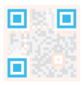
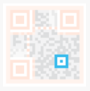
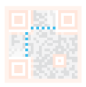
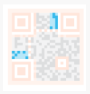
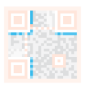
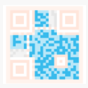
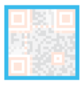
5. QR code recognition

5.1. Introduction to QR Code

A two-dimensional bar code is a black and white graphic that records data symbol information in a certain geometric shape distributed in a plane (two-dimensional direction) according to a certain pattern. In the coding, the concept of "0" and "1" bit streams that constitute the internal logic basis of the computer is cleverly used, and several geometric shapes corresponding to binary are used to represent text numerical information. It is automatically read by image input devices or photoelectric scanning devices to realize automatic information processing: it has some common features of bar code technology: each code system has its own specific character set; each character occupies a certain width; it has certain verification functions, etc. At the same time, it also has the function of automatically identifying information in different rows and processing graphic rotation change points. The two-dimensional code recognition technology of this project is mainly aimed at two-dimensional QR Code (Quick Response Code), which obtains and prints the information of QR code.

QR code is a type of two-dimensional bar code. QR comes from the abbreviation of "Quick Response" in English, which means quick response. It comes from the inventor's hope that QR code can allow its content to be decoded quickly. QR code not only has large information capacity, high reliability and low cost, but also can represent Chinese characters, images and other text information, has strong confidentiality and anti-counterfeiting properties and is very convenient to use. More importantly, the QR code technology is open source.

5.2. Structure of QR code

Image	Analysis
	**Positioning markings** Indicates the direction of the QR code.
	**Alignment markings** If the QR code is large, these additional elements help with positioning.
	**Timing pattern** Through these lines, the scanner can identify how large the matrix is.
	**Version information** This specifies the version number of the QR code being used. Currently, there are 40 different versions of QR codes. Versions used in the sales industry are usually numbered 1-7.
	**Format information** The format mode contains information about error tolerance and data mask modes and makes scanning the code easier.
	**Data and error correction keys** These modes store the actual data.
	**Quiet zone** This area is very important for the scanner to separate itself from the surroundings.

5.3. Features of QR Codes

The data values in QR codes contain repeated information (redundant values). Therefore, even if up to 30% of the QR code structure is destroyed, it does not affect the readability of the QR code. The storage space of QR codes is up to 7089 bits or 4296 characters, including punctuation and special characters, can be written into the QR code. In addition to numbers and characters, words and phrases (such as URLs) can also be encoded. As more data is added to the QR code, the code size increases and the code structure becomes more complex.

Installation (Note: Our factory program environment is already configured, no need to re-download and install)

First is pyzbar, download:

```
sudo apt-get install libzbar0
```

```
sudo pip install pyzbar
```

5.4. Code Implementation

After running the program, we can see the recognized content below the program, and the oled will also display the recognized content.

Source code path:

/home/pi/project_demo/07.AI_Visual_Recognition/05.QR_code_recognition//home/pi/project_demo/07.AI_Visual_Recognition/05.QR_code_recognition

