**Abstract:**

Today, we live in a world of fast paced. People do not have time to spare. Thus, we yearn for comfort. Most of us are too busy, and we do not have enough time even to go to the grocery store. Recently, new type of virtual stores is created to make shopping easier and faster. For this project, we created a Smart Virtual Store, this includes mobile application for users and C# application on server part of this store. By this application user can scan a QR code Located on a virtual product on this store and make a list of items to buy, then send his order to the server of the store to be delivered later to specified address .In this way of shopping, customers can exploit their smart phones to buy their needs without the trouble of carrying physical products.

Table of Contents

Introduction 3

Chapter 1: QRcode Generator6

1.1 What is QRcode 6

1.2 How to Encode 8

1.3 C# Application 18

Chapter 2: QRcode Reader 20

2.1 Components in QR code reader20

2.2 QR code properties 21

2.3 Operations in detail 22

2.4 C# Application 23

Chapter 3: Mobile Application24

3.1 Mobile application steps 24

3.2 Payment 31

3.3 Some of difficulties in this part31

Chapter 4: Communication Between Customer and Server.32

4.1 Server mobile 32

4.2 AT Commands 32

4.3 Some of the difficulties in this part 33

Chapter 5: server part of Smart Virtual Store ..34

5.1 Server Application 34

Chapter 6: Smart Virtual Store Website. .36

**Conclusion** .37

**Appendices** .38

Appendix A 38

Appendix B 46

**Introduction:**

At the beginning the idea of our project was object recognition with mobile phone. Mobile phone take an image of any object then recognize it. It seems that this is possible and applicable, the images taken with the mobile phone camera are sent to a server which does the expensive object recognition algorithms and the matching with a database. As communication method the Multimedia Message System (MMS) is used from the mobile phone to the server and the Short Message System (SMS) to send the answer back to the mobile phone.

We faced difficulties and discouragement in implementing this idea, especially because it needs image processing operations and we know nothing about it. The success rate of such project is not promising as most of our instructors said. As a result, we change in the idea so we can deal with it easier. We introduce the barcode and try to scan it to recognize the product.

So, we settled on using QRcode as key of each product on the virtual store (no physical products on shelves), and a quick scan for this code by the application on customer’s phone can deliver the order to customer later.

We implement the idea by build our own C# QRcode generator that encode data about each product, then build the C# reader for QRcode, after that we include the reader in a j2me mobile application which continue forming a list of products and request it to be delivered to him later, based on their choices , also we build a store’s server part C#, that receive the coming requests form customers mobiles and serves it .

By now we are still working in other 2 parts, First, is using ” paypal” accounts so the customer can pay money when he want to purchase items requested , money is cashed into server paypal account . Second is a website for the Smart Virtual Store, customers can create accounts on it and leave their own basic information so, they can be able to skip some of the data entry steps while using the application .

**Why QRcode ?**

QRcode is two dimensional barecode which is categorized in matrix code.

  
figure1:QR code sample

**Features of QRcode :**

* High-speed reading (QR is derived from "Quick Response")
* High capacity and high density
* Error correcting

A QR Code stores data in both vertical and horizontal directions and hence has more storage capacity per unit area. QR codes are readable from any direction and that data stored inside QR codes can be restored even if the corresponding image is partially damaged.



Figure2: difference between traditional 1D barcode and QR codes

**Advantages of QR codes**

* There is no need to write vital details down. A simple scan captures the desired information.
* QR Codes can be used to store addresses and URLs that can appear in magazines, on signs, buses, business cards or just about any product that users might need information about.

**Disadvantage of QR codes**

* Users must be equipped with a camera phone and the correct reader software that can scan the image of the QR Code. Currently only Smartphone’s are technically equipped to do this. Many users that have mobile phones that have cameras are unable to get QR reading software for their phones.

The following chapters show in detail what we did to create the smart virtual store.

**Chapter 1**:

**QRcode generator**

In this chapter we explain how to create QRcode :

## What is QRcode

### 1.1.1- Characters which can be encoded in QRcode (encode mode).

### We used 8bit byte data mode , in theory, 2953 characters or less can be stored in a QRcode by by using this mode .

### 1.1.2- Error correcting in QRcode.

QR code has a function of an error correcting for miss reading that white is black. Error correcting is defined in 4 levels :

* level L : About 7% or less errors can be corrected.
* level M : About 15% or less errors can be corrected.
* level Q : About 25% or less errors can be corrected.
* level H : About 30% or less errors can be corrected.

we choose level Q for our own generator.

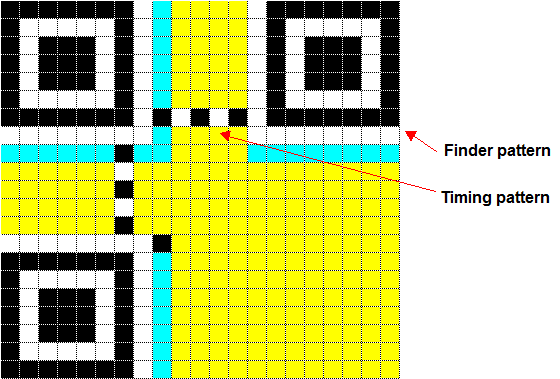
**1.1.3- Version in QR code.**

Size of QRcode is defined as version.  
Version is from 1 to 40, we used in the project version 4.  
Version 4 is 33\*33 matrix. And 4 modules increases whenever 1 version increases.

### 1.1.4- Structure of QR code symbol.

In figure 3 below, white or black parts are fixed in specification. They are "finder pattern" and "timing pattern".  
"Finder pattern": is used to help detection a position of QRcode in decoder application.  
"Timing pattern": is used to help determine a symbol's coordinate in decoder application.

The yellow parts, encoded data (including error correct code) are stored.  
And in cyan color parts, information of error correcting level and mask pattern(described later) are stored .This is called format information.



■encoded data ■format information

Figure 1.1: Structure of QR code symbol

In addition, version2 or higher has "alignment pattern" which is used to correct skewness in decoder application.

And version 7 or higher has "version information" which has information of version itself.

## 1.2  How To Encode

### 1.2.1 Capacity

Capacity of a QR code is determined by version ,error correcting level and encoding mode .

For example, in version 4 and error correcting level Q, 46 byte data can be stored.( See appendix A)

### 1.2.2 Encode to data code words.

8bit data is treated as a code word. We calculate or put data in this unit. In this section ,"how to transform input data to code words" is described.  
  
Now we think to encode example source data "ABCDE123" to QR code in version4 with error correcting level Q using byte mode .

#### Mode indicator

First, mode indicator is created in 4 bit long as binary representation.

* numeric mode : 0001
* alphanumeric mode : 0010
* 8bit byte mode: 0100

In our case, we select byte mode, so we have  
  
**0010**

#### Character count indicator

Character count indicator is character counts stored in each mode.

* numeric : 10bit long
* alphanumeric : 9bit long
* 8bit byte : 8bit long

In our case, data have 8 characters; we encode 8 in 8 bit long binary representations for byte mode.  
  
