# Barcode recognition

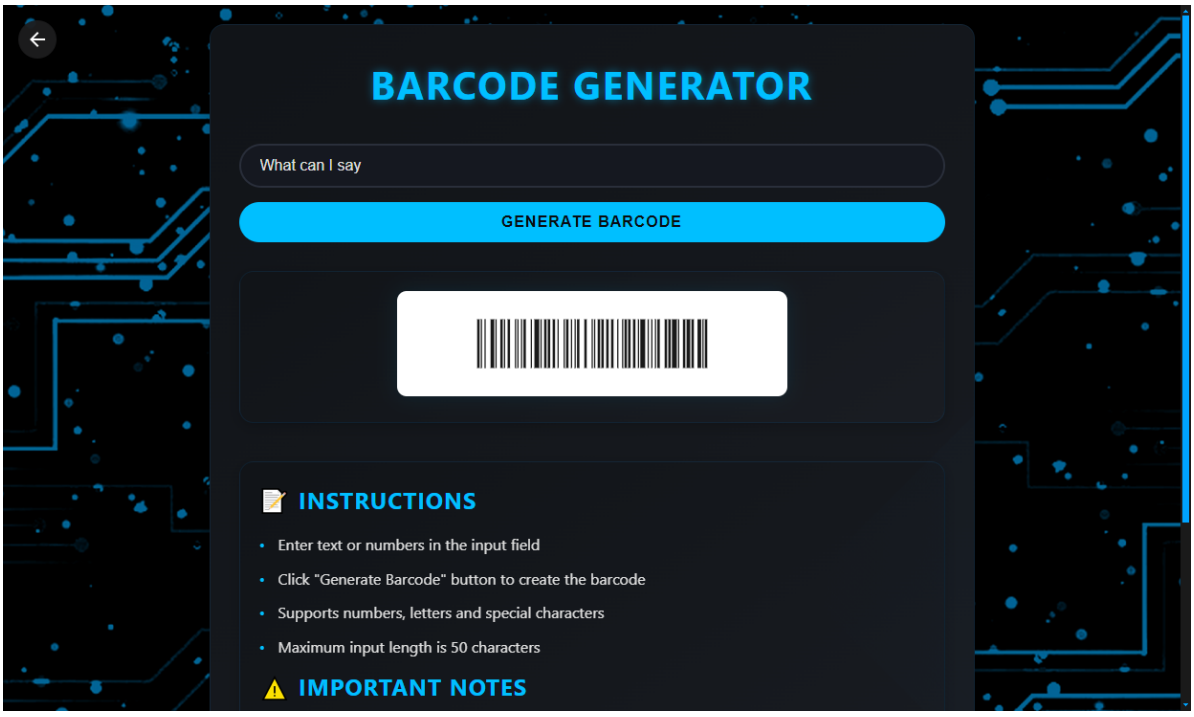


## Introduction to the results of routine experiments

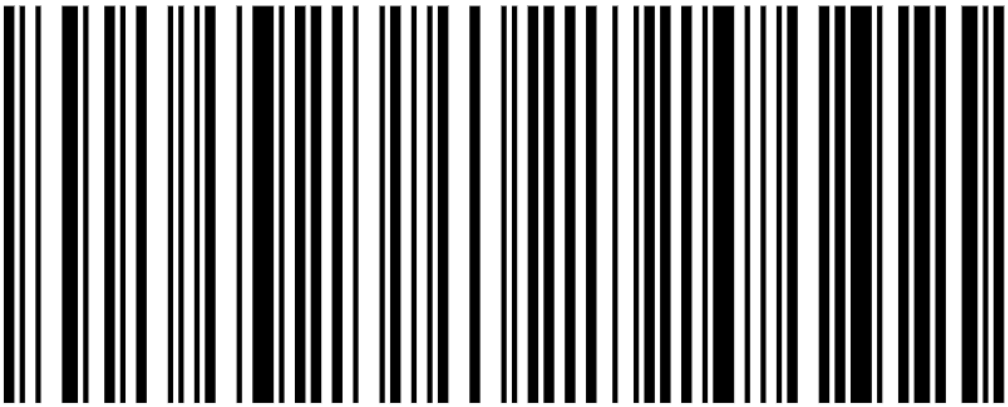
In this section, we will use K230 to recognize barcodes. First, we use the barcode generator to generate a barcode image.

