UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 3037

QR CODE READING

Matija Bačić

Zagreb, June 2022

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 3037

QR CODE READING

Matija Bačić

Zagreb, June 2022

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zagreb, 11 March 2022

MASTER THESIS ASSIGNMENT No. 3037

Student:	**Matija Bačić (0036507836)**
Study:	Computing
Profile:	Computer Science
Mentor:	assoc. prof. Marko Čupić

Title: **QR code reading**

Description:

QR or "Quick response" codes are matrix variants of barcodes. Today, they are widely used for advertising and inventory tracking in warehouses. As a part of this master's thesis, it is necessary to study suitable approaches for reading the position of the QR code within an image, as well as the mathematical basis of coding and decoding QR codes. It is necessary to develop a prototype system for reading the QR codes. The system should be able to find the QR code in an image, read it and decode it. Examine and comment the effectiveness of the implemented approach given different image illumination. Algorithms, source codes and results should be provided with the necessary explanations and documentation. Cite the literature used and indicate the assistance received.

Submission date: 27 June 2022

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

DIPLOMSKI ZADATAK br. 3037

Pristupnik:	**Matija Bačić (0036507836)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	izv. prof. dr. sc. Marko Čupić

Zadatak: **Očitavanje QR-koda**

Opis zadatka:

QR ili "Quick response" kodovi matrična su varijanta barkodova. Danas su uvelike korišteni za oglašavanje te praćenje inventara u skladištima. U okviru ovog diplomskog rada potrebno je proučiti prikladne pristupe za očitavanje pozicije QR koda na slici te matematičku podlogu kodiranja i dekodiranja QR kodova. Potrebno je razviti prototipni programski sustav za očitavanje QR kodova. Sustav treba moći pronaći QR kod na fotografiji, očitati ga te ga dekodirati. Ispitati i komentirati učinkovitost implementiranog pristupa s obzirom na različito osvjetljenje slike. Radu je potrebno priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 27. lipnja 2022.

Content

Introduction

Barcode is a system of presenting data in a machine-readable form. It was created in search of an efficient method of transferring information from physical media, such as paper, to a computer.

Most widely used types of barcodes are one-dimensional barcodes (also known as linear barcodes) and QR codes, which are a type of two-dimensional barcodes. Linear barcodes can store only small amounts of data, but they don't require any special hardware for encoding or decoding. While they are not resistant to errors or distortions, simplicity makes them favorable.

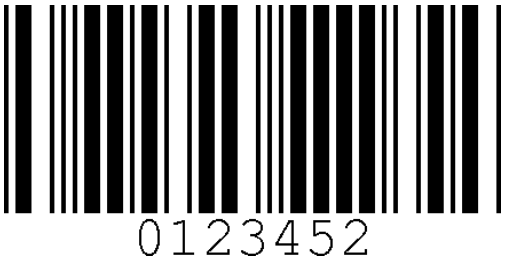

Figure 0.1 Example of a linear barcode in Code11 standard

Figure 0.1 shows an example of a linear barcode. It is made up of vertical black bars of various widths separated by white spaces.

Reading such barcode is a fairly easy procedure. Every character that can be encoded is listed in a reference table of the standard (in case of the Figure 0.1 we are talking about Code11 standard of linear barcodes). Characters are represented by three black bars with space in between them. Reference table thus describes thicknesses of black bars and spaces. It is made such that no two characters can be confused with one another. Table 0.1 is the reference table for Code11 standard of linear barcodes. In said table "0" denotes thin bar/space and "1" indicates thick bar/space.

Table 0.1 Code11 standard

Character	Widths
0	00001
1	10001
2	01001
3	11000
4	00101
5	10100
6	01100
7	00011
8	10010
9	10000
-	00100
Start/Stop	00110

2D barcodes, on the other hand, have an advantage in the amount of data they can store. Furthermore, they can overcome distortions and errors that can occur during a scanning process. Error correction methods help retrieve lost data even if as high as 30% of the barcode is damaged.

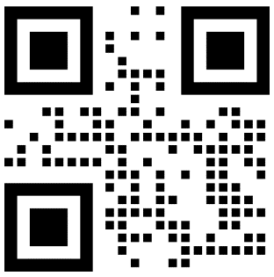

Figure 0.2 Example of a QR code

Most popular of 2D barcodes must be QR codes (Quick response codes). Today, they are widely used for advertising and inventory tracking in warehouses. Figure 0.2 shows an example of a QR code in which we can see a matrix-like structure.

Further study in this thesis will only be based on scanning, encoding and decoding QR codes. In the chapters that follow, version 1 of the QR code will be analyzed extensively while other versions will only be mentioned. The reason behind this decision is

that the only meaningful difference between versions is the size of the QR code i.e., its capacity.

1. QR code standard

QR codes can store different types of data. Official standard proposes 4 different types: numeric, alphanumeric, 8-bit byte data and Kanji data. Maximum capacity of data that a single QR code can store depends on the data type and version of the QR code. Standard proposes forty different versions. Table 1.1 shows maximum capacity of data for version 40 of the QR code.

As the only difference between the versions is the number of blocks (and thus the amount of storable data), I will focus on the version 1. Everything explained here for the version 1 is transferable to higher versions.

Table 1.1 Maximum capacity per data type

Data type	Maximum capacity using version 40-L
Numeric	7089
Alphanumeric	4296
8-bit	2953
Kanji	1817

1.1. QR code structure

1.1.1. Locator helper structures

Whole QR code is located between three finder patterns. Finder pattern is there to help with position detection. In other words, locating a QR code within a picture is done by scanning for finder patterns.

Figure 1.1 Finder pattern

Figure 1.1 shows such finder pattern. Scanning pixel by pixel horizontally and vertically in the middle we get the ratio of black and white pixels 1:1:3:1:1. As this pattern consist of concentric superimposed squares, we can conclude that the same ratio is also valid diagonally. With this information we can not only locate the QR code, but also infer the module size in pixels. Total size of the finder pattern is 7 by 7 modules.

Around the finder pattern there is a one module wide quite region. In other words, there is no information encoded here, there's just an empty white region.

In the event of regional image distortion, alignment patterns are used. They are, similar to finder pattern, three superimposed concentric squares positioned symmetrically around the diagonal running from the top left to bottom right corner. Total size of alignment patterns is 5 by 5 modules. Exact positions of this structures are listed in a table in the standard for every QR code version. Version 1 is exception as it does not have any alignment patterns.