0100 **00001000** 

* **Encoding data in binary representation**

In 8bit byte mode, each value is directly encoded in 8bit long binary representation.

“ABCDE123”

0100 **00001000 00001010 00001011 00001100 00001101 00001110 00000001 00000010 00000011**

#### Terminator

We add 0000 to result data in section 2.2.3. When length of encoded data is full in this version and error correcting level, terminator is not needed.  
0100 **00001000 00001010 00001011 00001100 00001101 00001110 00000001 00000010 00000011 0000**

#### Encode to code words

Result data in section 2.2.4 are delimited by 8bit.  
  
**01000000 10000000 10100000 10110000 11000000 11010000**

**11100000 00010000 00100000 00110000**

If last data length is less than 8, padded 0.

If count of code words is less than symbol's capacity (Appendex A) ,then we alternately put "11101100" and "00010001" until full capacity.

Capacity of 4-Q is 48 data code words

We repeatedly put "11101100" and "00010001" until reach the count of code words to 48

**01000000 10000000 10100000 10110000 11000000 11010000 11100000 00010000 00100000 00110000 11101100 00010001……………….. 11101100 00010001**

To decimal representation .

64 128 160 176 192 208 224 16 32 48 236 17 …………. 236 17

These are the data code words of the example .

### 1.2.3 Calculating error correcting code words

QR codes include error correction code words. These blocks of redundant data ensure that the QR code can still be read even if a portion of it is unreadable. Generating them is complicated, so we will attempt to explain it in simple terms.

QR codes use **Reed-Solomon error correction.**

To generating the error correction code words:

**Step 1 :** is to find out how many words you need to generate for your chosen QR version and error correction level.

In our case (version 4, Q EC level ) needs 52 EC code word .(see appebdix A)

**Step 2 : Create message polynomial**

Reed-Solomon error correction uses polynomials, so we construct a polynomial--called the **message polynomial**-- that uses the data code words that we generated in 2.1 part . We created 48 data blocks in the previous step, so our message polynomial will have 48 terms in it. Each data code word will be the coefficient of each term.

**Message polynomial coefficients**:

64 128 160 176 192 208 224 16 32 48 236 17 …………. 236 17

**Message polynomial**   
64x99+ 128x98+ 160x97+ 176x96+ 192x95+ 208x94+ 224x93+ 16x92+ 32x91+ 48x90+ 236x89+ 17x88+………..+ 236x47+ 17x48.

The exponent of the first term is : (number of data code words)+(number of EC code words) -1

In our case, this is 48 + 52-1 = 99. So, the first term of our polynomial is 64x99.

**Step 3: Create generator polynomial**

* **Select g(x) according count of error correcting code words.**

**Generator polynomial for 52 error correction code words:**

**x52+ α116x51+ α50x50+ α86x49+ α186x48+ α50x47+ α220x46+ α251x45+ α89x44+ α192x43+ α46x42+ α86x41+ α127x40+ α124x39+ α19x38+ α184x37+ α233x36+ α151x35+ α215x34+ α22x33+ α14x32+ α59x31+ α145x30+ α37x29+ α242x28+ α203x27+ α134x26+ α254x25+ α89x24+ α190x23+ α94x22+ α59x21+ α65x20+ α124x19+ α113x18+ α100x17+ α233x16+ α235x15+ α121x14+ α22x13+ α76x12+ α86x11+ α97x10+ α39x9+ α242x8+ α200x7+ α220x6+ α101x5+ α33x4+ α239x3+ α254x2+ α116x + α51**

We get this function by Generator Polynomial Tool available on internet

* **Divide the message polynomial by the generator polynomial**

To make this step easier, we multiply the generator polynomial by x47 so that its first term has x99 to match the message polynomial's first term, which also has x99.

**Generator polynomial multiplied by x47**

**X99+ α116x98+ α50x97+ α86x96+ α186x95+ α50x94+ α220x93+ α251x92+ α89x91+ α192x90+ α46x89+ α86x88+ α127x87+ α124x86+ α19x85+ α184x84+ α233x83+ α151x82+ α215x81+ α22x80+ α14x79+ α59x78+ α145x77+ α37x76+ α242x75+ α203x74+ α134x73+ α254x72+ α89x71+ α190x70+ α94x69+ α59x68+ α65x67+ α124x66+ α113x65+ α100x64+ α233x63+ α235x62+ α121x61+ α22x60+ α76x59+ α86x58+ α97x57 + α39x56+ α242x55+ α200x54 + α220x53+ α101x52+ α33x51+ α239x50+ α254x49+ α116x48 + α51x47**

* **multiply the generator polynomial by the first coefficient of the message polynomial**

First, we multiply the generator polynomial by the first coefficient of the message polynomial, which is 64. We convert 64 to its alpha equivalent, α6(see appendix A) and multiply each coefficient by it, as shown below.

**α6X99+ α6α116x98+ α6α50x97+ α6α86x96+ α6α186x95+ α6α50x94+ α6α220x93+ α6α251x92+ α6α89x91+ α6α192x90+ α6α46x89+ α6α86x88+ α6α127x87+ α6α124x86+ α6α19x85+ α6α184x84+ α6α233x83+ α6α151x82+ α6α215x81+ α6α22x80+ α6α14x79+ α6α59x78+ α6α145x77+ α6α37x76+ α6α242x75+ α6α203x74+ α6α134x73+ α6α254x72+ α6α89x71+ α6α190x70+ α6α94x69+ α6α59x68+ α6α65x67+ α6α124x66+ α6α113x65+ α6α100x64+ α6α233x63+ α6α235x62+ α6α121x61+ α6α22x60+ α6α76x59+ α6α86x58+ α6α97x57 + α6α39x56+ α6α242x55+ α6α200x54 + α6α220x53+ α6 α101x52+ α6α33x51+ α6α239x50+ α6α254x49+ α6α116x48 + α6α51x47**

To multiply the alphas, we just add the exponents together. For example, the second term has α6\*α116 The exponents are 6 and 116 . 6+116 = 122 so α6\*α228= α122 = 236

***64***X99+ ***236***x98 +***93*** x97+ ***91***x96+ ***130***x95+ ***93***x94+ ***72***x93+ ***97*** x92+226x91+ ***7***x90+ 20 x89+ ***91***x88+ ***109*** x87+ ***46*** x86 + ***3*** x85+ ***174***x84+ ***203***x83+ ***228*** x82+ ***69***x81+ ***24***x80+ ***180***x79+ ***190*** x78+ ***170***x77+ ***119*** x76+ ***27*** x75+ ***162*** x74+ ***132*** x73+32x72+ ***226***x71+ ***200***x70+ ***17*** x69+ ***190*** x68+ ***188***x67+ ***46*** x66+ ***147*** x65+ ***52***x64+ ***22*** x63+88 x62+ ***204*** x61+ ***24***x60+ ***211***x59+ ***91*** x58 + ***136***x57+ ***193*** x56 + ***233*** x55+ ***83*** x54+ ***72*** x53+ ***104*** x52 + ***53*** x51**+ 233x50+ 32x49+ 236x48 + 186x47**

* **XOR the resulting coefficient with the message polynomial’s coefficient**

**Message polynomial**

64x99+ 128x98+ 160x97+ 176x96+ 192x95+ 208x94+ 224x93+ 16x92+ 32x91+ 48x90+ 236x89+ 17x88+………..+ 236x47+ 17x48.

