# MiraCode Experiment

MiraCode is a multi-colored barcode which is used for data transmission. Compare with existing barcodes, MiraCode uses 16 colors which makes its unit block capacity four times as the wide used QR code.

## Architecture

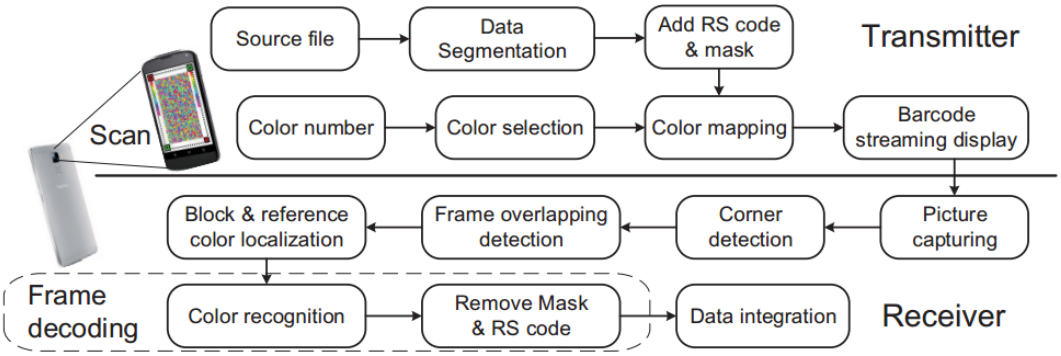


Figure 1 MiraCode system architecture

As shown is Fig. 1, MiraCode encodes source data into color barcode frames, and the receiver decodes these data by analyzing captured images. Different from existing QR code, which usually contains an URL link, MiraCode is able to transmitter data without file size limitation. As a result, MiraCode is more likely to be seen as a new communication technology . The system will segment the source file into different color barcode frames, and the combine all these decoded data segments into one file at the receiver side.

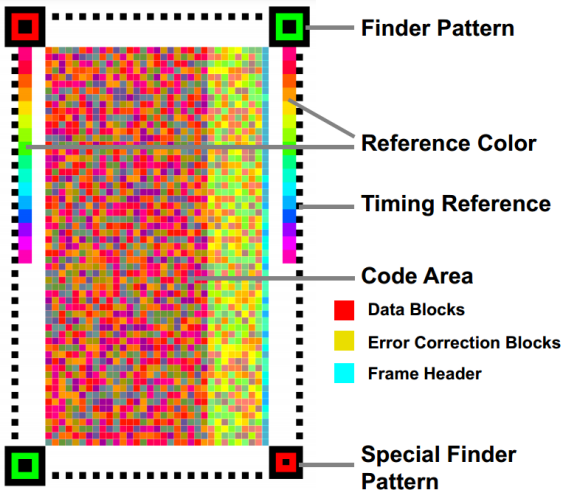


Figure 2 Frame Structure

Fig. 2 shows the frame structure of MiraCode. To make the code area easier to learn, we add some colors to separate different areas which have different functions.

## Working process

The working process of MiraCode is separated into two parts, the transmitter and the receiver.

The transmitter is to encode the cource file into a colored barcode streaming, which means different barcode frames containing data segments will be displayed on the transmitter screen. Considering that each frame of color barcode has a fixed capacity, each source file that need to be transmitted must be segmented to fit the frame standard. Then RS code will be added into these data segments in case of correcting errors that may occur during the image analyzing process in the receiver side. Then these data will be mapped into colored blocks in each barcode frame. In each frame there is a frame header which contains some frame information, for example, frame number, total frame number, data size, etc.

The receiver is to decode these barcode frames and combine them into one file. To remap these colors into data segments, we need to locate the barcode from the captured images first. The finder pattern is used to locate the four corners of the barcode. Then by using the timing reference, we can locate each little block in the code area. The reference colors at the two sides of the barcode is to help remap the colors into data. Then these remapped data will be corrected by using the RS code. At last all these data segments will be combined into one file, which is the received source file.

## Problems

We have implemented MiraCode on Android phones. After understanding the working process of MiraCode, we have left some functions for you to solve, I think it is a good opportunity to understand this project and get familiar with Android.

1. In the barcode generator, the function we need to complete is in the file whose path is: src/edu/SJTU/zhusy54/colorTrans/Buff2color.java. According to the source data, find the color number for each block in one barcode frame. In miracode, the number of all the blocks are 59*33. In each row, the first 24 blocks are data block, the later 8 blocks are used for error correction, and the last block is used to store frame information. There is an int array, withReedSolomon, stores each byte of this data segment, [0,255]. Note that error correction code has not been added into this data segment. Add the final color result into colorArr, which is also an int array, [0,15].
2. This problem aims at Android development. Please complete the layout file which is the layout of the first activity, the file path is res/layout/activity_main.xml. The final activity user interface should be like the image shown below. The id of these components should be the same as the list below:

```
EditText: "@+id/numText"
Button: "@+id/btnConfirm"
Button: "@+id/btnDefault"
Button: "@+id/btnTest"
```


3. In the barcode decoder, the function we need to complete is in the file whose path is: src/edu/SJTU/zhusy54/colorbarcodedecoder/DecoderThread.java. The function is:

```
private boolean calRefBloc(Point[] refVert, boolean isXaxis,
int stepUnit, boolean isSpecial,
      Point[] timingBlock)
```

This function calculate the location of each block in the timing reference, and store them in the array called timingBlock. In this function, refVert stores the start and end point of this timing refrence. isXaxis indicates whether the difference in the X axis is larger than that in the Y axis. stepUnit is 1 or -1, and isSpecial indicate whether this timing reference is affected by the special finder patter.

```
本函数中，refVert 存储同侧的两个 finder pattern 的位置，isXaxis 表明了该条 timingreference 中两点是否横坐标差距比纵坐标更大
* stepUnit 为正负 1，根据 timingreference 方向进行判断；isSpecial 表明是否受到 special finder patter 的影响
```