Dedicated MCU-based QR Code Scanner Using Image Processing

By

Mark Angelo B. Domingo
Patrick Jovit A. Jove
Dario Lemuel D. Ty

A Thesis Submitted to the School of Electrical, Electronics and Computer Engineering
In Partial Fulfilment of the Requirements for the Degree

Bachelor of Science in Computer Engineering

Mapua Institute of Technology
June 2014

APPROVAL SHEET

This is to certify that I have supervised the preparation of and read the thesis report prepared by **Mark Angelo Domingo, Patrick Jovit Jove, and Dario Lemuel Ty** entitled **"Dedicated MCU-based QR Code Scanner using Image Processing"** and that the said report has been submitted for final examination by the Oral Examination Committee.

JESUS M. MARTINEZ JR.
Thesis Adviser

As members of the Oral Examination Committee, we certify that we have examined this thesis report and hereby recommended that it be accepted in fulfillment of the thesis requirements for the degree in **Bachelor of Science in Computer Engineering.**

JOSHUA B. CUESTA
Panel Member

DIONIS PADILLA
Panel Member

JOSE LAZARO
Panel Member

This thesis paper is hereby approved and accepted by the School of Electrical Engineering, Electronics Engineering, and Computer Engineering in partial fulfillment of the requirements for the degree in **Bachelor of Science in Computer Engineering.**

FELICITO S. CALUYO
Dean, School of EECE

ACKNOWLEDGEMENT

Special people who have given their efforts, assistance and guidance to this thesis paper were behind the efficacious completion of this hard-earned work. This project would not have been accomplished if not because of them.

First of all, the researchers would like to express their heartfelt gratitude to our Lord Almighty for all the blessings bestowed upon them, giving them the talents, perseverance, and knowledge to entirely do the paper and prototype. Every paragraph written, every module tested successfully, every application implemented, and every conclusion formulated, are all lifted onto our Lord's glory.

The researchers are also grateful to Engr. Jesus Martinez Jr., the thesis adviser, for shining a light of initial interest in the presented topic and for the unceasing guidance throughout the course of documentations and prototype. They highlighted the need, purpose, and significance of the course, giving out clear, direct, and well-organized thoughts and contributions in addition to the researchers' work.

To each member of the group, each one is thankful for the dedication, contribution, perseverance, wit, and positivity that has been offered. Thankful for the healthy brain storming, decision making, and conflict resolving that seldom arise.

Finally, having been thankful of our families for the love, understanding, and undying support. They have provided the group with financial aid, and the emotional support that drives the group towards their goal.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The study entitled “Dedicated MCU-based QR Code Scanner Using Image Processing” aims to implement hardware dedicated QR code scanner to translate version 1 QR code to its text equivalent. The device involves a camera interfaced to a microcontroller with the output seen on a LCD display. Testing of the device involves selection of proper filters for the camera, and four possible attack scenarios for the position of the QR code and successfully recognizing the value embedded in the code using the standard decoding algorithm of QR codes.

Keywords: QR code, Microcontroller, Camera

Chapter 1

INTRODUCTION

Barcodes are symbols that can be read by optical machines. The barcode contains the data that is readable in those machines. Originally, the data inside the barcode are given with varying widths and spaces. These barcodes are usually referred to as a one-dimensional kind. Then, we use various shapes such as dots and rectangles in two-dimensional barcodes for the symbol. Barcode readers are used to scan barcodes and are usually seen in commercial establishments, it also presently used in more devices such as smartphones and tablets. Since 2D Barcodes such as the QR code stores more information than 1D Barcodes, it is most commonly used in advertisements where the URL of a certain company is encoded into the QR code, or even a QR equivalent conversion of a business card. QR codes may be used in a lot of possible applications such as Identification (replacement for the RFID tags), Data Storage, Advertisements, Business Cards, etc.

In early years barcodes were used. One dimensional barcodes are universally used as a product identification representation. This kind of barcodes are usually seen in books, magazines, appliances, medicines, postal main and food packaging. This allows merchants and suppliers to keep track of inventory both coming into stores or suppliers and being sold or transferred. It was present in almost all commercial merchandise. Decoders for the one-dimensional barcodes are dedicated. As one-dimensional barcodes are only scanned horizontally the capacity of the stored information is limited. The 2D barcode solves this kind of problem. In contrast with the one-dimensional barcode, 2D barcode such as QR Codes can

be scanned horizontally and vertically and as a result, this kind of barcode has larger capacity than 1D barcode. It can process the image even when not properly aligned. Given its obvious advantage, it is a better choice to use this technology. This kind of technology are usually decoded by using mobile devices such as smartphones and tablets. Translators and decoders are integrated in the gadgets. And because of that, QR codes are popularly used for mobile marketing.

There are limitations though, such as, the QR code scanners are only available on smartphones and web applications (such as online QR code scanners, readers, encoders and decoders). This limits the potential of the QR code and also the 2D barcode as a replacement for 1D barcode, a product identification symbol, business card identification, and a container of short messages. As of now, there are no available QR code scanners yet in the market with a dedicated hardware for decoding the QR codes.