For example: 64 *⊕* 64=0 , ***236****⊕* 128 =108 ,………..

**XOR result**:

108 x98+159x97+235 x96+……. and so on .

* **Repeat, using the result from previous step**.

Now, we repeat the process. First, we multiply the generator polynomial x by to get its lead term to match the x99 of the new result.

Then we take the first coefficient of the result from the previous step, which is 108. This is equal to α250. We multiply the generator polynomial by α250.

number of error correction code words=52.

After doing that, we XOR the result with the message polynomial coefficient as before. We repeat this process until the lead term of our result starts with x51, i.e when the final term of the polynomial has x0 (which is equal to 1) we are done. The coefficients of the polynomial are the error correction code words

A x51+ B x51+ C x50+……… + D.

The coefficients of the above polynomial are our error correction code words.

A,B, C,…………………,D

* **Put the error correction code words after the data code words**

Now that we have the error correction code words, we put them after the data code words that we created before .

64 128 160 176 192 208 224 16 32 48 236 17 …………. 236 17 **A ,B, C,…………………,D.**

* **Convert to binary.**

These data and Ec code words converted to Binary in order to put them in the QR code.

### 1.2.4 Data allocation

1 module means 1 bit. Result data in previous section are encoded to binary representation, and we allocate these encoded data.

We allocate fixed pattern which is "finding pattern" and "timing pattern" in advance.   
  
**Rule for allocation**

1. Now we think coordinate which has i lines j columns, and upper left corner is (0,0).  
For example, in version 4 (33 x 33) this has modules from (0,0) to (32,32).  
  
2. Start module is lower right corner.  
In example data (4-Q), start module is (32,32), and we put a data (0 or 1).  
  
3. Direction of movement (upper or lower) is kept. Direction is upper in first.  
  
4. We think 2 modules width.  
If we are in right module of 2 modules width...  
If left module is blank (not fixed pattern or version information etc), we move left module and put data.  
If left module is not blank, we move in direction which is kept, and put data  
  
If we are in left module....  
We check that blank module is in direction which is kept. If blank module is, we put data in right module in priority to left module of 2 modules width.  
If blank module is not,we move to a left module, and put data there.Then we turn  
direction which is kept.

If we have data "01234567 89ABCDEF GHIJKLMN"

|  |  |  |  |
| --- | --- | --- | --- |
| D | C | B | A |
| F | E | 9 | 8 |
| H | G | 7 | 6 |
| J | I | 5 | 4 |
| L | K | 3 | 2 |
| N | M | 1 | 0 |

In same data, fixed pattern "\*" is existed in 4\*2 center..

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | 8 | 7 | 6 |
| A | \* | \* | 5 |
| B | \* | \* | 4 |
| C | \* | \* | 3 |
| D | \* | \* | 2 |
| F | E | 1 | 0 |

### 

### Below figure is put finder pattern,timing pattern and code words in example data .

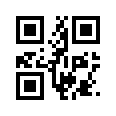


Figure1.2

### 1.2.5 Mask pattern

If one color modules are much than other color or pattern like "finder pattern" exists, decoder application can have many mistake. To prevent above case, best mask pattern which is selected from among 8 pattern is covered in QRcode.

#### Condition of masking

We have 8 mask pattern in QRcode. Mask pattern indicator is 3 bit long binary representation.

**We choose any pattern**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **mask pattern indicator** | **condition** | | 011 | (i+j) mod 3 = 0 | |  |

### 1.2.6 Format information

Format information includes error correcting level and mask pattern indicator in 15 bit long.  
First 2 bit are error correcting level

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **error correcting level** | **indicator** | | Q | 11 | |  |

 In next 3 bit, we put mask pattern indicator which is selected in previous section.  
  
 11　**011**

We put error correcting data which is Bose-Chaudhur Hocquenghem(BCH)(15,5) in right 10 bit.  
  
First, polynomial F(x) which coefficients are above 5 bit and x10 times is divided by below G(x).   
  
G(x)=x10+x8+x5+x4+x2+x+1  
  
In example data,5 bit data is "11011" and F(x) is below. F(x)=x14+x13+x11+x10

divide by G(x)....

Remainder R(x)= x9+x4+x2

So we get result as 11011 **1000010100**

Finally, we calculate exclusive logical sum "101010000010010" and above result to avoid that result data is not all 0.   
  
Format information is "011101000000110".

We put results in number of below figure.

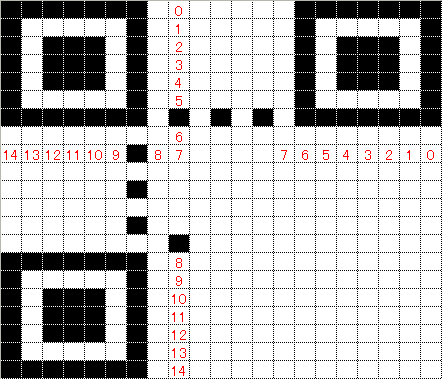


figure 1.3: put format information in QRcode

**1.3 C# Application**

In our project we select encode mode byte , error correction (Q ), version (4). And build a C# desktop application that can take a text data as input   
and output QRcode image of that input after make all previous operations on data to produce qrcode .



ssi

figure 1.4: our C# qrcode generator

We made Qrcode image for each product in store, qrcode encode information about this product like : name, price, size .