(You can use the barcode 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 Barcode button to generate



We zoom in on the generated barcode

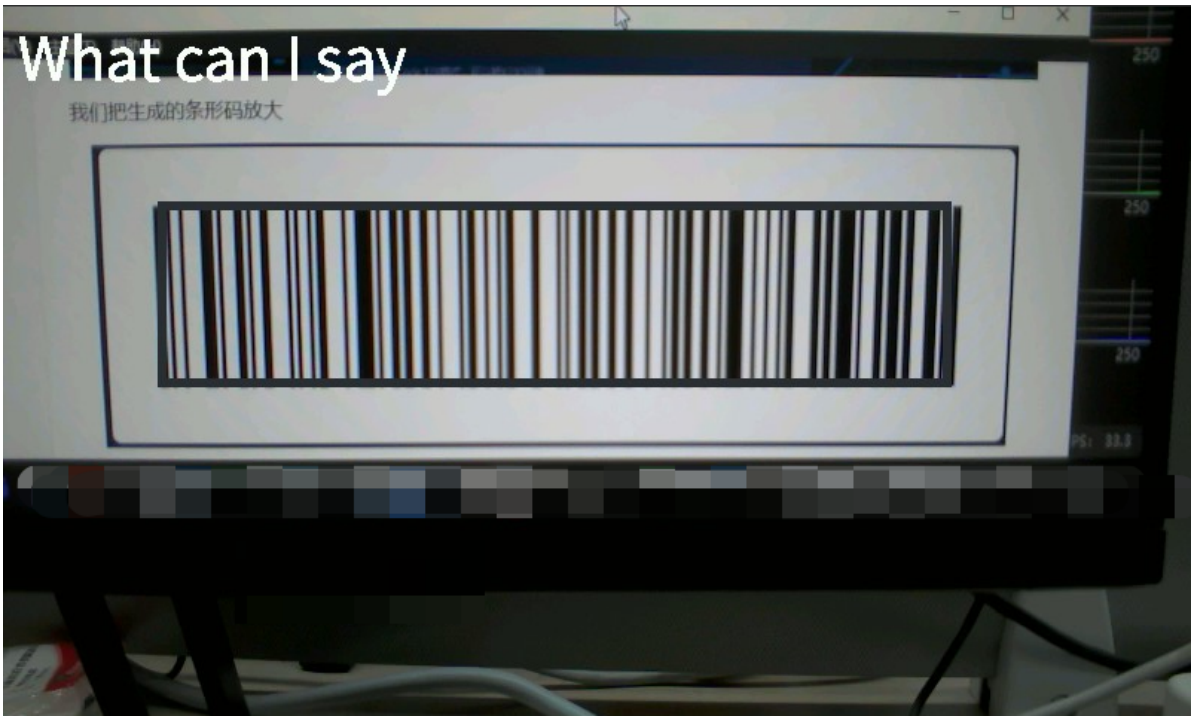


The example code for the K230 section of this section is located in: [Source Code/06.Codes/01.find_barcodes.py]

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.




At the same time, the lower left corner of CanMV IDE will also output our recognition results

```
Barcode CODE128, Payload "What can I say"
Barcode CODE128, Payload "What can I say"
Barcode CODE128, Payload "What can I say"
Barcode CODE128, Payload "What can I say"
Barcode CODE128, Payload "What can I say"
Barcode CODE128, Payload "What can I say"
Barcode CODE128, Payload "What can I say"
Barcode CODE128, Payload "What can I say"
Barcode CODE128, Payload "What can I say"
```