The aim of this study is to implement a dedicated microprocessor-based QR code scanner. These are the following objectives of this study: 1) Capture the image using the camera of the device and apply image processing to make the image practical for scanning (2) Measure the total time it takes to capture, process, decode, and display the result of the scanned QR code with respect to varying character length and data type encoding (3) Measure the success rate of decoding error free QR codes (4) Measure the success rate of decoding QR codes introduced with errors in different attack scenarios

This study is significant because this can extend the functionality of QR codes. Typically QR codes open links to websites, read business cards, and promote products through advertising. But with further research, we can make the QR scanner become an alternative to various code scanners like bar code scanners. It also examines how QR codes are decoded and how can it be improved its functionality and performance. The main purpose of this is to know the benefits of using a QR code decoder with a dedicated hardware.

The study focuses on the hardware implementation of QR code decoding. It is a dedicated device that can decode QR codes attached into different objects. These are the limitations that will be applied during the study: (1) The device testing will be limited only to QR code version 1 (2) The device will only correct error if it is less than the maximum allowable error (3) The device will not be tested on QR codes that uses Kanji or similar mode of character encoding (4) The device will output the result in its own display.

Chapter 2

REVIEW OF RELATED LITERATURE

QR Code Encoding

Generating a binary string

QR codes are encoded by the following steps: the first is that the data, the information about the encryption method, and the length of the encrypted data are included in the generated binary string. The four-bit data indicates the mode of data that is configured to be used. Encoding modes can be Numeric, Alphanumeric, Binary, or Japanese. Each mode has its own bit representation. The length of the data should be converted into binary. We need to add zeros (0) on the left of the length (converted into bits), if the number of bits of the length is less than the required bits. (ISO/IEC, 2006)

Different versions require different number of bits. For example, for versions 1 through 9, the length should be 9 bits if the encoding mode is alphanumeric. Next is to encode the data. Data encryption is divided into pairs, if the mode is alphanumeric, then for each pair, we get the ASCII value of the very first character and multiply by 45. This number is being added to the ASCII value of the other characters. After that, change the outcome into a binary string of 11 bits. There are different approach on how we can generate the binary string of the data being encoded, it depends on the encoding mode being used. Add up to 4 zeros to the end, to indicate that it is the end of the bit string. The generated bit string is then grouped into 8 bits (add 0's to the right if necessary). The number of 0s being added depends on the capacity

of the version in use. Different versions have different capacity of data bits. There are two special bit strings being used if the generated bit string is not long enough, these are 11101100 and 00010001. This should be placed at the end of the generated bit string in alternating order, until the size of the data bits required is met. (ISO/IEC, 2006)

Generating error correction

A QR code area for resolving errors are included. The data block of QR code can be read safely with this error correction codes even if the some parts of the QR code cannot be recognized. Reed-Solomon error correction is used in QR codes. The number of error correction code words that are being generated depends in the format version, mask pattern, error correction level and data inside which are specified in the QR code. By using Reed-Solomon Error Correction algorithm, the QR code information is first converted into polynomials, the number of points that explain the need for individual polynomials are defined, the point set is inserted back in the QR code and the data is then expressed as a polynomial. In other words, the error correction algorithm adds the QR code recovery information. There are four levels of error correction that is used in the QR code to the data and helps it depending on how much damage is expected to suffer in the QR code that is placed in a certain environment. With this, error corrections are needed. (ISO/IEC, 2006)

The levels of error correction are described in the figure below:

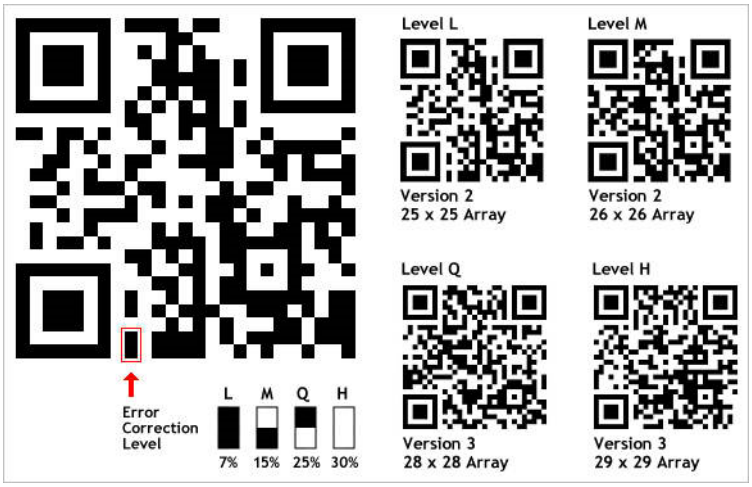


Figure 2.1: QR Code Error Correction Levels

There are two modules that are present in the bottom of the QR code beside the pattern recognition area. This determines the error correction level of the QR code. The error correction level solution affects the decoding process used in a QR code reader. The level of mistakes is one of the reasons why the QR code with the same data from different production QR codes use to fix the errors that are used in a particular website. There are variables in ISO/IEC 18004 which the error correction level being one of them that will lead to different views of a QR code image based on how a QR code creation website especially set this variables. After creating the error correction code words, it is placed after the data code words produced in the previous step. (ISO/IEC, 2006)