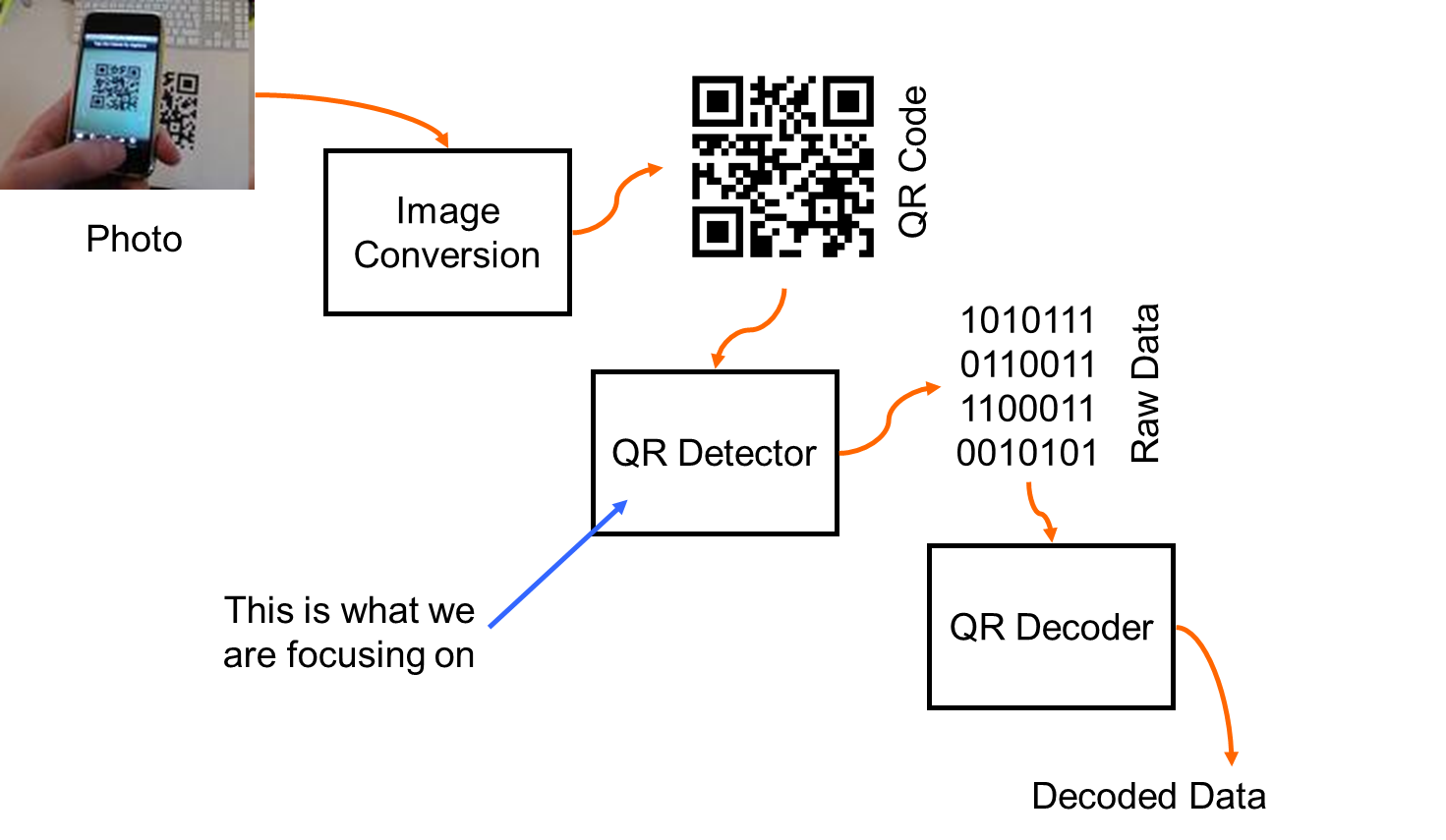
Figure 1.5 : how it would be in Virtual Store

**Chapter 2**:

**QRcode Reader**

The QR Code reader used to decode a QRCode image generated via the generator we made , or any generated image by other QR generators .

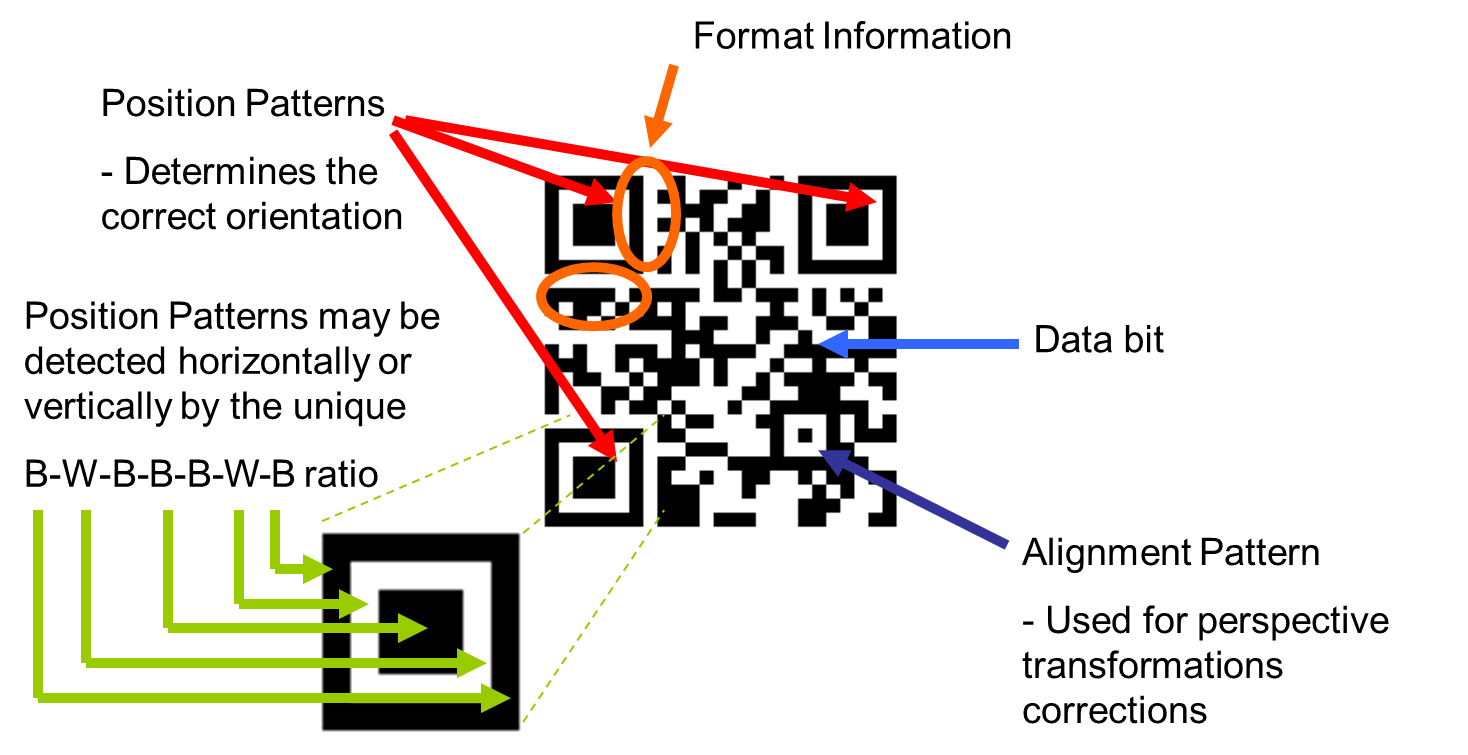
**2.1 Components in QR code reader:**



**Figure 1.1**

We note that the input to the reader is Image taken by Mobile  
the Output is the content text in that image .

**2.2 QR code properties:**



**Figure1.2**

**2.3 Operations in detail:**

The operation of decoding is pure code and we build many functions to do this, here we will describe the operations on image in general and to understand deeply, you can return to the code.