Last structure that helps with determining the version and module size is the timing pattern. It is a 1 module wide strip of alternating black and white modules in row 6 and column 6. Starting and ending module is always black.

All of these structures can be seen on Figure 1.2.

1.1.2. Data and error correcting symbols position

Each data type encodes data to bits (codewords) differently. Some use 8-bit symbols, some symbols are less than 8 bits long. Numeric and alphanumeric modes are efficient because they use less bits per symbol, but a drawback is that they can encode only small number of characters. For this reason, I chose to use 8-bit data symbols and encode text message “QR-Code” using ASCII table.

Standard proposes that data stream should start with 4-bit mode indicator (numeric, alphanumeric, 8-bit, kanji…). Indicator for byte encoding is set to be 0100. After that come set number of bits which denote data length. For byte mode and version 1 of the QR code it is standardized as one byte symbol. As our message is 7 characters long, 00000111 is the right symbol for indicating length.

Length of the data length symbol is 1 byte for version 1 through 9 and byte data encoding mode. For higher version 2 bytes are reserved. Once the above-mentioned message is encoded using ASCII table, data stream will look like this (including mode and length indicator): 0100 00000111 01010001 01010010 00101101 01000011 01101111 01100100 01100101.

By looking in the standard, one can determine how long the data stream should be for each version and error correction level. As we will be using version 1-H (version 1 with level H error correction), we need data stream to be exactly 9 bytes (72 bits) long. If the data stream is shorter than required number of codewords, add up to 4 zero bits. Following that rule, we add exactly 4 zeros making our data stream: 0100 00000111 01010001 01010010 00101101 01000011 01101111 01100100 01100101 0000.

In some data modes of the QR code, after the last step, length of the data stream may still not be the right size or not divisible by 8 which we need for error correction. If length is not divisible by 8, we add least possible number of zeros to make it divisible by eight. After that if it is still not the right size, we repeat these two bytes until we reach maximum capacity of the data stream: 11101100 00010001.

Figure 1.2 shows how data stream is positioned within the QR code. Bits of the data are placed starting at bottom right corner. Figure 1.3 shows how bits are positioned within a symbol.

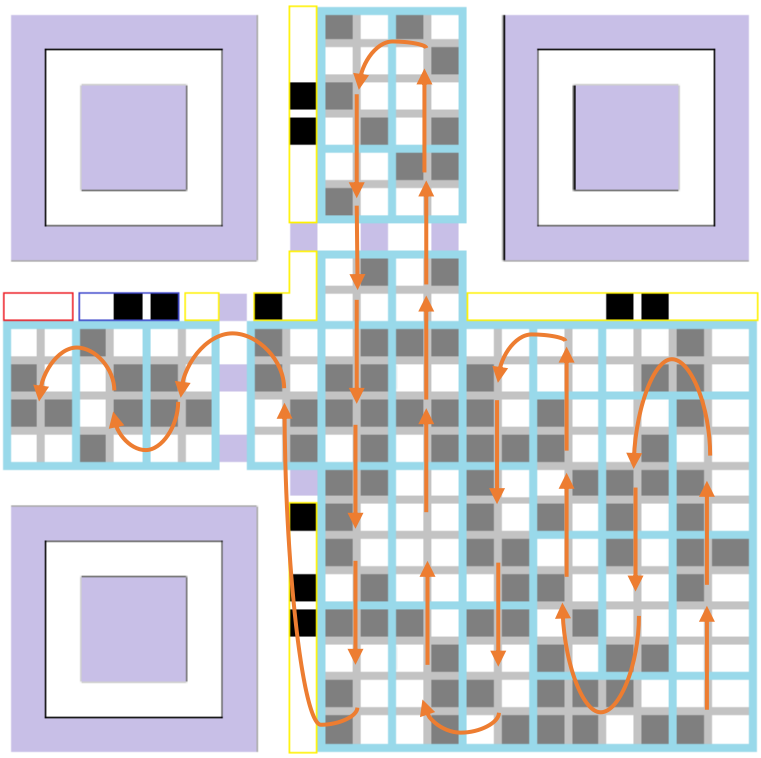

Figure 1.2 QR code structure (purple color – finder pattern, light blue color – data and error correction symbols, red color – error correction level, blue color – mask pattern, yellow color – format error correction, orange arrows – data stream)

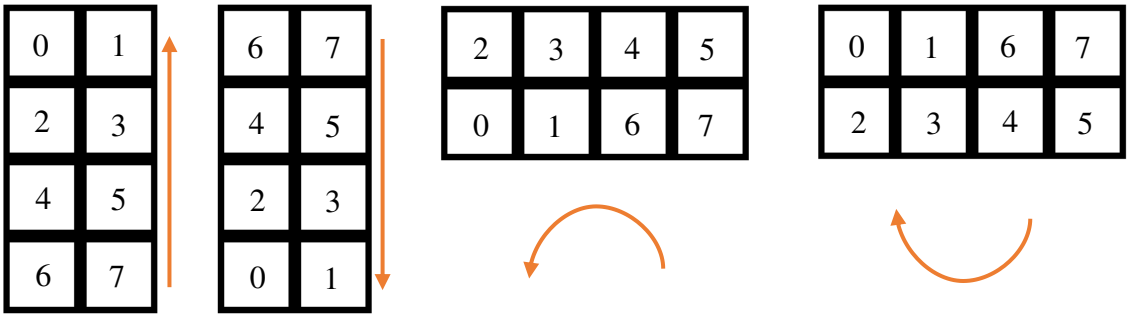

Figure 1.3 Data bits position within a symbol (0 – least significant bit, 7 – most significant bit)

1.1.3. Metadata

Certain metadata needs to be included in the QR code. These are for example masking type and error correction level. In version 2 or higher, a version information is included.

Masking is a process of applying XOR operation between bit pattern in the encoding region with a mask pattern. Goal of this process is to provide more evenly balanced number of dark and light modules. Also, it can reduce occurrence of patterns which can interfere with scanning process.

There are 8 predetermined masking patterns as shown on Figure 1.4 where “i” and “j” represent row and column number respectively (starting from 0). Said figure also shows 3-bit codes for each masking pattern.

Another metadata present in the QR code is error correction level. There are 4 levels of error correction that QR code supports: L, M, Q, H. Error correction level along with QR code version determines how much data can be stored and how many redundant bytes there are. Table 1.2 shows approximately how much data can be restored may the damage occur and also 2-bit codes which denote error correction level.