Choosing the mask pattern and finishing the QR code

QR code uses eight different mask patterns. The slight change in the mask pattern is according to a specific rule. These mask patterns are defined in the standard QR code. QR codes are different, but even if someone is using a mask pattern, all QR code reader should work well in this area, because it cannot fail, meaning that it must always be present (ISO/IEC, 2006). For QR code version 1, these patterns must always be present: position detection pattern, timing pattern and the black pixels as shown in the figure below:

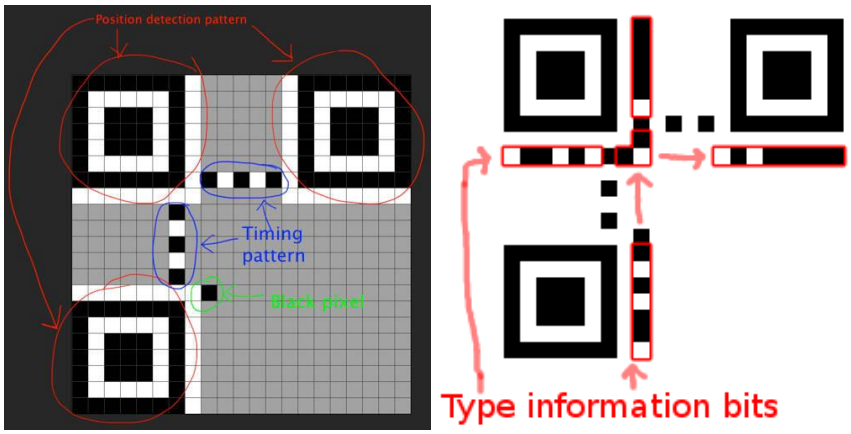


Figure 2.2: QR Code Structure

The information bit contains information about the level of error correction being used as well as the mask pattern. It is important for the base model that the decoder has to say on how the data is encoded in pixel or module is mandatory as far as possible. That information should not be eliminated. (ISO/IEC, 2006)

Image Processing

Acharya and Ray stated in 2006 that Image processing is a set of 2D image that is processed by a computer. An image processing in the production of an image or a set of characteristics or parameters related to image data such as a photograph or video frame of the image, is a method of signal processing. The image processing strategy involves image treatment as a 2D signal and the application of standard signal processing techniques. (Acharya and Ray, 2006). In MATLAB, an image can be processed by using the command imread('filename'). The image is then converted into 3 matrices that represents the intensity of its pixels in RGB respectively (Sandberg, 2011). In RGB separation, the matrix other than the one being filtered is zeroed. When R or Red is being filtered, Green and Blue are zeroed, when Green is being filtered, Red and Blue are zeroed, and when Blue is being filtered, Green and Red are zeroed.

QR Code Image Detection Using Run-Length Coding

QR stores more information than traditional barcodes. It can be created as a repository for Web addresses, contact information and product identification. QR code has only limited application because there is noise in the data collection, storage, identification and tracking. A-Lin et al. improved the performance of the bar code in practical applications. Improved the pre-treatment of the bar code, which consists of various ways for image modification, they tried to improve the performance of the QR code readability and detection. They captured QR code images using an industrial camera which only captures images in black and white. QR

code histogram equalization is experienced by the image to extend the level of the histogram. The histogram offers the QR code wide range of shades of gray. Optimal fitting of the image is also used to calculate the intensity and divide the gray levels of the image. Finally, the image is implemented with binarization, scanned each pixels and each is labeled with monochrome colors, depending on the results shown. They also proposed a novel way to find the QR code to the nature of the kind of QR code. Made from the data stored in a data value, the run-length encoding is a simple method of data compression (A-Lin et al., 2011). In this way, it is effective in the QR code. The characteristics, methods of implementation, use of the QR code to convert the length of a character from the source signal in the field of accounting. Then, the QR code is converted to runs with alternating pixels. Activities running between neighboring rows of module unit, are related to the end. Then they scan the whole image, a QR code has two modules. Since there is a possibility that errors occur in this process, error correction is important. Reed-Solomon correction algorithm is used. RS code for the correct data inside the QR code is generated. Even if the bar code is damage, it is the right algorithm of the bar code will correct the data on a regular basis. This detection method of improves the quality of the bar code. The error correction method is also effective. (A-Lin et al, 2011)