The decoder take a bitmap image and determine some fixed points in that image, then start decoding the image with respect to these points by converting the bitmap image to integer array then to gray scale image and get the finder pattern (position) and alignment pattern in the image and construct a grid of 0’s and 1’s then get the rest of image as data matrix

After the reading of all bits we get all data and RS blocks of data .  
Then using Reed Solomon Algorithm we extract the data blocks only from all blocks and finally we get the decoded bytes from each block with respect to QR code version and error correction level we use ,each of these bytes converted to ASCII string in order to Output this string as the output of reading this QR code image .

Each of previous operations is made by a function in the code.

**2.4 C# Application :**

We build a C# desktop application reader at first, and after It works correctly we convert it to java classes in the mobile application to produce a mobile QR code reader

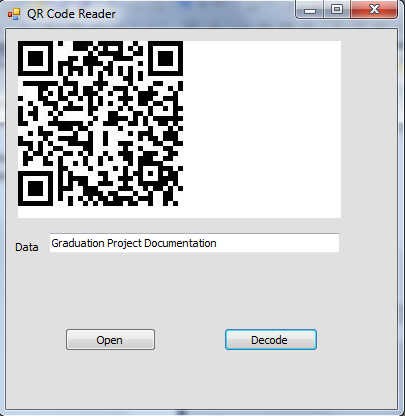


Figure 1.3: C# QRcode Reader

**Chapter 3**:

**Mobile Application**

This chapter shows how we construct the **j2me** mobile application

**“Smart Virtual Store**”.  
  
With our mobile application customer can capture image of product’s QRcode he want to purchase ,select the quantity needed and time of delivery , determine his address and send order to server ,then reply message from server arrived .

We used NetBeans program to write the j2me code of our application .

**3.1 Mobile application steps :**

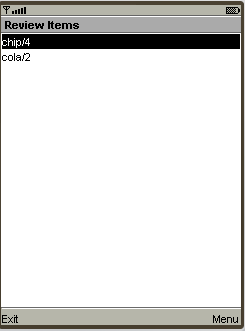
1. capture image with specified Resolution .



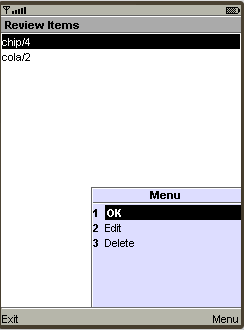
1. using QR Decoder (same QR Reader which built in C #)  
   QR Code decoder decode image and return the result .



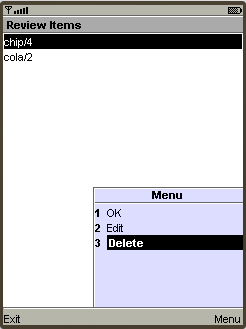
1. Repeat previous steps for each additional product needed .for each capture show result of decoding, and show the product picture, name of product, and price.
2. After capture images for all items, review a list shows customer choices of products and their prices.



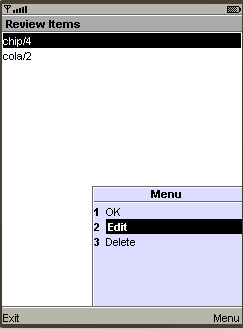
1. For each item in list customer can do actions (edit, delete ).



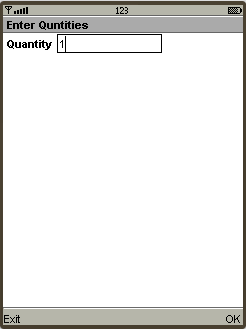
1. if you want to delete product after choosing it, you can delete from this form .



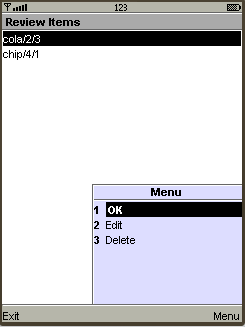
1. If you still want to buy this product select edit.



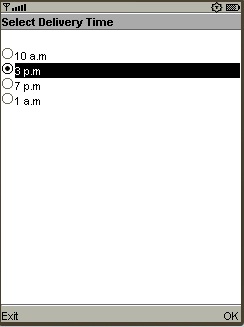
1. Here you enter the quantity of product ,then ok .



1. You can see list of (product/price/quantity ).



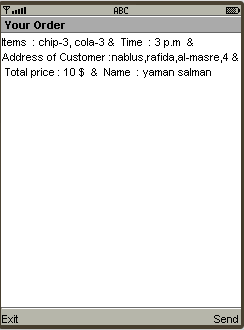
1. After select all products and their quantities, for each product select delivery time.



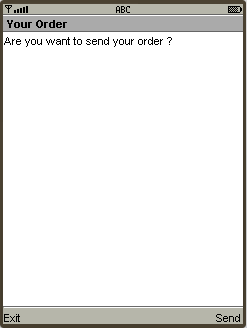
1. Then enter your address (city/street/building/ home number) and your name.



1. Application collect all information you entered in an order (items, time, address ,quantity ,your name ),but this message don’t appear to you .



1. Send message to server using package MESSEGING .MESSEGING CONNECTION ,then select send command .



**3.2 Payment :**

From the mobile application ,customer can pay the cost of his order by using his paypal account , account linked to visa card or bank account, application reach url for the store’s account on paypal website ,so customer can cash money on store’s account .

Now we are working on paypal sandbox accounts , in order to pay virtually without visa card , and we wish to successfully finish this part .

**3.3 Some of difficulties in this part :**

QRcode reader in mobile application faced some problems in scanning the image taken by mobile, so some times it can’t return a correct result for reading.

In order to avoid this problem we try capture the image in a better elimination environment, and using smart phone’s camera with higher resolution (5 Mpixel), we notice that this circumstances reduces the problem.

**Chapter 4**:

**Communication Between Customer and Server.**

We use short messages service ( SMS) as a communication tool from the customer mobile (using our j2me app) to **the server mobile** and vice versa.

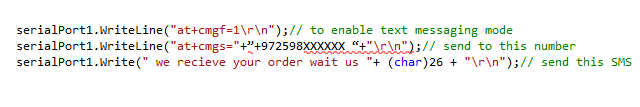
**4.1 Server mobile ?!**

Yes, In order to make messages reach the Server (pc) we connect a mobile to pc using Bluetooth (serial port) and this mobile will receive all messages that comes to the server from customers .

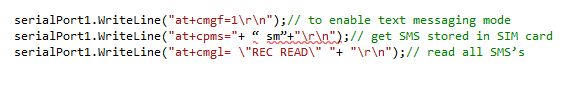
The server application can read this mobile messages and also send messages to customers via this mobile from the pc itself by using AT Commands. We handle all these operations in A C# application.

**4.2 AT commands:**

For send message, write this code:



For read messages stored on the mobile, write this code:



Of course, these codes execute correctly after we connect the mobile with server laptop via Bluetooth ,and open the port of this connection .

**4.3 Some of the difficulties in this part:**

We face a problem of choosing a type of mobiles that AT commands implemented correctly on, we spend a lot of time in trying so many types , but at the end we realize that we need a Nokia phone with series 40 , Precisely Nokia 6320i .