Now, the question may arise, how does one decide which masking pattern to use. Standard proposes a penalty-based algorithm. Algorithm masks the QR code with all masking patterns, penalizes them based on some undesirable criteria and picks the one with least amount of penalization. Undesirable criteria include adjacent modules in row/column in the same color, block of modules in the same color, 1:1:3:1:1 ratio pattern in row/column and high proportion of dark modules in the whole symbol. In the paper of the standard, penalization points can be found for each of these occurrences. Understandably, some of these features are more unwanted than others so there exist different penalization for each of them.

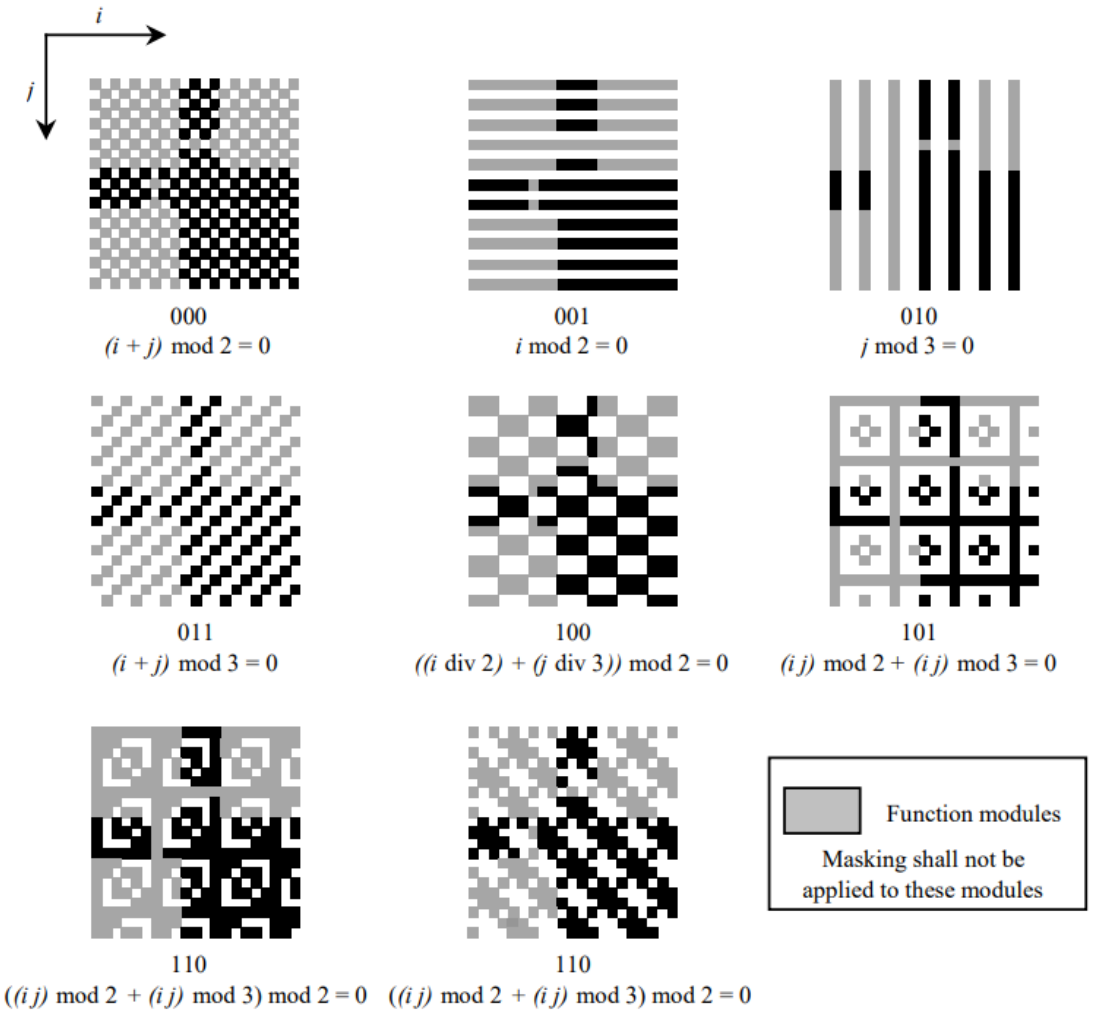

Figure 1.4 Masking patterns [1]

Table 1.2 Error correction levels

Error correction level	Recover capacity (aprox.)	2-bit code
L	7%	01
M	15%	00
Q	25%	11
H	30%	10

2. Error correction

Goal of error correction coding is to introduce redundancy to data so that it can be recovered in case of communication channel imperfections. QR codes use two well-known methods: BCH coding and Reed-Solomon coding.

2.1. Abstract algebra

Abstract algebra is a set of advanced topics that deal with algebraic structures like groups, rings, and fields. It is a basis for the Reed-Solomon coding used in QR codes. In the following subchapters, some of the abstract algebra topics are explained which are prerequisites for understanding the Reed-Solomon encoding.

2.1.1. Number theory

Number theory is a branch of mathematics that studies properties of the set of natural numbers. On that set two operations – addition and multiplication are well defined (meaning that result of these operations on natural numbers are always within set) with these properties:

$$a + b = b + a;\ a * b = b * a \text{ (commutativity)}$$

$$a + (b + c) = (a + b) + c;\ a * (b * c) = (a * b) * c \text{ (associativity)}$$

$$a * (b + c) = ab + ac \text{ (distributivity)}$$

We will first define a few ideas from number theory that will help us later understand group theory.

Definition 1. (Divisibility) We say that a whole number different from zero – a, divides whole number b if there exists whole number z such that b = az. We denote this as a|b.

Theorem 1. For any natural number a and any whole number b, there exist whole numbers q and r such that b = q*a + r.

Definition 2. We say that a is a common divisor of b and c if a|b and a|c for some whole numbers a, b, and c. The largest among them we call the greatest common divisor and write gdc(b, c) = a.

Theorem 2. gcd(a, b) = min({ax + by : x,y ∈ Z} ∩ N)

Definition 3. We say that n different numbers $a_1, \dots, a_n$ are relatively prime if $\gcd(a_1, \dots, a_n) = 1$

Theorem 3. Factorization of every natural number n > 1 is unique up to ordering of prime factors.