Identification of QR Codes Based On Pattern Recognition

In practical applications, Sun stated in 2011 that QR codes are widely adopted on surface of different materials other than paper. The texture features are different with each material. QR code modules are different from each other: there are dots, seriate squares and discrete squares. Generating QR code are also different: presswork and labels are used in

papers, laser printing in plastics, dot printing, laser carving and electro-chemical etching on metal surfaces. Sun et. al. provided a method that is based on pattern recognition to decode QR codes of different modules in various materials. The algorithm includes two sections which are training and real-time identifications. Pattern recognition is used to assign a label to a given object. This includes the reorganization, classification and interpretation of things. In related to the study, things are divided into two classes which are the object and the QR code. Pre-processing the QR code is essential in the identification system as this influences the accuracy of identification. This is aimed to remove the factors which brings a negative influence and also to improve the identification rate of the image (Sun et al., 2011). This includes grey processing and filtering. The texture features are invariant to grey-scale and rotation and because of that, they had adopted multi-resolution histogram and local binary patterns. During training, they split and marked the QR codes in sample images and split the images into different block. In each sub block, the method calculated the image texture features such as Multi Resolution Histogram and Local Binary Pattern which were invariant to the grey-scale and rotation. Texture features were trained through Spatial Boost algorithm of Shia Avidan to get a QR code identifier. In real-time identification, the identifier splits the input image and use the texture characters as the input vector to classify the sub-image in background or 2D barcode. After that, they combined the sub-block which were the 2D barcode into QR code. After the last training, they had reached 100% rate of detection in the test database of their experiment. The algorithm used in identifying the QR code in various things using pattern recognition was effective. (Sun et al., 2011)

QR Code Detection Speed

For QR code decoders, it is necessary that the symbol has been entered correctly. For the QR code is suitable for decoding, there must be at least 30% of the corresponding image frame (Belussi et al). Images captured depend on the blur, rotation, noise, uneven lighting, perspective distortion or partial occlusion area code. Decoders may not identify these images as negative factors such as noise and distortion. Belussi and Hirata's goal is to establish a methodology for detecting bar code for arbitrarily obtained images. Using Viola-Jones framework to increase recognition of the boosted cascade of classifiers, which can be used for rapid detection of objects in the image. This gives a better and more effective way to concentrate on the detection process in important areas of the image of the QR code, and also a very fast calculation in accordance with the pattern classification approach. During the image formation, detection accuracy is 90 %. There are a number of false positive that were controlled. Search pattern have sizes of 8 x 8 and larger, including those that have been played with a certain perspective distortion can be detected. A post-processing algorithm has been proposed to aggregate the detected part of the QR code. The processing time is consistent with real-time applications. (Belussi et al.)

Malicious Pixels Using QR Codes as Attack Vector

Keiseberg stated in 2012 that malicious pixels can be inserted at the QR and can be used in different ways. These modules can be changed either from white to black or black to white. There are several areas where the QR code module when manually changed with a pen

or marker, can changed the original coded data. The masks, character encoding mode, indicator of character count and the data and error correction area are the areas of attack. Among the four possible area, the data and error correction is the largest part of the code. By changing some of the modules in this area the data code is changed. (Keiseberg et al., 2012)

Arduino Uno and the ArduCAM

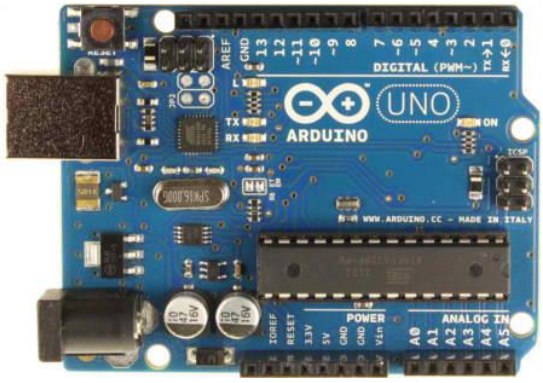


Figure 2.3: Arduino UNO Microcontroller

Arduino Uno is based on ATmega328 microcontroller, it features 14-pin digital I/O, 6-pin input analog, USB port, power socket, the ISCP header, ceramic resonator and a reset button. Aside from being general input/output ports, digital pins can also be used to emulate serial communication using the software serial library included in the arduino IDE. The software serial library emulates digital pins to act as serial ports Tx and Rx, transmitter and receiver, respectively. Tx and Rx can be assigned within the program code by declaring a variable with a type of SoftwareSerial, and passing 2 values to the variable name which corresponds to the pins of the arduino. The first value is the pin that is treated as the Tx or the transmitter, while the second value the pin that is treated as the Rx or receiver of the serial

data. The USB port is used for burning the program developed in the Arduino IDE, and to supply power to the arduino. There are several shields that can be attached directly to the arduino board. Shields are boards that can be plugged on top of the Arduino PCB extending its capabilities. (Arduino 2013)

Arducam is a camera shield for Arduino boards. Most of the I/O ports on Arduino are used by Arducam for its own use. The I/O ports on arduino however are extended on the GPIO ports (General Purpose Input/Output ports) on Arducam's. The following figure shows the arducam's pinouts which is also the extension of arduinos I/O Ports:

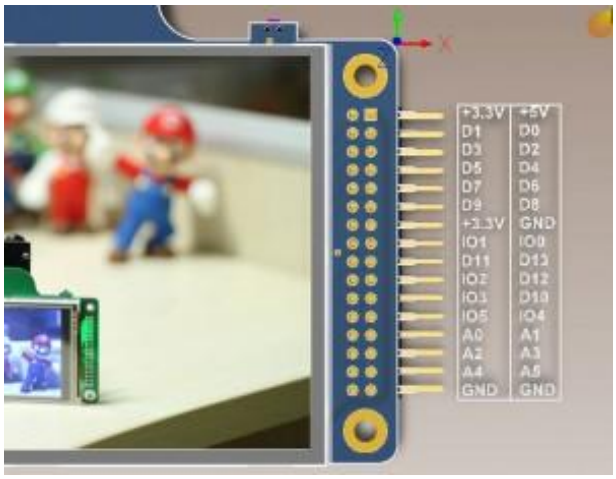


Figure 2.4 Arducam Pinouts

Pins 1 and 2 are used by the camera, and cannot be used as input or output pin, neither be used as tx/rx pins for software serial emulation. ArduCAM can take Bitmap and JPEG image files and can be saved to a SD or TF cards. Images are captured in RAW format, which contains binary data that is stored on the LCD's image buffer. ArduCAM has also a 3.2'' LCD screen that provides real-time previews of the camera. The LCD's screen resolution is 320 pixels by 240 pixels, and since images taken by the arducam comes directly from the LCD's image buffer, the size of images in pixels are limited to the LCD's screen resolution. As of now, the

Arducam cannot do real time video recording and real time video processing. (ArduCAM, 2013)

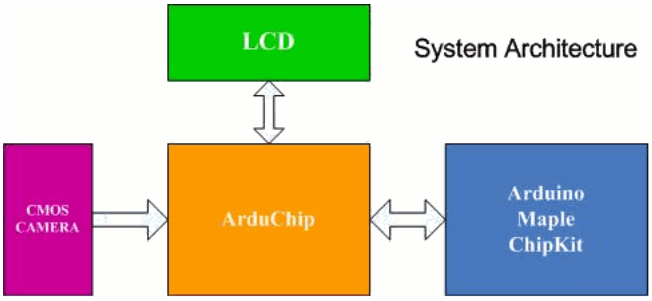


Figure 2.5: ArduCAM System Architecture

Figure 2.6: ArduCAM shield Rev. B and Camera Module

ArduCAM already have three revisions. The ArduCAM shield Rev. B has a LCD module, a 3.2 inch QVGA (320x240) TFT LCD. The MCU platform has two libraries: the UTFT for the LCD and ArduCAM library for connections via I2C and accessing the internal control registers. The ArduiChip do the DMA transfer between the camera module and the GDDRAM of the LCD, control the camera data stream timing and GPIO expansion. (ArduCAM, 2013)

According to Micron, the creators of the camera module, the MT9D11 Camera Module is a CMOS image sensor that is low-power, low-cost and progressive scanning capabilities. It has 2 Megapixel resolution and can have up to 15 frames per second at full resolution. The camera module is supported by ArduCAM and can be used in certain applications that involves

imaging. The camera module can be attached to the Digital Video Port of the ArduCAM. This port can support multiple camera modules from 0.3 Megapixels up to 5 Megapixels. (Micron Technologies, 2004)

When the ArduCAM is powered on, it will start in preview mode. The chip will access the camera data to be passed to the LCD screen with frame rate up to 30 fps. When the capture button is pressed, the screen will freeze and enters the capture mode, which the MCU will read the image data from the GDDRAM of the LCD and save it to the SD or TF card. (ArduCAM, 2013)

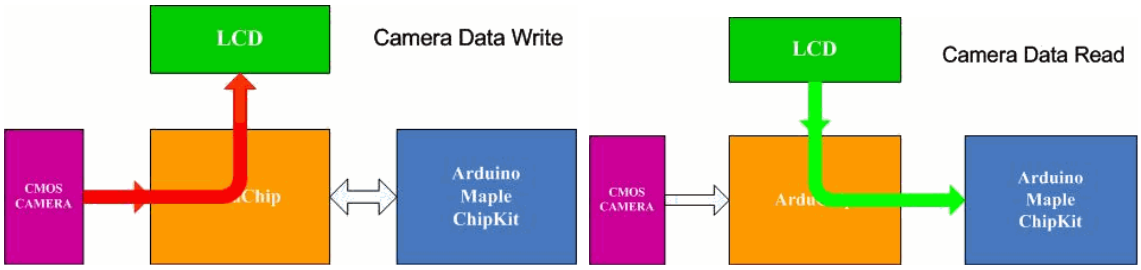


Figure 2.7: ArduCAM Data Read and Write

RGB555 and RGB565

RGB555 is a kind of 16 bit color format and pixels which are represented by two bytes, the hi-byte and the lo-byte. The 555 characters stands for the number of bits that is used in every color value. In this case, only 15 bits are needed, with 5 bits used for very color value. The last bit or most significant bit is unused or a “don’t care” value. The image’s pixel organization buffer starts from left to right and bottom up. (The Imaging Source, 2013)