Figure4.1: Nokia 6320i

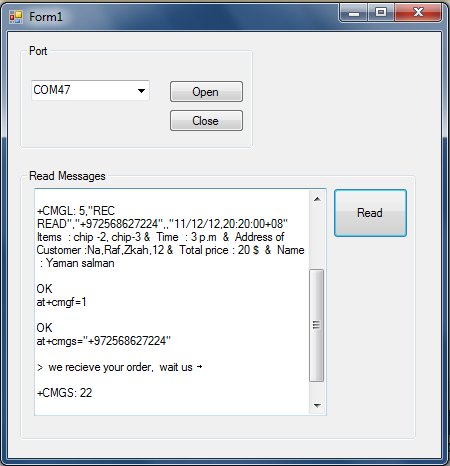
**Chapter 5**:

**Server Part of Smart Virtual Store.**

**5.1 Application**

It is C# application placed in the market machine to serve requests from customers when they send their orders via their phones.

After a message reach to the server ,this message contains the items the customer order and basic info about customer ,server will store this info in its data base and after make sure that customer ‘s order is stored ,server will send an acknowledgment message to customer mobile that tell him to wait the items .

  
figure5.1: C# Application on server part

In this figure we see the application that read messages of server mobile from the pc .

and also we see AT commands of read message, also of send messages to customer mobile number who send the a request to the server .

**Chapter 6**:

**Smart Virtual Store Website.**

By now, we are working in a website using ASP.net .

Customers can sign up for a personal accounts, put their information needed by the application, so they be able to skip some of the data entry steps in the application. Just by sign in to the account from the mobile application then start purchasing items , website will continue the payment process automatically based on paypal account of user stored in his Smart Virtual Store account .

Aslo customer can sign in to his profile and view his Purchases during a specified period of time , or since he register in the Store .

Additional features may be added to the website.

**Conclusion**

399 hours and 46 minutes. This is the amount of time spent by the average person shopping in one year. Today, we live in a world of fast paced. People do not have time to spare. Thus, we yearn for comfort. Most of us are too busy, and we do not have enough time even to go to the grocery store.

The **Smart Virtual Store** is QRcode scanning products displayed in the train station for example. Imagine walking out of a train and the discovery of what appears to be the grocery shelf, with pictures of milk, apples, rice, and any other items you would find in the grocery store. All of these images carry a small barcode underneath. After downloading our application on your smartphone, you can scan the item you want. Product image then appears on the screen of the phone. The application then asks when and where you want to be delivered, followed by the billing information. When you arrive home later that day, it's right there waiting for you.

We think that the **Smart Virtual Store** will go to the usage and development.

**Appendices**

Appendix A : Tables and Figures

Table1:Some of Qrcode Versions

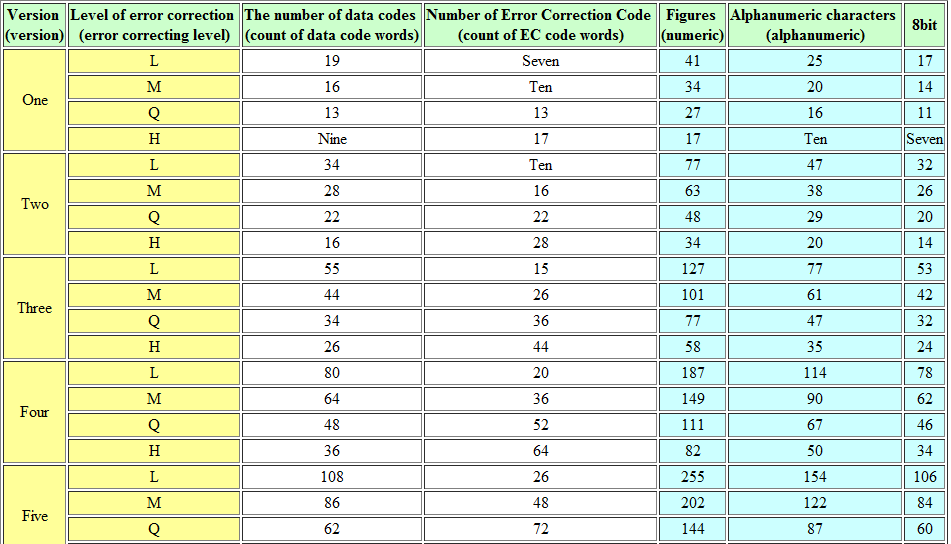
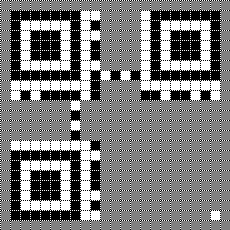


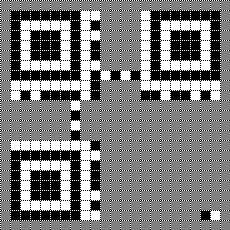
Table2: Log Antilog Table for Galois Field 256

