



THỊ GIÁC MÁY TÍNH

# NHẬN DIỆN BARCODE - QR CODE

BÁO CÁO BÀI TẬP LỚN

## Nhóm 18

1. Lương Hữu Phú Lộc – 1511844
2. Phạm Ngọc Khôi Nguyên – 1512221
3. Nguyễn Trọng Phúc – 1512534
4. Mai Thiện Quang – 1512640

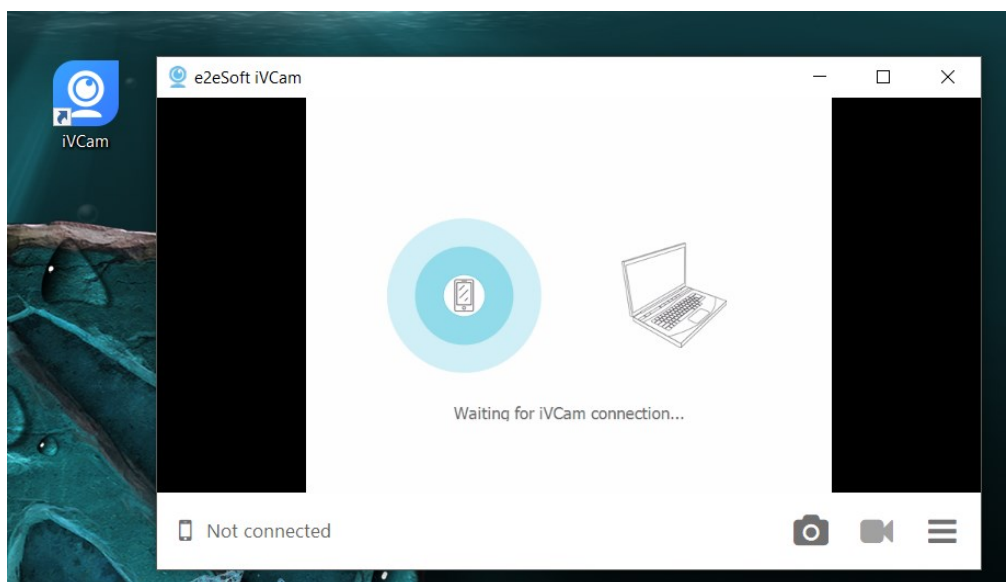
GVHD: TS. NGUYỄN ĐỨC THÀNH

## Yêu cầu

Đọc mã vạch bar code, QR code



\*Do webcam của laptop có chất lượng hình ảnh không được tốt, và cũng dễ tăng độ chính xác khi nhận diện, nhóm sử dụng phần mềm **iVCam** để kết nối camera của điện thoại iOS với laptop. Phần mềm này chỉ có trên hệ điều hành iOS, để kết nối thì cả máy tính và điện thoại phải cài đặt phần mềm đồng thời access chung một mạng wifi. **(Đối với điện thoại android vẫn có những phần mềm khác để kết nối điện thoại với laptop)**



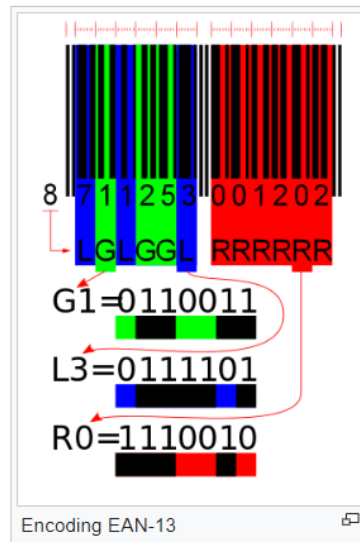
*Giao diện phần mềm trên máy tính*

## PHẦN 1: BARCODE

### I. Nhận diện Barcode

#### 1. Đặc điểm của barcode

- Loại barcode nhóm nhận diện là loại EAN-13, là loại barcode chủ yếu dùng ở châu Âu. Loại barcode này có các đặc điểm như sau:

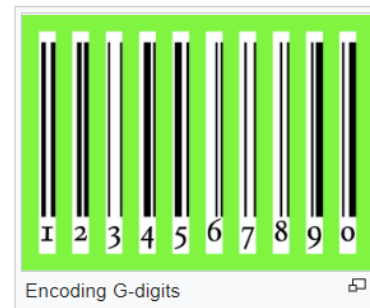


- Trong barcode có các vạch dài hơn nằm ở 2 đầu và ở giữa, gọi là các vạch bảo vệ
- Có tổng cộng 13 số, mỗi số được biểu thị bằng 8 vạch đen trắng với bề dày của 1 vạch bằng bề dày của vạch đen ngoài cùng bên trái, ngoại trừ số ngoài cùng bên trái. Con số này quy định cách mã hoá cho các số từ vị trí 2 → 6 như sau:

Số đầu tiên	2	3	4	5	6	7
0	Lẻ	Lẻ	Lẻ	Lẻ	Lẻ	Lẻ
1	Lẻ	Lẻ	Chẵn	Lẻ	Chẵn	Chẵn
2	Lẻ	Lẻ	Chẵn	Chẵn	Lẻ	Chẵn
3	Lẻ	Lẻ	Chẵn	Chẵn	Chẵn	Lẻ
4	Lẻ	Chẵn	Lẻ	Lẻ	Chẵn	Chẵn
5	Lẻ	Chẵn	Chẵn	Lẻ	Lẻ	Chẵn
6	Lẻ	Chẵn	Chẵn	Chẵn	Lẻ	Lẻ
7	Lẻ	Chẵn	Lẻ	Chẵn	Lẻ	Chẵn
8	Lẻ	Chẵn	Lẻ	Chẵn	Chẵn	Lẻ
9	Lẻ	Chẵn	Chẵn	Lẻ	Chẵn	Lẻ

- Bảng mã hoá chẵn lẻ tương ứng với các số như sau:

Giá trị số	Lẻ	Chẵn
0	0001101	0100111
1	0011001	0110011
2	0010011	0011011
3	0111101	0100001
4	0100011	0011101
5	0110001	0111001
6	0101111	0000101
7	0111011	0010001
8	0110111	0001001
9	0001011	0010111



Với 0 là tương ứng với vạch trắng, 1 tương ứng với vạch đen, ta có thể để ý 1 số sẽ gồm 2 vạch đen và 2 vạch trắng xen kẽ nhau (tuy nhiên bề dày của các vạch thì không giống nhau)

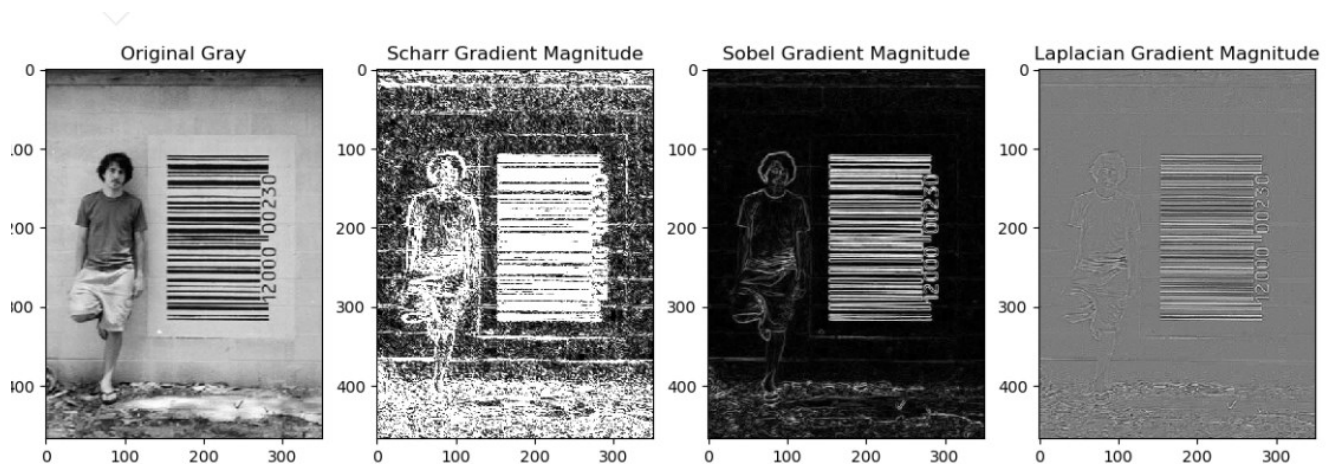
Quy ước bề dày của vạch nhỏ nhất là 1, ta sẽ tính toán bề dày các vạch khác để rút gọn bảng trên, ví dụ:

000 → sẽ tương ứng với độ dày = 3

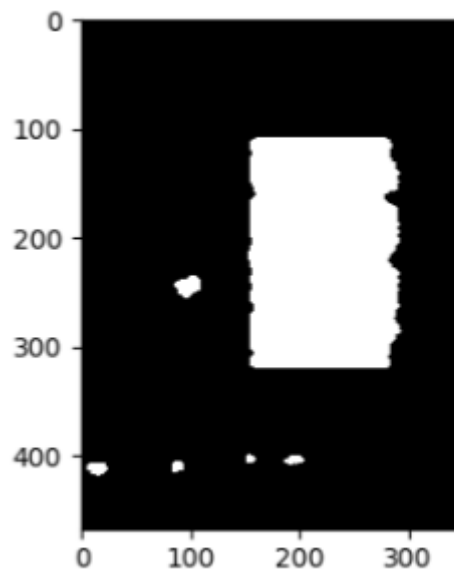
Như vậy ta có thể rút gọn 0001101 → 3211

## 2. Giải thuật nhận diện và giải mã barcode

- Sử dụng Sobel Gradient do barcode gồm các vạch đen trắng xen kẽ nhau tạo nên các cạnh liên tiếp nhau như ví dụ dưới đây:



- Sau khi có được kết quả từ Sobel Gradient, sử dụng threshold và một số phép biến đổi hình thái thì vùng barcode sẽ trở thành một vùng trắng có kích thước lớn so với các vùng khác trong hình (như hình dưới). Sử dụng phương pháp này ta có thể xác định được vị trí barcode một cách dễ dàng.



- Sau khi tìm được vị trí barcode, crop vùng barcode và sử dụng affine để đưa barcode về nằm ngang để thuận lợi cho việc giải mã

## Giải mã barcode

1. Với barcode đã affine về nằm ngang, lấy đường chính giữa của barcode để giải mã (có thể lấy thêm 1 vài đường khác ở những vị trí khác tuy nhiên với 1 đường chính giữa kết quả của nhóm cũng đã tương đối tốt)
2. Sau khi lấy được đường chính giữa, xác định vị trí 2 vạch đen ngoài cùng (2 biên) của barcode bằng cách đếm từ trái và từ phải sang cho đến khi gặp pixel đen đầu tiên, 2 vị trí này đặt là **left** và **right**



3. Sau đó, thực hiện vòng lặp đếm từ vị trí **left** đến **right** để xác định bề dày của từng vạch đen – trắng trong barcode bằng cách đếm số lượng pixel giống nhau
4. Ta thấy rằng các vạch đen trắng dài hơn các vạch còn lại có bề dày nhỏ nhất (các vạch bảo vệ) nên chuẩn hoá bề dày này là 1 và tính toán bề dày các vạch còn lại trong barcode theo chuẩn này
5. Khi có được bề dày các vạch, ghép độ dài 4 vạch liên tiếp nhau (bỏ qua các vạch bảo vệ) để nhận diện chữ số tương ứng với vạch. Ví dụ số 9 trên hình có bề dày là: **3112**, và đồng thời xác định kiểu mã hoá cho từng số, ví dụ 3112 so trong bảng chuẩn sẽ tương ứng với chuỗi bit là 0001011, như vậy số 9 này là kiểu mã hoá lẻ (L).
6. Sau khi tìm được tất cả các số tương ứng với các vạch, do số ngoài cùng (như trên hình là số 5) không được biểu diễn bằng vạch nào cả, nhưng số này quy định cách thức mã hoá vạch → ta có thể dựa vào cách thức mã hoá để suy ra số này, ví dụ số 5 có kiểu mã hoá là LGGLLG tương ứng với các số từ vị trí 2 đến 7 trong barcode

## II. Nhận xét kết quả

- Barcode nhận diện tương đối chính xác, tốc độ nhanh, tuy nhiên kết quả còn phụ thuộc vào độ sáng của môi trường, cũng như chất lượng webcam.