# Code Explanation

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

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

## Complete code

```
# 导入必要的模块 Import required modules
import time
import gc
from media.sensor import Sensor
from media.display import Display
from media.media import MediaManager
import image

# 定义条形码类型映射字典 Define barcode type mapping dictionary
BARCODE_TYPES = {
    image.EAN2: "EAN2",
    image.EAN5: "EAN5",
    image.EAN8: "EAN8",
    image.UPCE: "UPCE",
    image.ISBN10: "ISBN10",
    image.UPCA: "UPCA",
    image.EAN13: "EAN13",
    image.ISBN13: "ISBN13",
    image.I25: "I25",
    image.DATABAR: "DATABAR",
    image.DATABAR_EXP: "DATABAR_EXP",
    image.CODABAR: "CODABAR",
    image.CODE39: "CODE39",
    image.PDF417: "PDF417",
    image.CODE93: "CODE93",
    image.CODE128: "CODE128"
}

def barcode_name(code):
    """
    获取条形码类型名称
    Get barcode type name
    """
    return BARCODE_TYPES.get(code.type(), "UNKNOWN")

def init_camera():
    """
    初始化摄像头设置
    Initialize camera settings
    """
    sensor = Sensor()  # 构建摄像头对象 Create camera object
    sensor.reset()     # 复位和初始化摄像头 Reset and initialize camera
    # 设置帧大小为LCD分辨率(640x480) Set frame size to LCD resolution (640x480)
    sensor.set_framesize(width=640, height=480)
    # 设置输出图像格式 Set output image format
    sensor.set_pixformat(Sensor.RGB565)
```

```
    return sensor

def main():
    # 初始化摄像头 Initialize camera
    sensor = init_camera()

    # 初始化显示设置 Initialize display settings
    Display.init(Display.ST7701, to_ide=True)

    # 初始化media资源管理器 Initialize media resource manager
    MediaManager.init()

    # 启动sensor Start the sensor
    sensor.run()

    # 创建时钟对象用于FPS计算 Create clock object for FPS calculation
    clock = time.clock()

    while True:
        clock.tick()

        # 捕获图像 Capture image
        img = sensor.snapshot()

        # 查找图像中所有条形码 Find all barcodes in the image
        codes = img.find_barcodes()

        for code in codes:
            # 用矩形框标记条码位置 Mark barcode position with rectangle
            img.draw_rectangle(code.rect(), thickness=6, color=(46, 47, 48))

            # 获取条码类型和内容 Get barcode type and content
            code_type = barcode_name(code)
            payload = code.payload()

            # 打印条码信息 Print barcode information
            print(f"Barcode {code_type}, Payload \"{payload}\"")

            # 在图像中显示条码内容 Display barcode content in image
            img.draw_string_advanced(10, 10, 40, payload, color=(255, 255, 255))

        # 显示处理后的图像 Display processed image
        Display.show_image(img)

        # 执行垃圾回收 Perform garbage collection
        gc.collect()

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

## Code structure

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

1. Module import part:
    - Imported necessary modules such as time processing, garbage collection, sensors, displays, media management and image processing
2. Barcode type definition:
    - Created a dictionary `BARCODE_TYPES` mapping various barcode types (such as EAN13, ISBN13, CODE128, etc.)
3. Main functions:
    - `barcode_name()`: Get the name of the barcode type
    - `init_camera()`: Initialize the camera, set the resolution (800x480) and image format
    - `main()`: The main operating logic of the program
4. Main program flow:
    - Initialize the camera, display, and media manager
    - Enter an infinite loop and continue executing:
        - Capturing an Image
        - Find barcodes in images
        - For each barcode found:
            - Mark the location with a rectangular box
            - Get and display barcode type and content
        - Display the processed image
        - Perform garbage collection

# find_barcodes([roi])

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

This function finds all 1D barcodes within the specified ROI and returns a `image.barcode` list containing objects. For more information, see `image.barcode` the documentation of the object.

For best results, a window of 640 pixels long and 40/80/160 pixels wide is recommended. The lower the vertical window, the faster it will run. Since barcodes are linear, one-dimensional images, they need to have a higher resolution in one direction and a lower resolution in the other. Note that the function scans horizontally and vertically, so you can use a window of 40/80/160 pixels wide and 480 pixels long. Be sure to adjust the lens so that the barcode is in the area of sharpest focus. Blurred barcodes cannot be decoded.

This function supports all the following 1D barcodes:

- `image.EAN2`
- `image.EAN5`
- `image.EAN8`
- `image.UPCE`
- `image.ISBN10`
- `image.UPCA`
- `image.EAN13`

- `image.ISBN13`
- `image.I25`
- `image.DATABAR (RSS-14)`
- `image.DATABAR_EXP (RSS-Expanded)`
- `image.CODABAR`
- `image.CODE39`
- `image.PDF417`
- `image.CODE93`
- `image.CODE128`
- `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.

# `BarCode` kind

The barcode object is returned by `image.find_barcodes` the function.

## Constructor

```
class image.barcode
```

Please call `image.find_barcodes()` the function to create this object.

### `corners`

```
barcode.corners()
```

This method returns a list of tuples containing the four corners of the barcode, each tuple is in the format of (x, y). The four corners are usually arranged in a clockwise direction starting from the top left corner.

### `rect`

```
barcode.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 barcode bounding box in .

### `x`

```
barcode.x()
```

This method returns the x-coordinate of the barcode's bounding box as an integer. You can also get this value doing [0] on the object.

```
barcode.y()
```

This method returns the y-coordinate of the barcode's bounding box (an integer). You may also get this value doing [1] on the object.

## `w`

```
barcode.w()
```

This method returns the width of the barcode's bounding box as an integer. You may also get this value doing [2] on the object.

## `h`

```
barcode.h()
```

This method returns the height of the barcode's bounding box as an integer. You may also get this value using [3] on the object.

## `payload`

```
barcode.payload()
```

This method returns the barcode's payload string, for example: "Quantity".

You can also get this value doing [4] on the object.

## `type`

```
barcode.type()
```

This method returns the type of the barcode (an integer). Possible types include:

- image.EAN2
- image

.EAN5

- image.EAN8
- image.UPCE
- image.ISBN10
- image.UPCA
- image.EAN13
- image.ISBN13
- image.I25
- image.DATABAR
- image.DATABAR_EXP
- image.CODABAR
- image.CODE39
- image.PDF417 (to be enabled in the future, not currently available)
- image.CODE93
- image.CODE128

You can also get this value doing [5] on the object.

## `rotation`

```
barcode.rotation()
```

This method returns the barcode's rotation angle (measured in radians, as a floating point number).

You may also get this value doing [6] on the object.

## `quality`

```
barcode.quality()
```

This method returns an integer representing the number of times the barcode was detected in the image.

As you scan a barcode, each new scan line decodes the same barcode. Each time this process occurs, the barcode value increases.

You may also get this value doing [7] on the object.