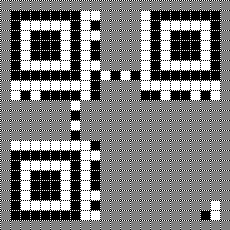
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Exponent of α** | **Integer** |  | **Integer** | **Exponent of α** |
| 0 | 1 |  |  |  |
| 1 | 2 |  | 1 | 0 |
| 2 | 4 |  | 2 | 1 |
| 3 | 8 |  | 3 | 25 |
| 4 | 16 |  | 4 | 2 |
| 5 | 32 |  | 5 | 50 |
| 6 | 64 |  | 6 | 26 |
| 7 | 128 |  | 7 | 198 |
| 8 | 29 |  | 8 | 3 |
| 9 | 58 |  | 9 | 223 |
| 10 | 116 |  | 10 | 51 |
| 11 | 232 |  | 11 | 238 |
| 12 | 205 |  | 12 | 27 |
| 13 | 135 |  | 13 | 104 |
| 14 | 19 |  | 14 | 199 |
| 15 | 38 |  | 15 | 75 |
| 16 | 76 |  | 16 | 4 |
| 17 | 152 |  | 17 | 100 |
| 18 | 45 |  | 18 | 224 |
| 19 | 90 |  | 19 | 14 |
| 20 | 180 |  | 20 | 52 |
| 21 | 117 |  | 21 | 141 |
| 22 | 234 |  | 22 | 239 |
| 23 | 201 |  | 23 | 129 |
| 24 | 143 |  | 24 | 28 |
| 25 | 3 |  | 25 | 193 |
| 26 | 6 |  | 26 | 105 |
| 27 | 12 |  | 27 | 248 |
| 28 | 24 |  | 28 | 200 |
| 29 | 48 |  | 29 | 8 |
| 30 | 96 |  | 30 | 76 |
| 31 | 192 |  | 31 | 113 |
| 32 | 157 |  | 32 | 5 |
| 33 | 39 |  | 33 | 138 |
| 34 | 78 |  | 34 | 101 |
| 35 | 156 |  | 35 | 47 |
| 36 | 37 |  | 36 | 225 |
| 37 | 74 |  | 37 | 36 |
| 38 | 148 |  | 38 | 15 |
| 39 | 53 |  | 39 | 33 |
| 40 | 106 |  | 40 | 53 |
| 41 | 212 |  | 41 | 147 |
| 42 | 181 |  | 42 | 142 |
| 43 | 119 |  | 43 | 218 |
| 44 | 238 |  | 44 | 240 |
| 45 | 193 |  | 45 | 18 |
| 46 | 159 |  | 46 | 130 |
| 47 | 35 |  | 47 | 69 |
| 48 | 70 |  | 48 | 29 |
| 49 | 140 |  | 49 | 181 |
| 50 | 5 |  | 50 | 194 |
| 51 | 10 |  | 51 | 125 |
| 52 | 20 |  | 52 | 106 |
| 53 | 40 |  | 53 | 39 |
| 54 | 80 |  | 54 | 249 |
| 55 | 160 |  | 55 | 185 |
| 56 | 93 |  | 56 | 201 |
| 57 | 186 |  | 57 | 154 |
| 58 | 105 |  | 58 | 9 |
| 59 | 210 |  | 59 | 120 |
| 60 | 185 |  | 60 | 77 |
| 61 | 111 |  | 61 | 228 |
| 62 | 222 |  | 62 | 114 |
| 63 | 161 |  | 63 | 166 |
| 64 | 95 |  | 64 | 6 |
| 65 | 190 |  | 65 | 191 |
| 66 | 97 |  | 66 | 139 |
| 67 | 194 |  | 67 | 98 |
| 68 | 153 |  | 68 | 102 |
| 69 | 47 |  | 69 | 221 |
| 70 | 94 |  | 70 | 48 |
| 71 | 188 |  | 71 | 253 |
| 72 | 101 |  | 72 | 226 |
| 73 | 202 |  | 73 | 152 |
| 74 | 137 |  | 74 | 37 |
| 75 | 15 |  | 75 | 179 |
| 76 | 30 |  | 76 | 16 |
| 77 | 60 |  | 77 | 145 |
| 78 | 120 |  | 78 | 34 |
| 79 | 240 |  | 79 | 136 |
| 80 | 253 |  | 80 | 54 |
| 81 | 231 |  | 81 | 208 |
| 82 | 211 |  | 82 | 148 |
| 83 | 187 |  | 83 | 206 |
| 84 | 107 |  | 84 | 143 |
| 85 | 214 |  | 85 | 150 |
| 86 | 177 |  | 86 | 219 |
| 87 | 127 |  | 87 | 189 |
| 88 | 254 |  | 88 | 241 |
| 89 | 225 |  | 89 | 210 |
| 90 | 223 |  | 90 | 19 |
| 91 | 163 |  | 91 | 92 |
| 92 | 91 |  | 92 | 131 |
| 93 | 182 |  | 93 | 56 |
| 94 | 113 |  | 94 | 70 |
| 95 | 226 |  | 95 | 64 |
| 96 | 217 |  | 96 | 30 |
| 97 | 175 |  | 97 | 66 |
| 98 | 67 |  | 98 | 182 |
| 99 | 134 |  | 99 | 163 |
| 100 | 17 |  | 100 | 195 |
| 101 | 34 |  | 101 | 72 |
| 102 | 68 |  | 102 | 126 |
| 103 | 136 |  | 103 | 110 |
| 104 | 13 |  | 104 | 107 |
| 105 | 26 |  | 105 | 58 |
| 106 | 52 |  | 106 | 40 |
| 107 | 104 |  | 107 | 84 |
| 108 | 208 |  | 108 | 250 |
| 109 | 189 |  | 109 | 133 |
| 110 | 103 |  | 110 | 186 |
| 111 | 206 |  | 111 | 61 |
| 112 | 129 |  | 112 | 202 |
| 113 | 31 |  | 113 | 94 |
| 114 | 62 |  | 114 | 155 |
| 115 | 124 |  | 115 | 159 |
| 116 | 248 |  | 116 | 10 |
| 117 | 237 |  | 117 | 21 |
| 118 | 199 |  | 118 | 121 |
| 119 | 147 |  | 119 | 43 |
| 120 | 59 |  | 120 | 78 |
| 121 | 118 |  | 121 | 212 |
| 122 | 236 |  | 122 | 229 |
| 123 | 197 |  | 123 | 172 |
| 124 | 151 |  | 124 | 115 |
| 125 | 51 |  | 125 | 243 |
| 126 | 102 |  | 126 | 167 |
| 127 | 204 |  | 127 | 87 |
| 128 | 133 |  | 128 | 7 |
| 129 | 23 |  | 129 | 112 |
| 130 | 46 |  | 130 | 192 |
| 131 | 92 |  | 131 | 247 |
| 132 | 184 |  | 132 | 140 |
| 133 | 109 |  | 133 | 128 |
| 134 | 218 |  | 134 | 99 |
| 135 | 169 |  | 135 | 13 |
| 136 | 79 |  | 136 | 103 |
| 137 | 158 |  | 137 | 74 |
| 138 | 33 |  | 138 | 222 |
| 139 | 66 |  | 139 | 237 |
| 140 | 132 |  | 140 | 49 |
| 141 | 21 |  | 141 | 197 |
| 142 | 42 |  | 142 | 254 |
| 143 | 84 |  | 143 | 24 |
| 144 | 168 |  | 144 | 227 |
| 145 | 77 |  | 145 | 165 |
| 146 | 154 |  | 146 | 153 |
| 147 | 41 |  | 147 | 119 |
| 148 | 82 |  | 148 | 38 |
| 149 | 164 |  | 149 | 184 |
| 150 | 85 |  | 150 | 180 |
| 151 | 170 |  | 151 | 124 |
| 152 | 73 |  | 152 | 17 |
| 153 | 146 |  | 153 | 68 |
| 154 | 57 |  | 154 | 146 |
| 155 | 114 |  | 155 | 217 |
| 156 | 228 |  | 156 | 35 |
| 157 | 213 |  | 157 | 32 |
| 158 | 183 |  | 158 | 137 |
| 159 | 115 |  | 159 | 46 |
| 160 | 230 |  | 160 | 55 |
| 161 | 209 |  | 161 | 63 |
| 162 | 191 |  | 162 | 209 |
| 163 | 99 |  | 163 | 91 |
| 164 | 198 |  | 164 | 149 |
| 165 | 145 |  | 165 | 188 |
| 166 | 63 |  | 166 | 207 |
| 167 | 126 |  | 167 | 205 |
| 168 | 252 |  | 168 | 144 |
| 169 | 229 |  | 169 | 135 |
| 170 | 215 |  | 170 | 151 |
| 171 | 179 |  | 171 | 178 |
| 172 | 123 |  | 172 | 220 |
| 173 | 246 |  | 173 | 252 |
| 174 | 241 |  | 174 | 190 |
| 175 | 255 |  | 175 | 97 |
| 176 | 227 |  | 176 | 242 |
| 177 | 219 |  | 177 | 86 |
| 178 | 171 |  | 178 | 211 |
| 179 | 75 |  | 179 | 171 |
| 180 | 150 |  | 180 | 20 |
| 181 | 49 |  | 181 | 42 |
| 182 | 98 |  | 182 | 93 |
| 183 | 196 |  | 183 | 158 |
| 184 | 149 |  | 184 | 132 |
| 185 | 55 |  | 185 | 60 |
| 186 | 110 |  | 186 | 57 |
| 187 | 220 |  | 187 | 83 |
| 188 | 165 |  | 188 | 71 |
| 189 | 87 |  | 189 | 109 |
| 190 | 174 |  | 190 | 65 |
| 191 | 65 |  | 191 | 162 |
| 192 | 130 |  | 192 | 31 |
| 193 | 25 |  | 193 | 45 |
| 194 | 50 |  | 194 | 67 |
| 195 | 100 |  | 195 | 216 |
| 196 | 200 |  | 196 | 183 |
| 197 | 141 |  | 197 | 123 |
| 198 | 7 |  | 198 | 164 |
| 199 | 14 |  | 199 | 118 |
| 200 | 28 |  | 200 | 196 |
| 201 | 56 |  | 201 | 23 |
| 202 | 112 |  | 202 | 73 |
| 203 | 224 |  | 203 | 236 |
| 204 | 221 |  | 204 | 127 |
| 205 | 167 |  | 205 | 12 |
| 206 | 83 |  | 206 | 111 |
| 207 | 166 |  | 207 | 246 |
| 208 | 81 |  | 208 | 108 |
| 209 | 162 |  | 209 | 161 |
| 210 | 89 |  | 210 | 59 |
| 211 | 178 |  | 211 | 82 |
| 212 | 121 |  | 212 | 41 |
| 213 | 242 |  | 213 | 157 |
| 214 | 249 |  | 214 | 85 |
| 215 | 239 |  | 215 | 170 |
| 216 | 195 |  | 216 | 251 |
| 217 | 155 |  | 217 | 96 |
| 218 | 43 |  | 218 | 134 |
| 219 | 86 |  | 219 | 177 |
| 220 | 172 |  | 220 | 187 |
| 221 | 69 |  | 221 | 204 |
| 222 | 138 |  | 222 | 62 |
| 223 | 9 |  | 223 | 90 |
| 224 | 18 |  | 224 | 203 |
| 225 | 36 |  | 225 | 89 |
| 226 | 72 |  | 226 | 95 |
| 227 | 144 |  | 227 | 176 |
| 228 | 61 |  | 228 | 156 |
| 229 | 122 |  | 229 | 169 |
| 230 | 244 |  | 230 | 160 |
| 231 | 245 |  | 231 | 81 |
| 232 | 247 |  | 232 | 11 |
| 233 | 243 |  | 233 | 245 |
| 234 | 251 |  | 234 | 22 |
| 235 | 235 |  | 235 | 235 |
| 236 | 203 |  | 236 | 122 |
| 237 | 139 |  | 237 | 117 |
| 238 | 11 |  | 238 | 44 |
| 239 | 22 |  | 239 | 215 |
| 240 | 44 |  | 240 | 79 |
| 241 | 88 |  | 241 | 174 |
| 242 | 176 |  | 242 | 213 |
| 243 | 125 |  | 243 | 233 |
| 244 | 250 |  | 244 | 230 |
| 245 | 233 |  | 245 | 231 |
| 246 | 207 |  | 246 | 173 |
| 247 | 131 |  | 247 | 232 |
| 248 | 27 |  | 248 | 116 |
| 249 | 54 |  | 249 | 214 |
| 250 | 108 |  | 250 | 244 |
| 251 | 216 |  | 251 | 234 |
| 252 | 173 |  | 252 | 168 |
| 253 | 71 |  | 253 | 80 |
| 254 | 142 |  | 254 | 88 |
| 255 | 1 |  | 255 | 175 |

