps())  # Print current FPS | 打印当前FPS

    except KeyboardInterrupt:
        print("Program terminated by user")
        # Clean up resources | 清理资源
        sensor.close()
        gc.collect()

if __name__ == "__main__":
    main()
```

Code structure

The main functions and structure of this section's routine are as follows:

1. Camera control: Initialize the camera, set the resolution to 640x480 and the pixel format to RGB565
2. Image Processing:
   - Continuously capture camera images
   - Find the QR code in each frame
   - When a QR code is detected, a rectangular frame is used to mark the QR code position on the screen.
   - Display the QR code content on the screen
3. Display output:
   - Real-time display of processed images
   - Print the decoded content of the QR code

    - Display the current frame rate (FPS)
4. Program Structure:
    - Adopt modular design to divide different functions into independent functions
    - Contains exception handling mechanism
    - Resources will be cleaned up at the end of the program

find_qrcodes()

```
image.find_qrcodes([roi])
```

This function finds all QR codes within the specified ROI and returns a `image.qrcode` list containing objects. For more information, see `image.qrcode` the documentation of the object.

In order for this method to work successfully, the QR code on the image needs to be as flat as possible. You can get a flat QR code that is not affected by lens distortion by using `sensor.set_windowing` the function to zoom in on the center of the lens, using `image.lens_corr` the function to remove barrel distortion from the lens, or by replacing the lens with a narrower field of view. Some machine vision lenses do not produce barrel distortion, but they cost more than the standard lenses provided by OpenMV, and these lenses are distortion-free.

- `roi` is a rectangle tuple specifying the region of interest `(x, y, w, h)`. If not specified, ROI defaults to the entire image. The operation is limited to pixels within this region.

Note: Compressed images and Bayer images are not supported.

kind `QRCode`

The QR code object is returned by `image.find_qrcodes` the function.

Constructor

```
class image.qrcode
```

Please use `image.find_qrcodes()` the function to create this object.

`corners`

```
qrcode.corners()
```

This method returns a list of tuples containing the four corners of the QR code, each tuple is in the format of (x, y). The order of the four corners is usually starting from the upper left corner and arranged in a clockwise direction.

`rect`

```
qrcode.rect()
```

This method returns a rectangle tuple (x, y, w, h) that can be used in other image processing methods, such as `image.draw_rectangle` the QR code bounding box in .

`x`

```
qrcode.x()
```

This method returns the x coordinate of the QR code's bounding box (int). You may also get this value using the [0] object.

`y`

```
qrcode.y()
```

This method returns the y coordinate of the QR code's bounding box (int). You may also get this value using [1] on the object.

`w`

```
qrcode.w()
```

This method returns the width of the QR code's bounding box (int). You may also get this value using [2] on the object.

`h`

```
qrcode.h()
```

This method returns the height of the QR code's bounding box (int). You may also get this value using [3] on the object.

`payload`

```
qrcode.payload()
```

This method returns the payload string of the QR code, such as a URL. You can also get the value by using the index [4].

`version`

```
qrcode.version()
```

This method returns the version number of the QR code (int). You can also get this value by using the [5] object.

`ecc_level`

```
qrcode.ecc_level()
```

This method returns the error correction level of the QR code (int). You may also get this value using [6] on the object.

`mask`

```
qrcode.mask()
```

This method returns the QR code mask (int). You may also get this value using [7] on the object.

`data_type`

```
qrcode.data_type()
```

This method returns the data type of the QR code. You can also get the value by using the [8] object.

`eci`

```
qrcode.eci()
```

This method returns the ECI (Encoding Indicator) of the QR code, which is used to store the encoding of the data bytes in the QR code. This value should be checked when processing QR codes that contain non-standard ASCII text. You can also get this value by using the index [9].

`is_numeric`

```
qrcode.is_numeric()
```

If the data type of the QR code is in digital format, it returns True.

`is_alphanumeric`

```
qrcode.is_alphanumeric()
```

Returns True if the data type of the QR code is alphanumeric.

`is_binary`

```
qrcode.is_binary()
```

Returns True if the data type of the QR code is in binary format. To correctly handle all types of text, check `eci` if is True to determine the text encoding of the data. Usually standard ASCII, but may be UTF-8 with two-byte characters.

`is_kanji`

```
qrcode.is_kanji()
```

If the data type of the QR code is in Japanese Kanji format, it returns True. If the return value is True, you need to decode the string yourself, because Japanese Kanji characters are 10 bits each, and MicroPython does not support parsing such text.