## PHẦN 2: QR CODE

### I. Thành phần QR code

#### 1. Định nghĩa

- QR Code (mã QR) được tạo ra bởi Denso Wave (công ty con của Toyota) vào năm 1994, có hình dạng bao gồm các điểm đen và ô vuông nằm trong ô vuông mẫu trên nền trắng. QR Code có thể được đọc nhanh hơn, tiết kiệm thời gian và không gian so với các loại mã vạch truyền thống.

- Mã QR có kích thước thay đổi từ 21x21 pixels đến 177x177 pixels. Kích thước này gọi là các versions. Trong giới hạn môn học và thời gian cho phép, nhóm chỉ thực hiện giải mã version 1 (21x21) để làm mẫu.

#### 2. Chế độ mã hóa dữ liệu

- Có 4 chế độ (mode) để mã hóa dữ liệu tùy thuộc vào tính chất dữ liệu. Mỗi mode sử dụng 1 phương pháp riêng để chuyển dạng chữ sang bit nhị phân.

Mode Name	Mode Indicator
Numeric Mode	0001
Alphanumeric Mode	0010
Byte Mode	0100
Kanji Mode	1000
ECI Mode	0111

- Trong báo cáo này, chọn mode Byte để giải mã và dữ liệu chủ yếu là dạng kí tự chữ. Mã chỉ thị tương ứng là 0100. Số lượng bit tương ứng với mỗi mode:

- Numeric mode: 10 bits
- Alphanumeric mode: 9 bits
- Byte mode: 8 bits
- Japanese mode: 8 bits

- Mỗi kí tự dạng Byte sẽ được mã hóa 8bit (đối với version 1) theo chuẩn UTF-8

### 3. Mã sửa lỗi

- Từ dữ liệu có được sau khi mã hóa, 1 tập hợp các mã sửa lỗi được tạo ra từ chuỗi data này theo thuật toán Reed-Solomon. Việc so sánh dữ liệu và vùng sửa lỗi giúp máy quét chuẩn thực dữ liệu có chính xác hay không.

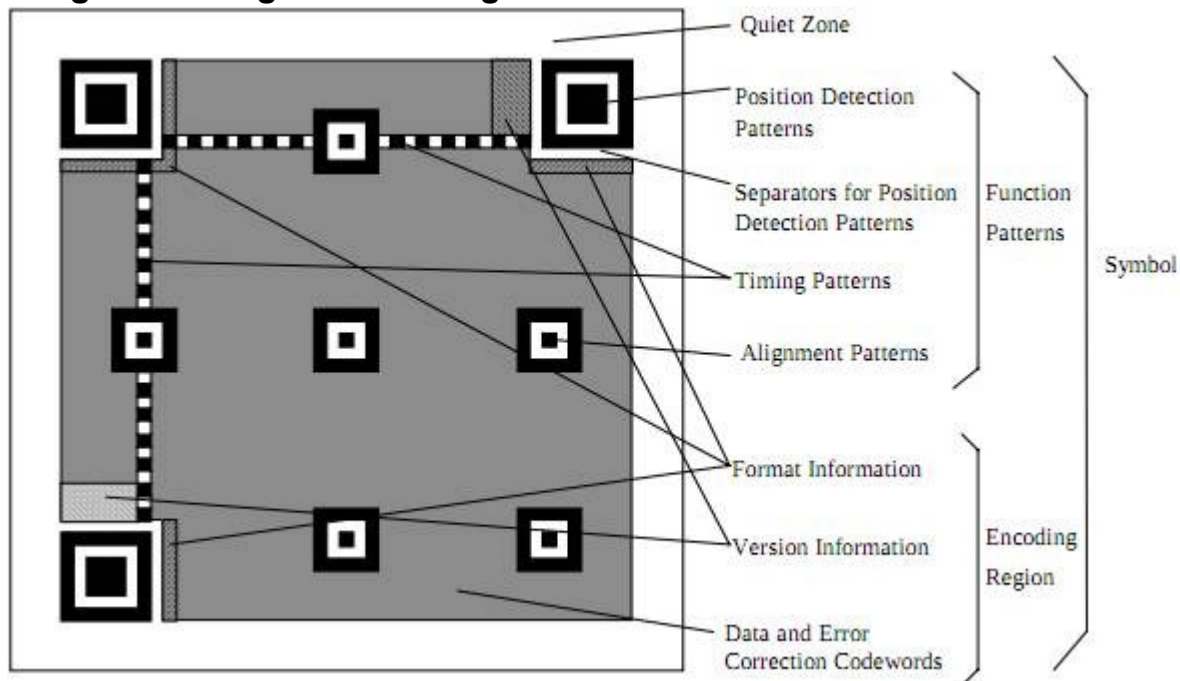
- Vì điều kiện thời gian không cho phép, thuật toán sửa lỗi dữ liệu sẽ không được áp dụng trong báo cáo này. Do đó sẽ có những sai sót trong quá trình đọc dữ liệu từ QR code.