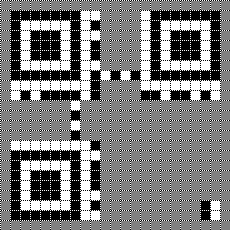
**Add Data Bits :**

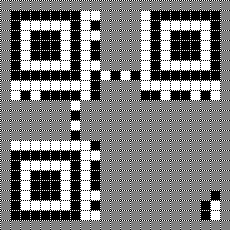
**Upward Column**

  
First pixel added

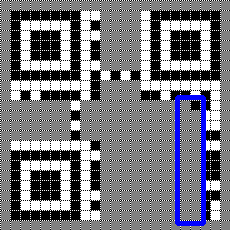
  
Second pixel added

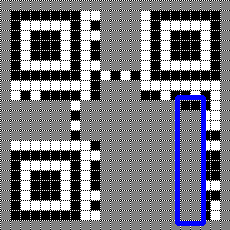
  
Third pixel added

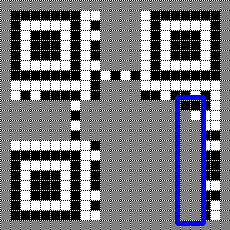
  
Fourth pixel added

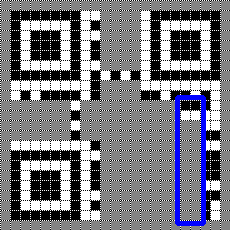
  
Fifth pixel added

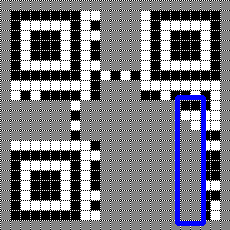
#### Downward Column

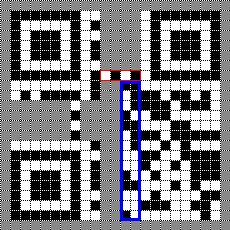
  
Downward pixel #1 added

  
Downward pixel #2 added

  
Downward pixel #3 added

  
Downward pixel #4 added

  
Downward pixel #5 added

Be Sure to Skip the Horizontal Timing Pattern  
When you reach the first timing pattern…

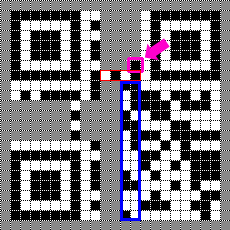
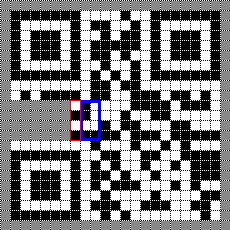
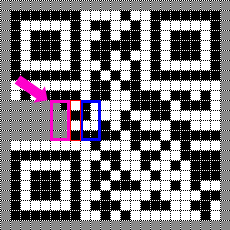
  
…be sure to skip it.

Figure 1 : Add data bits to qrcode

Be Sure to Skip the Vertical Timing Pattern  
When you reach the second timing pattern…

  
…be sure to skip it.

Appendix B : Extra photographs