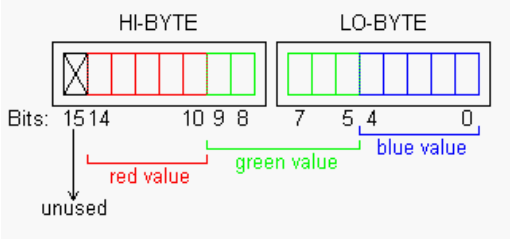


Figure 2.8: RGB555 Structure

The color format RGB565 is like RGB555 color format. Unlike the RGB555 format, the RGB565 has six bits that are for green values instead of RGB555's five bits. With this bit length, all of the RGB565's 16-bits are used. The pixel organization of this color format in the image buffer starts from left to right and bottom up. The human eye is sensitive to some colors, like the color green. RGB565 format is used to accommodate this fact. RGB565 can be converted to RGB555 easily by shifting the green value to the right or disregarding the LSB of the green bits. (The Imaging Source, 2013)

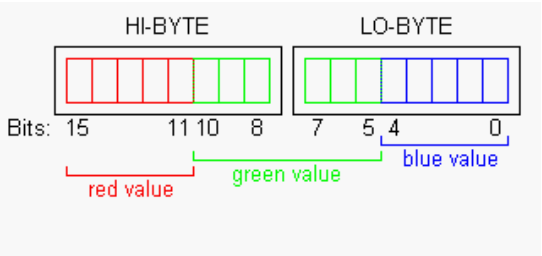


Figure 2.9: RGB565 Structure

Raspberry Pi

The raspberry pi is a powerful microcomputer that has 700 Mhz ARM processor and 512 MB of RAM. It can be interfaced to other devices through its GPIO (General Purpose Input/Output) pins. The GPIO pins are capable of sending and receiving digital signals to and

from other devices respectively. The following figure shows the GPIO pin mapping of the Raspberry Pi:

Figure 2.10: Raspberry Pi GPIO Pin Configuration

GPIO pins 14 (TxD) and 15 (RxD) can be used for serial communication with the device. The GPIO pins can be programmed using high level programming languages such as C and Python. The Raspbian Wheezy, a debian operating system port for the raspberry pi, supports a wide range of libraries available all over the internet, including barcode decoders such as Zbar. The configuration file on the raspberry pi is responsible for the boot process of the pi. This can be modified to enable a script to be run at startup.

Zbar

Zbar is a C-coded library used for real\ding barcodes. It is open source which can be used in C++, Perl, Python and Ruby (Brown, 2008). Its feature includes high-speed, real time

scanning, small memory usage, no floating point operations, and small code size. The library can decode different types of bar codes, including QR codes.

Zbar accommodates software development of different levels of interfaces depending on the program's and/or the programmers' needs. Low-level interfaces include the basic decoder and scanner, intermediate interfaces include the ImageScanner, Window abstraction and the Video abstraction. Lastly, High-level interfaces include widgets for front-end use and the processor interface that adds a scanning window to an application with no GUI. (Brown, 2008)

Chapter 3

Microcontroller-based QR Code Scanner and Decoder Using Image Processing

Abstract

The study entitled “Dedicated MCU-based QR Code Scanner Using Image Processing” aims to implement hardware dedicated QR code scanner to translate version 1 QR code to its text equivalent. The device involves a camera interfaced to a microcontroller with the output seen on a LCD display. Testing of the device involves selection of proper filters for the camera, and four possible attack scenarios for the position of the QR code and successfully recognizing the value embedded in the code using the standard decoding algorithm of QR codes.

Keywords: QR code, Microcontroller, Camera

Introduction

Barcodes are symbols that can be read by optical machines. The barcode contains the data that is readable in those machines. Originally, the data inside the barcode are given with varying widths and spaces. These barcodes are usually referred to as a one-dimensional kind. Then, we use various shapes such as dots and rectangles in two-dimensional barcodes for the symbol. Barcode readers are used to scan barcodes and are usually seen in commercial establishments, it also presently used in more devices such as smartphones and tablets. Since 2D Barcodes such as the QR code stores more information than 1D Barcodes, it is most commonly used in advertisements where the URL of a certain company is encoded into the QR code, or even a QR equivalent conversion of a business card. QR codes may be used in a lot of possible applications such as Identification (replacement for the RFID tags), Data Storage, Advertisements, Business Cards, etc.

In early years, one dimensional barcodes were used. One dimensional barcodes are universally used as a product identification representation. This kind of barcodes are usually seen in books, magazines, appliances, medicines, postal main and food packaging. This allows merchants and suppliers to keep track of inventory both coming into stores or suppliers and being sold or transferred. It was present in almost all commercial merchandise. Decoders for the one-dimensional barcodes are dedicated. As one-dimensional barcodes are only scanned horizontally the capacity of the stored information is limited. The 2D barcode solves this kind of problem. In contrast with the one-dimensional barcode, 2D barcode such as QR Codes can be scanned horizontally and vertically and as a result, this kind of barcode has larger capacity than 1D barcode. It can process the image even when not properly aligned. Given its obvious advantage, it is a better choice to use this technology. This kind of technology are usually decoded by using mobile devices such as smartphones and tablets. Translators and decoders are integrated in the gadgets. And because of that, QR codes are popularly used for mobile marketing.