### 4. Masking data

- Để đọc dữ liệu chính xác hơn, QR code sẽ được che phủ bởi 1 mask. Cụ thể hơn chỉ có phần data và sửa lỗi mới được masking. Do đó để chuyển đổi data về nguyên gốc cần phải xác định mask nào đang được sử dụng.

- Thông tin về mask được mã hóa thành 3 bit trong chuỗi format thể hiện cho các pattern tương ứng từ 0 đến 7

### 5. Vùng chức năng cơ bản trong QR code



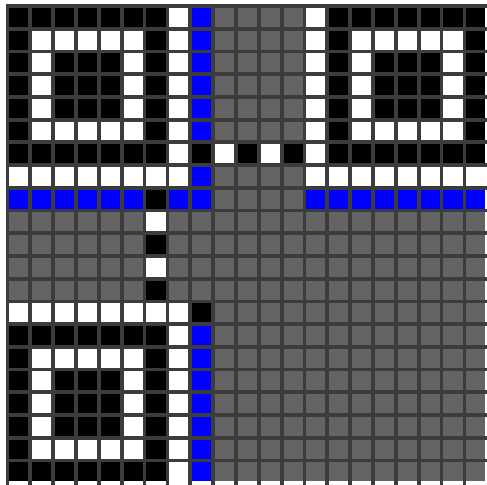
- Finder pattern: Dùng để xác định hiện diện của mã QR và chiều quay tương ứng

- Separator: vùng chia tách với data

- Timing pattern: vùng kết nối giữa các finder pattern

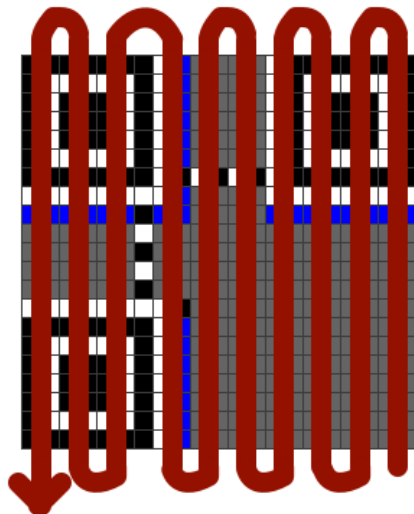


- Alignment pattern: Vùng căn chỉnh, chức năng tương tự như finder pattern nhưng phần lớn dùng để tránh lỗi khi xoay các QR code
- Dark module: 1 module đen luôn có tại vị trí này
- Format area: Là vùng dành ra để lưu trữ thông tin version, mask, error correction level (cấp sửa lỗi)