Let α(p) and β(p) denote the exponent of prime number p in the factorization of some natural number a and b respectively. Then we can write:

$$\gcd(a, b) = \prod_p p^{\min(\alpha(p),\ \beta(p))}$$

Definition 4. If some whole number m different from zero divides the difference of a-b then we say that a is congruent with b modulo m. We write that as a ≡ b (mod m).

Definition 5. Set $\{x_1, \dots, x_m\}$ is called a complete reminder system modulo m if for every y ∈ Z there exists exactly one x_i for which $y = x_i \pmod m$.

Definition 6. Reduced reminder system modulo m is defined as set of all elements a ∈ {0, 1, …, m} such that gcd(a, m) = 1 and for every whole number y for which gcd(y, m) = 1 there exists exactly one a_i for which $y = a_i \pmod m$.

Every reduced reminder system module m has the same number of elements. That number is denoted as φ(m) and is called Euler's phi function.

Theorem 4. If gcd(a, m) = 1 then $a^{\varphi(m)} \equiv 1\ (mod\ m)$

Question arises how to calculate φ(m) efficiently. We can count all the natural numbers less than m which are relatively prime with m, but here's a better method. Here's one that is obvious: for every prime number p, φ(p) = p – 1. Next definition and theorem will tell us how to do it for every number regardless of prime or not.

Definition 7. Function U: N -> C is a multiplicative function if

1. U(1) = 1,
2. U(ab) = U(a) * (b) for each a,b such that gcd(a, b) = 1

Theorem 5. Euler's function is a multiplicative function and for natural number n>1 holds $\varphi(n) = n \prod_{p|n}(1 - \frac{1}{p})$

Definition 8. Let a and n be relatively prime natural numbers. Smallest number d for which $a^d \equiv 1 \ (mod\ n)$ is called an order of a.

Definition 9. If order of number a modulo n is φ(n) than we say that a is a primitive root modulo n.

We can intuitively conclude that this holds: if a is a primitive root modulo m then a generates whole multiplicative group modulo m. This means that if we take every power of the number a from 0 to φ(m)-1 modulo m, then we get every element of the reduced reminder system modulo m. Proof of this and all other theorems can be found in papers listed in bibliography.

2.1.2. Group theory

Definition 1. Let X be a non-empty set and let there be a binary operation $\alpha : X \times X \to X$. Let that operation be associative. Then, set X along with binary operation α is called a semigroup. We denote this as (X, α)

Definition 2. Semigroup is a monoid if there exists a neutral element $e \in X$ such that $e*x=x*e=x$ for every $x \in X$.

We can then easily see that $(N, +)$ is a semigroup, while $(N \cup \{0\}, +)$ is a monoid.

Theorem 1. If in a semigroup there exists a neutral element, it is unique.

Definition 3. We say that a monoid is a group if for every $x \in X$ there exists x^{-1} such that $x*x^{-1}=x^{-1}*x=e$

Theorem 2. If in a monoid there exists an inverse for some $x \in X$, it is unique.

Definition 4. We say that a group (X, *) is commutative or Abelian if for every $x, y \in X$ holds that $x*y = y*x$.

Definition 5. Let (X, *) be a group. If Y is a subset of X and $y^{-1} \in Y$ and $y_1 * y_2 \in Y$ for every $y_1, y_2 \in Y$. Then we say that (Y, *) is a subgroup of (X, *) and we write $Y \leq X$.

Definition 6. A group generated with one element is called a cyclic group.

For example, we can see that the group (Z, +) is generated with element 1. Because 1 is element of the group, so is it's inverse -1. We also need a neutral element 0. Now applying binary operator + to every element of set {-1, 0, 1} and adding the newly created element to the set, we get set of all whole numbers Z.

Let $Z_p = \{0, \ldots, p - 1\}$ and $*_p$ is multiplication modulo p. Another example then can be group $(Z_p \setminus \{0\}, *_p)$ (where p is a prime number) which is generated with every primitive root modulo p. Because $\varphi(p) = p - 1$, every member of $Z_p \setminus \{0\}$ which has order of $\varphi(p)$ generates whole $Z_p \setminus \{0\}$. We do not include member 0 because it does not have an inverse in Z_p (or even in Z for that matter) so it would not form a group.

Definition 7. Let (Y, *) be a subgroup of (X, *). The left cosets of Y in X are the sets obtained by applying binary operation * between some fixed element of X and every element of Y. I.e., it a left coset is $xY = \{x*y: \text{every } y \in Y\}$ for fixed $x \in X$. Right coset is defined similarly as $Yx = \{y*x: \text{every } y \in Y\}$ for fixed $x \in X$.

Definition 8. Subset $Y \leq X$ is a normal subset it $xY = Yx$ for every $x \in X$. We denote that as $Y \lhd X$.

Notice that for Abelian groups every subgroup is a normal group as the Abelian groups are commutative.

Definition 9. Let N be a normal subgroup of G. Let G/N be defined as set of all left (or right) cosets of N in G. Define binary operation on this set as (aN)(bN)=(ab)N. This binary operation is well defined since a*b must be in G (because G is a group), (ab)N is in G/N. Now G/N with the defined binary operation is a group and is called a quotient group.

For example, $(Z/Z_5, +)$ is a quotient group which divides Z into 4 different sets. Elements in the same set have equal reminder modulo 5.

2.1.3. Ring theory

Definition 1. A ring is defined as (R, +, *) where (R, +) is an Abelian group and (R, *) is a semigroup. Also, distributivity of * in respect to + needs to be satisfied. In other words, x*(y+z) = x*y+x*z and (x+y)*z = x*z + y*z for every x,y,z from R.

Definition 2. Polynomial ring (written as R[t]) is defined as every polynomial whose coefficients are from R and they form a ring a ring (R, +, *). Sum and product of two polynomials is performed by applying operations to their coefficients in the ring R.

Definition 3. (Analogously to subgroups in group theory) Let (R, +, *) be a ring and $I \subseteq R$. If I is a ring with respect to operations from ring R, then we say that I is a subring of R.

Definition 4. (Analogously to normal subgroups in group theory) Continuing on previous definition, if for every $x \in R$, $xI \subseteq I$ and $Ix \subseteq I$, then we say that I is an ideal of R.