There are limitations though, such as, the QR code scanners are only available on smartphones and web applications (such as online QR code scanners, readers, encoders and decoders). This limits the potential of the QR code and also the 2D barcode as a replacement for 1D barcode, a product identification symbol, business card identification, and a container of short messages. As of now, there are no available QR code scanners yet in the market with a dedicated hardware for decoding the QR codes.

The aim of this study is to implement a dedicated microprocessor-based QR code scanner. These are the following objectives of this study: (1) Capture the image using the camera of the device and apply image processing to make the image practical for scanning (2) Measure the total time it takes to capture, process, decode, and display the result of the scanned QR code with respect to varying character length and data type encoding (3) Measure the success rate of decoding error free QR codes (4) Measure the success rate of decoding QR codes introduced with errors in different attack scenarios

This study is significant because this can extend the functionality of QR codes. Typically QR codes open links to websites, read business cards, and promote products through advertising. But with further research, we can make the QR scanner become an alternative to various code scanners like bar code scanners, replacing devices like barcode scanner and smartphones for scanning bar codes such as QR codes. The hardware will have a built in decoding for QR codes just like barcodes scanner which decodes one-dimensional barcodes. It also examines how QR codes are decoded and how can it be improved its functionality and performance. The main purpose of this is to know the benefits of using a QR code decoder with a dedicated hardware.

The study focuses on the hardware implementation of QR code decoding. It is a dedicated device that can decode QR codes attached into different objects. These are the limitations that will be applied during the study: (1) The device testing will be limited only to QR code version 1 (2) The device will only correct error if it is less than the maximum

allowable error (3) The device will not be tested on QR codes that uses Kanji or similar mode of character encoding (4) The device will output the result in its own display.

Methodology

The MCU-based QR code scanner is used to decode different QR codes with various character encoding specifically: numeric, alphanumeric, and binary. Errors are introduced at different attack scenarios to test the systems capability of correcting it. The Original QR Codes are generated using the online QR code generator by Raco Industries. Their online QR code generator is very manageable that it allows us to choose the version of the QR code, level of error correction, and the mode of character encoding that we want to use. The QR code is captured using a camera interfaced into a microcontroller, specifically, the Arduino Uno interfaced with Arducam Shield Rev. B respectively. The image is fetched from the image buffer of the LCD attached to Arducam byte-per-byte using FIFO (First in, First Out) algorithm, and then processed by the Arduino to produce an image practical for QR code decoding. The processed image is then sent to the Raspberry Pi that runs on Linux via Serial Communication. The Raspberry Pi uses the Zbartools library to decode the image, then sends the result back to the Arduino using Serial Communication. The Arduino then displays the result on the LCD attached to the Arducam shield. The original encoded data is compared to the decoded message. The total time of the whole process, from capturing the image to displaying the result is measured using an ordinary stopwatch, while the decoding time is displayed on the Raspberry Pi's terminal. The QR codes used in this research are encoded using an online QR code generator. From the code text box, we entered the original message, then we set the level of error correction to L or low, in which 7% of the original message can

be restored, then we set the character encoding to alphanumeric, and then left the other fields unchanged, then clicked on the generate button. We repeated these process while giving variation to the character encoding of the QR code. When the codes are scanned using our system, it might result to a success, or a failure. The result is considered a success if the decoded message by the system matches the original encoded message. Therefore, the ratio of success is calculated using the formula:

$$SUCCESS\ RATE = \left[\frac{Number\ of\ Codes\ Successfully\ decoded}{Total\ number\ of\ Encoded\ codes}\right] x100\%$$

There are four possible attack scenarios that could possibly change or corrupt the message encoded in the QR code. These errors are fixed by the scanner in order for it to display the original encoded message. We intentionally introduced errors on these areas and scanned it using the dedicated scanner, then compare it to the encoded message. If the decoded message matches the original encoded message, it is counted as one of the successfully decoded codes.

The statistical test used in the following tests is the Chi-Square Test. This kind of test was chosen because it was simple and it is commonly used for the goodness of fit, which is referred to how the results observed close to what it is expected based on the hypothesis given. The formula for the chi-square test is:

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

Where:

O is the observed data in each category

E is the expected data based on the hypothesis

$\sum$ is the sum of calculations for each category

X^2 is the chi-square value.

In every test conducted, there is a chi-square value that will be computed. In testing with red filters, the null hypothesis is that there will at least 95% will fail the test and the alternative hypothesis is that at least 5% of the tested QR code will be successfully decoded. The same hypotheses applies to the blue filters. In the test using green filter, the null hypothesis is that at least 95% of the QR codes will be decoded successfully and the alternative hypothesis is that at least 5% of the QR codes will pass. These hypotheses also applies in the testing of unmodified and modified QR codes. The degrees of freedom used is 1 and the level of significance used in the test is 0.05.