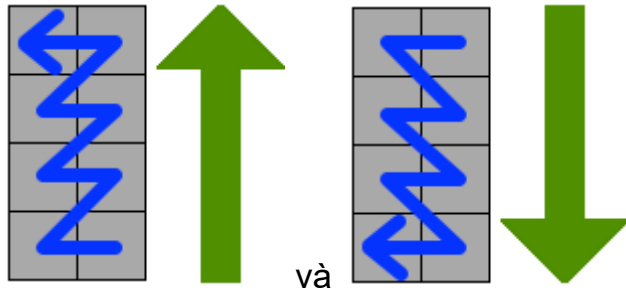


## 6. Sắp xếp bit data

- Bắt đầu từ góc dưới bên phải:



- Tuy nhiên, các bit data phải được sắp xếp tránh các vùng chức năng.
- Vì mỗi 8 bit là 1 kí tự nên vùng data được chia làm các khối với thứ tự giải mã như sau:



## II. Quy trình giải mã QR Code

- Xác định QR code

Dựa vào sự xuất hiện của finder pattern để xác định vị trí của QR code, trên ma trận ảnh. Đồng thời tọa độ các đỉnh và kích thước một module (ô vuông). Từ đó chuyển toàn bộ QR thành ma trận nhị phân (trắng – 1, đen – 0)

- Tìm vùng format

Vùng format cho phép trích xuất thông tin version, masking và error correction level. Từ đó lựa chọn được giải thuật phù hợp để lật ngược mask.

- Unmasking

Từ mã chỉ thị masking trong vùng format, lựa chọn thuật toán unmask phù hợp để đưa data về nguyên gốc

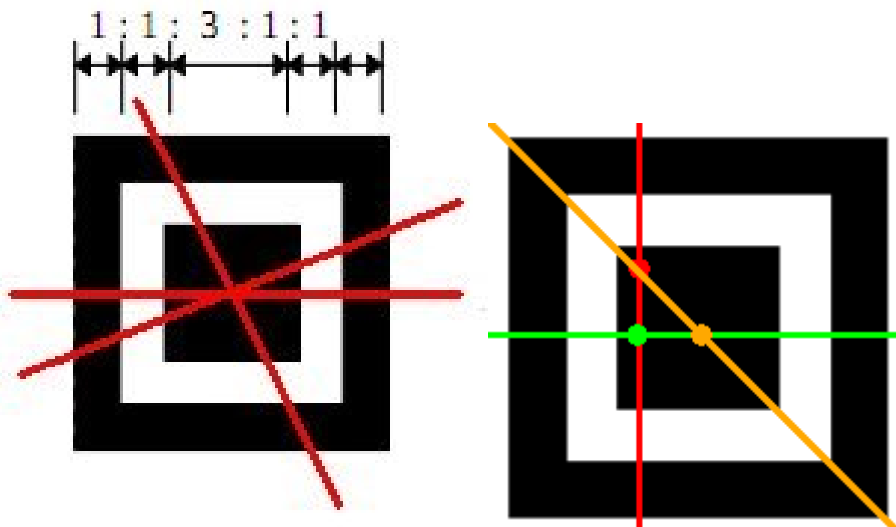
- Tách dữ liệu

Loại bỏ các vùng chức năng, đọc các bit data theo thứ tự zig zag cho mỗi khối (block) 8 bit. Bắt đầu từ góc dưới phải. Số lượng block phải ứng với data length.

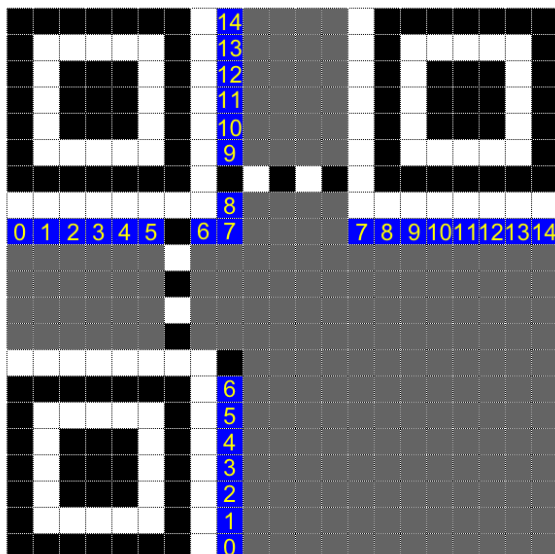
- Chuyển dạng kí tự

## III. Giải thuật

- Lọc và đưa ảnh về dạng ảnh xám có phân ngưỡng để chỉ còn lại 2 màu đen và trắng
- Tìm kiếm các finder pattern bằng cách xét tỉ lệ 1:1:3:1:1 của các pixel trắng và đen. Do pattern có tính đối xứng nên có thể kiểm tra theo 3 hướng dọc, ngang và chéo để chắc chắn. Từ đó tìm được tâm của pattern và kích thước của mỗi module (1 điểm đen – trắng). Nội suy theo kích thước module để tìm tọa độ các góc.