Definition 5. (Analogously to quotient groups in group theory) Continuing on previous definition, as (R, +) is an Abelian group, then (I, +) is a normal subgroup so quotient group R/I is well defined. If we introduce multiplication as:

$$(x + I) * (x' + I) = (x*x') + I$$

, then we say that R/I is a quotient ring.

Example of a quotient ring is $Z_m = Z/mZ$.

Definition 6. Let R be a ring with operations + and *. If there exists a case where $x*y=0$ for $x \in R\backslash\{0\}$ and $y \in R\backslash\{0\}$ then we say that x is a divisor of zero. If in a ring there are no divisors of 0, we say that a ring is an integral domain.

Theorem 1. If and only if m is a prime number, Z_m is an integral domain.

2.1.4. Galois field algebra

Definition 1. A field is a commutative integral domain in which every non-zero member has an inverse with respect to multiplication operation. In other words (F, +, *) is a field if:

1) (F, +) is an Abelian group
2) (F, *) is an Abelian group
3) Distributivity: x*(y+z) = x*y+x*z and (x+y)*z = x*z + y*z for $\forall x,y,z \in F$

Theorem 1. Every finite commutative integral domain is a field.

Corollary 1. (From theorem 1 in 2.1.3 and theorem 1 in 2.1.4) If p is a prime number, then Z_p is a field.

Definition 2. Galois field is a finite field with q elements (F_q or GF(q)). If p is prime, we have already seen that we can construct field F_p as Z/pZ. There also exists a field with $q=p^n$ elements.

Theorem 2. If F_q is a field with $q=p^n$ elements, then every member of that field satisfies equation $X^q-X=0$. In other words, set of polynomial decomposition of $X^q-X=0$ over F_p is the field F_q.

Example 1. (Construction of the finite field F_9) As $9=3^2$, members of F_9 are all polynomials $F_3[t]$ with degree less than 2. So, elements are:

$$0, 1, 2, t, t + 1, t + 2, 2t, 2t + 1, 2t + 2$$

Addition is well defined as addition of coefficients modulo 3. Multiplication can be problematic as we can easily fall out of the set. So, we need an irreducible polynomial in $F_3[t]$ of degree 2 so that we can multiply modulo that polynomial. It can be shown that using any irreducible polynomial in $F_3[t]$ of degree 2 well defines multiplication. They are: t^2+1, t^2+t+2, t^2+2t+2. Using any of them, we get a field F_9 as $F_3[t]/(h(t))$ where h(t) is one of those three irreducible polynomials. Why do we need an irreducible polynomial? If the polynomial can be reduced we will get a divisor of zero which will brake the rules of it being a field.

Another question might be why not make F_9 as Z_9. If you try to do that, you will get elements which are not invertible. In other words, in Z_q there are exactly $\varphi(q)$ invertible elements.

Definition 3. Minimal polynomial of some element α over field F[x] is a monic polynomial of the lowest degree in that field such that α is the root of that polynomial. A monic polynomial is a polynomial whose leading coefficient is equal to 1.

Definition 4. It can be shown that for every F_q there exists an irreducible monic polynomial whose root forms a multiplicative group of a finite field F_q. We call that polynomial a primitive polynomial as its roots are primitive elements of the multiplicative group. If α is a primitive element, then a primitive polynomial is minimal polynomial of α.

Notice 1. Primitive polynomial must have a non-zero constant term.

Theorem 3. Take an irreducible monic polynomial P(x) with degree m over F_p. If the lowest degree n such that P(x) divides x^n-1 is $n=p^m-1$, then P(x) is a primitive polynomial for F_q where $q=p^m$.

Theorem 4. Over GF(p^m) there exist exactly $\varphi(p^m - 1)/m$ primitive polynomials of degree m.

Unfortunately, there does not exist an efficient algorithm of finding a primitive polynomial. Instead, each irreducible polynomial needs to be checked. For that reason, a table of primitive polynomials exist which can be referenced as needed.

Example 2. Let us now show that $p(x) = x^2 + x + 1$ is a primitive polynomial for F_4. Let α be the root of $p(x)$, i.e. $p(\alpha) = 0$. That means every time we see $\alpha^2 + \alpha + 1$ we can replace it with zero, or we can use $\alpha^2 = \alpha + 1$ (which follows from $p(\alpha) = 0$) to lower the degree when multiplying. Table 2.1 shows how to generate F_4 as multiplicative group of α. Last 2 rows are redundant. They are there to show how to work with higher powers of α.

Table 2.1 Relationship between powers of α and polynomials

Powers of α	As polynomial
0	0
1	1
α	α
α^2	$\alpha + 1$
α^3	$\alpha\alpha^2 = \alpha(\alpha + 1) = \alpha^2 + \alpha = \alpha + 1 + \alpha = 2\alpha + 1 = 1$
α^4	$(\alpha + 1)(\alpha + 1) = \alpha^2 + \alpha + \alpha + 1 = \alpha$

Now we can see how we can easily transfer between powers of α and polynomial representation of F_q where $q=p^n$. Thus, if we need to multiply, representation as powers of α will be more useful, while for addition, polynomial representation is easier to work with.

Another representation that is useful when working with $GF(2^n)$ is representation as a power of 2. In other words, take a polynomial of degree m and convert it to binary by lining up all its coefficients starting from the one which multiplies unknown with the power m. Now we can take that binary number and represent it in a decimal form. For example, polynomial x^4+x^2+x+1 over GF(2)[x] can be represented as a binary number $10111_2=16+4+2+1=23$. Now if we take prime element to be represented as 2, i.e. α^1, and represent the prime polynomial in binary form, we can then take any element of the extension field and represent it as a power of 2. Log-antilog table of GF(256) can be found in the Appendix A.

2.2. Reed-Solomon codes

Reed-Solomon (RS) codes for error detection and error correction are widely used in CDs, DVDs and QR codes. Since this method encodes symbols (sets of bits) rather then single bits, it is efficient for detecting and correcting burst errors.

RS codes use Galois fields to represent codewords. Specifically, we will use 8 bit (one byte) symbols so GF(256) = GF(2^8) with primitive polynomial F(x) will be the field we will do calculations on.

RS codes are a subset of much general Bose–Chaudhuri–Hocquenghem (BCH) codes. All BCH codes use generator polynomial to generate error correction codewords. The idea is that if the message is transferred correctly (without errors in the communication channel) or the error detection capabilities are excited, then the received message will be divisible by the generator polynomial. If the received message has a reminder when dividing by the generator polynomial, we know some errors occurred.