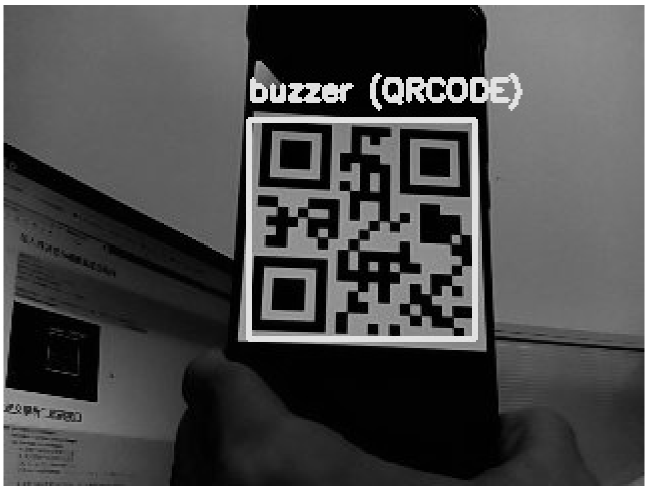
```
#导入oled屏幕库 Import oled screen library
import sys
sys.path.append('/home/pi/software/oled_yahboom/')
from yahboom_oled import *
# 创建oled对象 Create an oled object
oled = Yahboom_OLED(debug=False)
#bgr8转jpeg格式 bgr8 to jpeg format
import enum
import cv2

def bgr8_to_jpeg(value, quality=75):
    return bytes(cv2.imencode('.jpg', value)[1])
```

```
import cv2
import numpy as np
import pyzbar.pyzbar as pyzbar
from PIL import Image
import ipywidgets.widgets as widgets

# from picamera2 import Picamera2, Preview
# import libcamera

image_widget = widgets.Image(format='jpeg', width=640, height=480)
display(image_widget)                                      #显示摄像头组件 Display camera components
```

```
def decodeDisplay(image):
    barcodes = pyzbar.decode(image)
    for barcode in barcodes:
        # 提取二维码的边界框的位置 Extract the position of the bounding box of the QR code
        # 画出图像中条形码的边界框 Draw the bounding box of the barcode in the image
        (x, y, w, h) = barcode.rect
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 225, 225), 2)

        # 提取二维码数据为字节对象，所以如果我们想在输出图像上 Extract the QR code data as a bytes object, so if we want to output it on the image
        # 画出来，就需要先将它转换成字符串 To draw it, you need to convert it into a string first
        datatemp=0
        barcodeData = barcode.data.decode("utf-8")
        barcodeType = barcode.type

        # 绘出图像上条形码的数据和条形码类型 Draw the barcode data and barcode type on the image
        text = "{} ({})".format(barcodeData, barcodeType)
        cv2.putText(image, text, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX,0.5, (0, 225, 225), 2)

        # 向终端打印条形码数据和条形码类型 Print barcode data and barcode type to the terminal
        print("[INFO] Found {} barcode: {}".format(barcodeType, barcodeData))
        if(barcodeData!=datatemp):
            oled.clear()
            qrtype_str=f'[INFO] Found {barcodeType}'
            qrcode_str=f'barcode: {barcodeData}'
            oled.add_line(qrtype_str, 1)
            oled.add_line(qrcode_str, 3)
            oled.refresh()
            datatemp=barcodeData
    return image

def detect():
    oled.init_oled_process() #初始化oled进程 Initialize oled process
    camera = cv2.VideoCapture(0) #打开摄像头/dev/video0
    camera.set(3, 320)
    camera.set(4, 240)
    camera.set(5, 30)  #设置帧率 Setting the frame rate
    # fourcc = cv2.VideoWriter_fourcc(*"MPEG")
    # camera.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P',
'G'))
    # camera.set(cv2.CAP_PROP_BRIGHTNESS, 40) #设置亮度 -64 - 64  0.0 Set
Brightness -64 - 64 0.0
    # camera.set(cv2.CAP_PROP_CONTRAST, 50) #设置对比度 -64 - 64  2.0 Set Contrast
-64 - 64 2.0
    # camera.set(cv2.CAP_PROP_EXPOSURE, 156) #设置曝光值 1.0 - 5000  156.0 Set the
exposure value 1.0 - 5000 156.0
    ret, frame = camera.read()

    # picam2 = Picamera2()
    # camera_config = picam2.create_preview_configuration(main=
{"format":'RGB888',"size":(320,240)})
    # camera_config["transform"] = libcamera.Transform(hflip=1, vflip=1)
    # picam2.configure(camera_config)
    # picam2.start()
    # frame = picam2.capture_array()

    image_widget.value = bgr8_to_jpeg(frame)
    try:
        while True:
            # 读取当前帧 Read the current frame
            ret, frame = camera.read()
            #frame = picam2.capture_array()
            # 转为灰度图像 Convert to grayscale image
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            im = decodeDisplay(gray)

            cv2.waitKey(5)
            image_widget.value = bgr8_to_jpeg(im)
            # 如果按键q则跳出本次循环 If you press the q key, you will exit this
loop.
            if cv2.waitKey(10) & 0xFF == ord('q'):
                break
    except:
        #picam2.stop()
        camera.release()
        # 恢复屏幕基础数据显示 Restore basic data display on screen
        os.system("python3 /home/pi/software/oled_yahboom/yahboom_oled.py &")
```

```
detect()
```

```
#最后需要释放掉摄像头的占用 Finally, you need to release the camera's occupancy
camera.release()
oled.init_oled_process() #初始化oled进程 Initialize oled process
# 恢复屏幕基础数据显示 Restore basic data display on screen
os.system("python3 /home/pi/software/oled_yahboom/yahboom_oled.py &")
# picam2.stop()
# picam2.close()
#最后需要释放掉摄像头的占用 Finally, you need to release the camera's occupancy
```


buzzer (QRCODE)