- Trích xuất dạng nhị phân của các module trên toàn bộ QR code.
- Các vùng chức năng có sẵn tọa độ nên có thể bám theo đó để tìm kiếm thông tin format và xác định mask của QR. Chuỗi format nên được xác định 2 lần để tránh sai sót. Giải ngược lại các bit tương ứng theo mask đã xác định.



- Đọc các bit data zig zag theo block và chuyển về dạng kí tự. So sánh số lượng kí tự với data length để kết thúc quá trình đọc.







#### IV. Nhận xét

- QR code có rất nhiều version và độ phức tạp cũng như cấu trúc thay đổi trên mỗi version. Việc giải mã khá là phức tạp, do đó nhóm chỉ chọn version 1 để thực hiện. Vì tính đơn giản nên sức chứa tối đa của version này là 17 kí tự

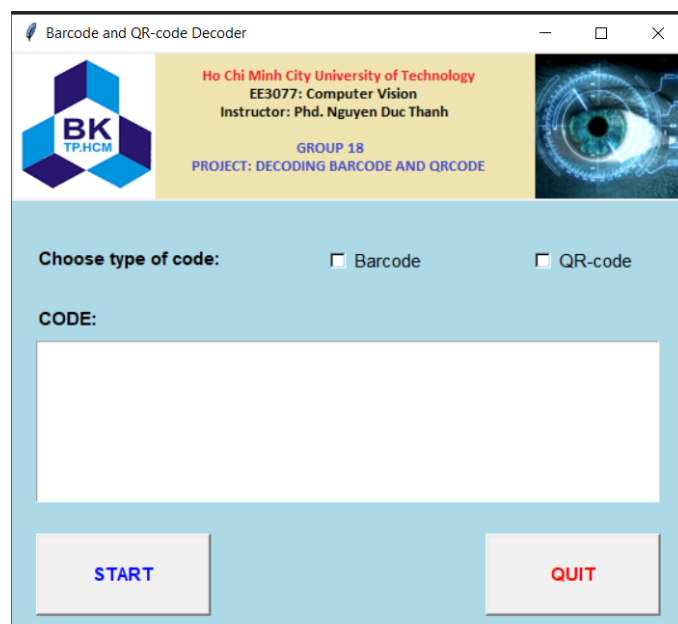
- Giải thuật sửa lỗi (error correction) chưa được áp dụng trong sản phẩm nên tỉ lệ nhận diện sai không nhỏ, chỉ cần lỗi 1 bit data là dữ liệu đã sai. Tuy nhiên, hiện tại tỉ lệ chính xác của phần mềm là 90%.

### PHẦN 3: CHƯƠNG TRÌNH

Folder Code trong báo cáo của nhóm như sau:

 __pycache__	11/18/2018 4:42 A...	File folder	
 Barcode.py	11/18/2018 12:30 ...	PY File	9 KB
 DHBK_logo.ppm	11/18/2018 1:51 A...	PPM File	54 KB
 main.py	11/18/2018 4:17 A...	PY File	8 KB
 QRCode.py	11/18/2018 3:52 A...	PY File	17 KB
 setting.py	11/17/2018 11:56 ...	PY File	1 KB

- Trong đó file main là chương trình chính, 2 file Barcode.py và QRCode.py là 2 thư viện để nhận diện barcode và QRCode do nhóm tự viết, file setting.py là những thông số điều chỉnh cho file QRCode
- Khi chạy file main, giao diện như sau:



- Để nhận diện ta chọn loại code, sau đó bấm start để mở camera.