From the previous paragraph, it can be concluded that messages are represented as polynomials whose coefficient are from GF(256), i.e. they are message symbols.

There exist 2 approaches for generating BCH codes. First one is called the non-systematic approach. To achieve a divisibility by generator polynomial g(x), codeword c(x) is calculated by multiplying the message polynomial m(x) by g(x). In other words,

$$c(x) = m(x) \cdot g(x).$$

While this approach seems simple to implement, it does not yield desirable results as we cannot easily read the original message m(x).

Second approach (systematic approach) is to append the error correction symbols to the end of the message, thus not obstructing readability of the initial message. Remember that the codeword must be a multiple of g(x). Thus, this approach divides the shifted message m'(x) by the g(x) and adds the reminder onto the m'(x). Newly generated c(x) is then divisible by g(x). In other words,

$$c(x) = x^{n-k}m(x) + x^{n-k}m(x) \; mod \; g(x).$$

where n is block length of the code, k is message length, n-k is parity check symbol length. Shifting operation makes sure that there is enough space at the end of the message for the error correction symbols.

Another important observation is the error correcting capability of the code, i.e. how many symbols can be recovered in case of error. That value is denoted as t and is given by this relation:

$$2t = n - k$$

Generator polynomial for the RS codes is then defined as:

$$g(x) = \prod_{i=1}^{2t}(x - \alpha^i)$$

where α is the primitive element of used Galois field, i.e. the root of F(x).

2.2.1. Encoding example

Let's now take the message from the chapter 1.1.2 and try to encode it using Reed-Solomon code. Our message looks as follows: 0100 00000111 01010001 01010010 00101101 01000011 01101111 01100100 01100101 0000.

First step is to divide it into groups of 8 bytes and use log-antilog table for GF(256) in order to switch between integer form (i.e. polynomial form) and representation as power of α. After converting the message to decimal numbers, we get: 64, 117, 21, 34, 212, 54, 246, 70, 80. Now use the log-antilog table to get the powers of alpha representation: α^6, α^{21}, α^{141}, α^{101}, α^{41}, α^{249}, α^{173}, α^{48}, α^{54}. Therefore, our message polynomial is as follows:

$m(x) = \alpha^6x^8 + \alpha^{21}x^7 + \alpha^{141}x^6 + \alpha^{101}x^5 + \alpha^{41}x^4 + \alpha^{249}x^3 + \alpha^{173}x^2 + \alpha^{48}x + \alpha^{54}$

Reading from the QR code standard, we have QR code version 1-H with 26 codewords, 9 data codewords and 17 error correction codewords. Thus, our generator polynomial is as follows:

$g(x) = (x - \alpha^0)\cdots(x - \alpha^{16}) = x^{17} + \alpha^{43}x^{16} + \alpha^{139}x^{15} + \alpha^{206}x^{14} + \alpha^{78}x^{13} + \alpha^{43}x^{12} + \alpha^{239}x^{11} + \alpha^{123}x^{10} + \alpha^{206}x^9 + \alpha^{214}x^8 + \alpha^{147}x^7 + \alpha^{24}x^6 + \alpha^{99}x^5 + \alpha^{150}x^4 + \alpha^{39}x^3 + \alpha^{243}x^2 + \alpha^{163}x + \alpha^{136}$

Now, calculate codeword $c(x) = x^{17}m(x) + x^{17}m(x) \bmod g(x)$. After performing polynomial long division, we now finally have our codeword.

$c(x) = x^{17}m(x) + \alpha^{129}x^{16} + \alpha^{70}x^{15} + \alpha^{164}x^{14} + \alpha^{135}x^{13} + 0x^{12} + \alpha^{171}x^{11} + \alpha^{47}x^{10} + \alpha^{74}x^9 + \alpha^{48}x^8 + \alpha^{28}x^7 + \alpha^{65}x^6 + \alpha^{85}x^5 + \alpha^{123}x^4 + \alpha^{188}x^3 + \alpha^{200}x^2 + \alpha^{58}x + \alpha^{200}$

In other words, converting codeword to decimal, we get: 64, 117, 21, 34, 212, 54, 246, 70, 80, 23, 94, 198, 169 ,0, 179, 35, 137, 70, 24, 190, 214, 197, 165, 28, 105, 28.

2.2.2. Decoding steps

Decoding the systematic code is a relatively simple process. If no error occurred, we could read first k coefficients from the received polynomial R(x) (as R(x) in this case will be the same as c(x)). However, if R(x) is not divisible by g(x), then we know an error occurred and first we must carry out the error correction algorithm. In other words, we must find the closest word to the received polynomial R(x) which is a valid codeword in our RS coding system.

Error correction process is divided in five steps each of which is explained in the next subchapters.

2.2.2.1 Syndrome calculation

From the received word, calculate the syndrome. In other words, calculate 2t syndrome values as follows:

$$S_i = R(\alpha^i), for\ i = 1, \dots, 2t$$

Since powers of α (up to 2t) are roots of g(x), they are the roots of c(x) as well. Thus, all the syndromes will be equal to zero if no error occurred or error detection capability is exceeded, i.e. iff R(x) is a valid codeword.

2.2.2.2 Error-locator polynomial

Error-locator polynomial σ(x) whose inverse of roots represent the error-locator numbers z_i. Degree of the polynomial σ(x) is denoted as T and represents number of symbol errors that have occurred.

Most efficient algorithm known today for finding the error-locator polynomial is the Berlekamp's iterative algorithm for RS codes. It is presented by the Table 2.2.

Table 2.2 Berlekamp's algorithm

i	$\sigma^{(i)}(x)$	d_i	h_i	$i-h_i$
-1	1	1	0	-1
0	1	S_1 (Syndrome)	0	0
1	…	…	…	…
…	…	…	…	…
2t	σ(x)	-	-	-

Rows where i=-1 and i=0 are setup of the algorithm. Row i=2t yields error-locator polynomial σ(x). In other words, $\sigma^{(2t)}(x) = \sigma(x)$.

Rules for populating the table are as follows:

1. If $d_i = 0$ then $\sigma^{(i+1)}(x) = \sigma^{(i)}(x)$ and $h_{i+1} = h_i$
2. If $d_i \neq 0$ then find a j-th row such that $j < i$ with the largest value in the column denoted as $i-h_i$.
   $\sigma^{(i+1)}(x) = \sigma^{(i)}(x) + d_i d_j^{-1} x^{(i-j)} \sigma^{(j)}(x)$
   $h_{i+1} = \max(h_i, h_j+i-j)$
3. $d_{i+1} = S_{i+2} + \sigma_1^{(i+1)} S_{i+1} + \cdots + \sigma_{h_{(i+1)}}^{(i+1)} S_{i+2-h_{(i+1)}}$,
   where $\sigma_j^{(i+1)}$ are the coefficients of
   $\sigma^{(i+1)}(x) = 1 + \sigma_1^{(i+1)} x + \sigma_2^{(i+1)} x^2 + \cdots + \sigma_{h_{(i+1)}}^{(i+1)} x^{h_{(i+1)}}$

We can now calculate the reciprocal $\sigma_r(x)$ whose roots, as was said before, are error-locator numbers z_i. $\sigma_r(x) = x^T \sigma(x^{-1})$

2.2.2.3 Error locations

We can now find the error locations. First, we need error location numbers z_i, i.e. the roots of $\sigma_r(x)$. This can easily be obtained by a brute force method known as Chien search. In other words, we need to try every single power of α from 0 up to and including n-1. Then, we can find the error locations as $x_i = X^{\log_\alpha z_i}$.

2.2.2.4 Error values

Syndromes S, error-locator numbers z, and error values y are linked via this equation:

$$S_i = \sum_{j=1}^{T} y_j z_j^i$$

So, we can obtain error values by solving systems of linear equations. It is possible that we will be presented with more equations in this system than the number of unknowns. In that case, discard the excess ones.

2.2.2.5 Estimate of error

Now, we can construct the estimate of error as:

$$E'(x) = \sum_{i=1}^{T} y_i x_i$$

And we can obtain estimate of the transmitted codeword C'(x).

$$C'(x) = R(x) + E'(x)$$

3. Developing a prototype system

3.1. Scanning the QR code

Images we get from the camera sensor need to be transformed into black and white images. This process is called image binarization. This can be done in one of three ways:

1. Transform pixel into black pixel if its grayscale value is less than 0.5, transform into white otherwise.

2. Calculate the average of grayscale values for all pixels and transform pixel into black pixel if its grayscale value is less than the average. Transform to the white otherwise.

3. Divide image into n by m rectangular regions and perform option number 2 in each region.

As luminescence of an image can be vastly different in different regions, option number 3 proved to be the best option, while options 1 and 2 loose more data than desired. In other words, if the QR code is in a shade and other parts of an image are overly exposed to the sunlight, with options 1 and 2 we would loose entire portion of a space which is in the shade including the QR code.

While determining the position of the QR code within a picture, finder pattern is all we need. By scanning the image horizontally, vertically, and diagonally while looking for the correct ration of black and white pixels, we can precisely determine the position of the QR code.

3.2. Decoding the QR code

QR Code decoder implementation can be seen on the Figure 3.1 class diagram.

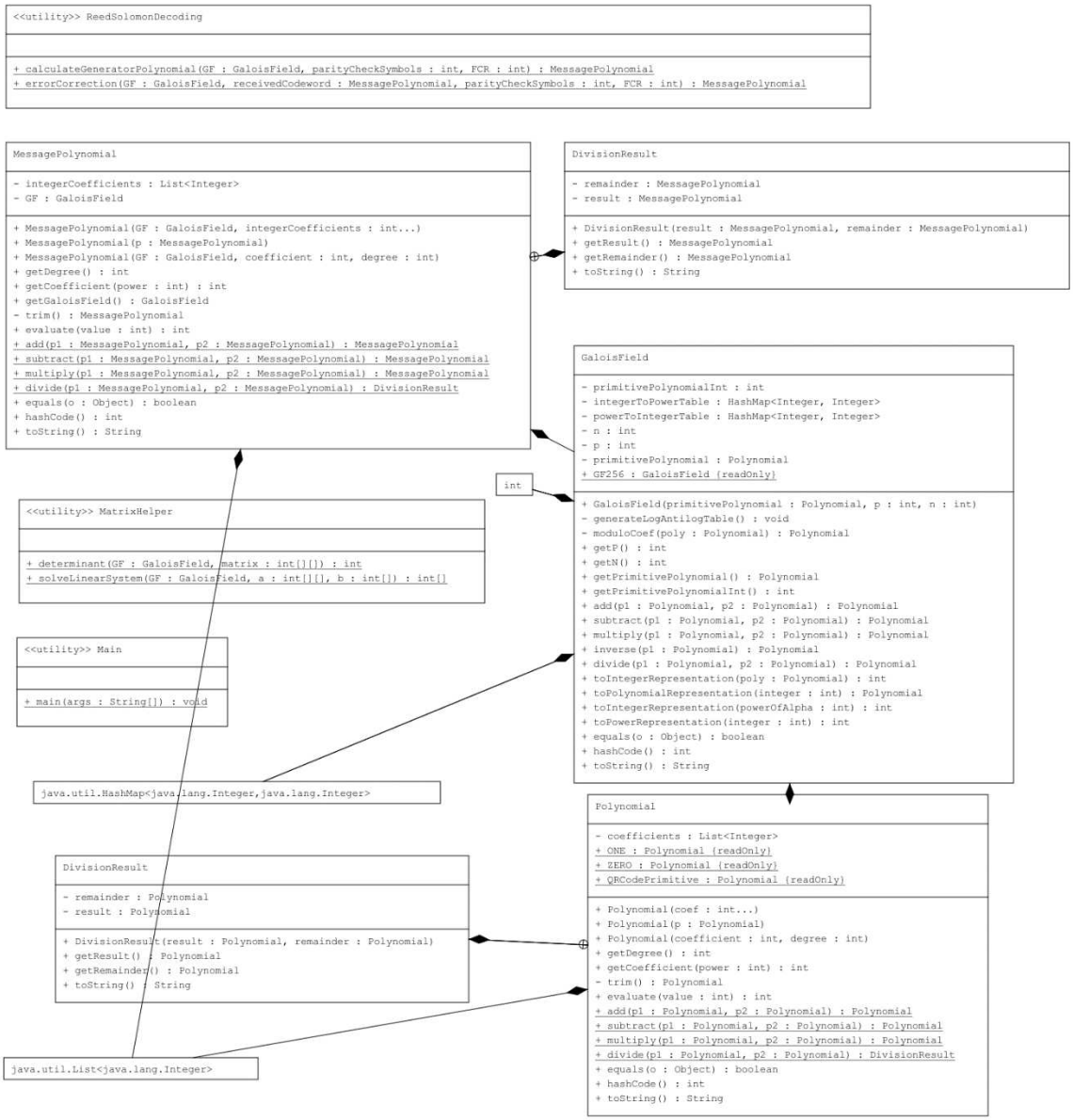

Figure 3.1 Class diagram

Conclusion

QR code is a complex and resilient type of a 2D barcode. It is a powerful tool for transferring data from physical to digital media. Their complexity, which is highly due to a mathematics knowledge requirement, and error resilience is sometimes not needed. Because of that, linear barcodes still play a major role in transferring information.

QR codes are transmitted through noisy channels – image sensors. For that reason, QR codes provide a method of error correction – Reed-Solomon coding. Although it is not perfect, it can correct burst errors what is most important in this case.

Success of decoding a QR code highly depends on the scanning process. Error correction is there only because of the scanning process imperfections. As one image can have vastly different luminance in different regions of that image, scanning algorithm must consider these regions before binarization. As we have seen in this thesis, viewing the image as just one region leads to loss of important information in case the QR code is in a less illuminated part of the image.

Bibliography

[1] Information technology — Automatic identification and data capture techniques — Bar code symbology — QR Code; ISO/IEC 18004; First edition; 16.5.2000.

[2] https://en.wikiversity.org/wiki/Reed-Solomon_codes_for_coders; May 2022.

[3] https://en.wikipedia.org/wiki/QR_code; May 2022.

[4] https://www.thonky.com/qr-code-tutorial/; 27.10.2021.; May 2022.

[5] https://github.com/zxing/zxing; May 2022.

[6] Tutorial on Reed-Solomon error correction coding; Geisel, William A.; August 1, 1990; NASA Technical Memorandum 102162;
https://ntrs.nasa.gov/citations/19900019023

[7] OFDM Reed Solomon Coding; Silicon DSP Corporation;
https://www.youtube.com/watch?v=K26Ssr8H3ec

Appendix A. Log-antilog table for GF(256)

Using primitive polynomial $x^8+x^4+x^3+x^2+1$ with primitive element $\alpha=2$.

Power of α	Integer representation
- ∞ (by convention)	0
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	29
9	58
10	116
11	232
12	205
13	135
14	19
15	38
16	76
17	152
18	45
19	90
20	180
21	117
22	234
23	201
24	143
25	3
26	6

27	12
28	24
29	48
30	96
31	192
32	157
33	39
34	78
35	156
36	37
37	74
38	148
39	53
40	106
41	212
42	181
43	119
44	238
45	193
46	159
47	35
48	70
49	140
50	5
51	10
52	20
53	40
54	80
55	160

56	93
57	186
58	105
59	210
60	185
61	111
62	222
63	161
64	95
65	190
66	97
67	194
68	153
69	47
70	94
71	188
72	101
73	202
74	137
75	15
76	30
77	60
78	120
79	240
80	253
81	231
82	211
83	187
84	107
85	214
86	177
87	127
88	254

89	225
90	223
91	163
92	91
93	182
94	113
95	226
96	217
97	175
98	67
99	134
100	17
101	34
102	68
103	136
104	13
105	26
106	52
107	104
108	208
109	189
110	103
111	206
112	129
113	31
114	62
115	124
116	248
117	237
118	199
119	147
120	59
121	118

122	236
123	197
124	151
125	51
126	102
127	204
128	133
129	23
130	46
131	92
132	184
133	109
134	218
135	169
136	79
137	158
138	33
139	66
140	132
141	21
142	42
143	84
144	168
145	77
146	154
147	41
148	82
149	164
150	85
151	170
152	73
153	146
154	57

155	114
156	228
157	213
158	183
159	115
160	230
161	209
162	191
163	99
164	198
165	145
166	63
167	126
168	252
169	229
170	215
171	179
172	123
173	246
174	241
175	255
176	227
177	219
178	171
179	75
180	150
181	49
182	98
183	196
184	149
185	55
186	110
187	220

188	165
189	87
190	174
191	65
192	130
193	25
194	50
195	100
196	200
197	141
198	7
199	14
200	28
201	56
202	112
203	224
204	221
205	167
206	83
207	166
208	81
209	162
210	89
211	178
212	121
213	242
214	249
215	239
216	195
217	155
218	43
219	86
220	172

221	69
222	138
223	9
224	18
225	36
226	72
227	144
228	61
229	122
230	244
231	245
232	247
233	243
234	251
235	235
236	203
237	139
238	11
239	22
240	44
241	88
242	176
243	125
244	250
245	233
246	207
247	131
248	27
249	54
250	108
251	216
252	173
253	71

254	142	255	1

Sažetak

QR kodovi vrsta su 2D barkodova. Dvije su faze očitavanja ovih barkodova – skeniranje i dekodiranje. Prva faza, faza skeniranja, otkriva poziciju QR koda na fotografiji tražeći unaprijed definirane uzorke pretrage. Nakon otkrivene pozicije, QR kod pretvara se u niz bitova koje čine informaciju. Kako se barkodovi prenose kroz nesavršene komunikacijske kanale, dolazi do grešaka u prijenosu informacija. Prema ISO standardu, QR kodovi zato upotrebaljavaju Reed-Solomon kodove kao metodu otkrivanja i ispravljanja grešaka. Reed-Solomon kodovi poruke prikazuju kao polinome čiji koeficijenti odgovaraju podacima koje prenosimo. Također, oslanjaju se na Galoisova polja (konačna polja) kako bi nad tim koeficijentima izvodili operacije zbrajanja, oduzimanja, množenja, dijeljenja i potenciranja.

Summary

QR codes are type of 2D barcodes. There are two phases of reading these barcodes - scanning and decoding. The first phase, the scanning phase, reveals the position of the QR code in an image by scanning for predefined finder patterns. Once the position is detected, the QR code is converted into a stream of bits that represent the information. As barcodes are transmitted through noisy communication channels, errors can occur in the transmission of information. By the ISO standard, QR codes therefore use Reed-Solomon codes as a method of detecting and correcting errors. Reed-Solomon codes represent message as polynomials whose coefficients correspond to the data we transmit. They rely on Galois fields (finite fields) to perform addition, subtraction, multiplication, division, and exponentiation operations on these coefficients.