Conceptual Framework

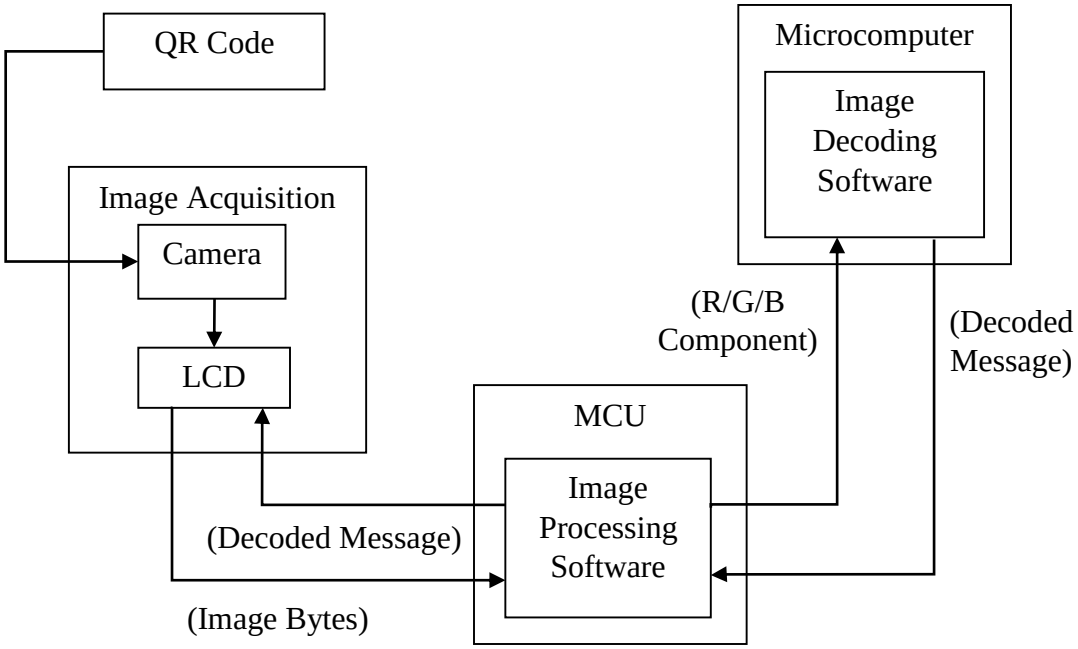


Figure 3.1: Conceptual Framework

QR Code image will be acquired using a camera module. Image coming from the camera is displayed on the LCD. The LCD display will temporarily store the image from the camera, then sends the image byte by byte to the MCU which processes the image to separate the RGB component. The red, green, or blue component of the image will be used by the decoder installed on a microcomputer to decode the image and send the result back to the MCU. The MCU will control the LCD to display the decoded message. Depending on the success rate of decoding the image, the group will choose the best component of the image that will be used for the experiment.

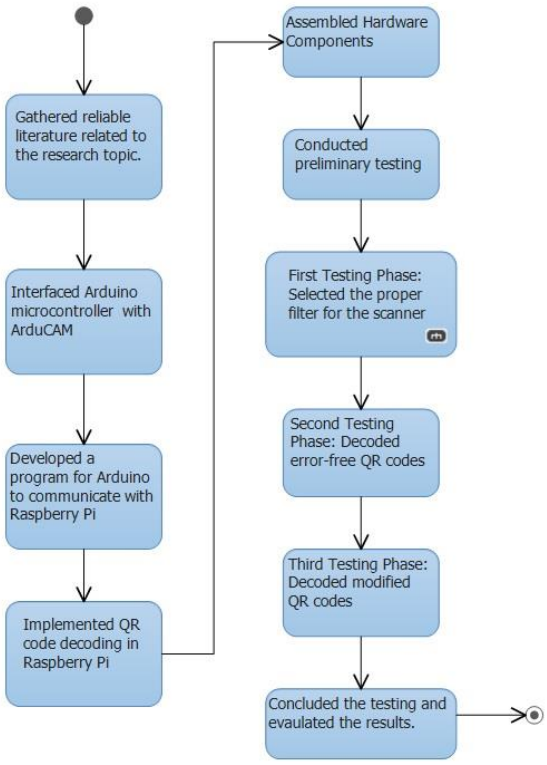


Figure 3.2: Project Activity Diagram

The project started with gathering of reliable resources for the related literature of the study. The studies gathered are related to QR code encoding and decoding algorithm, error correction and image processing. The hardware components that are needed in the project are also included. After the gathering the sources, the group started interfacing the Arduino microcontroller to the ArduCAM shield for the QR code scanner. Next is the group undergoes software development where the program for the Arduino to communicate with the Raspberry Pi was developed. The QR decoding was also implemented in Raspberry Pi. The hardware is then assembled to create the dedicated QR code scanner. When the communication between all the hardware components was successfully established. The hardware and software was undergone preliminary testing. This part of the test was to check whether the hardware and software was working properly and to eliminate bugs. The testing phase was then started after preliminary testing. There are three phases of testing. The first was evaluation and selection of the proper image filter to be used in decoding QR codes, second was the decoding of error-free QR-codes and lastly, decoding the modified QR-codes. All of these tests were recorded by the decoded message from the image, the time it takes from image capture until display of the results and the success rate of each tests. After all tests was finished, the group then evaluates the results of the tests and provided the conclusion of the project.

First Testing Phase: Selected the proper filter for the scanner

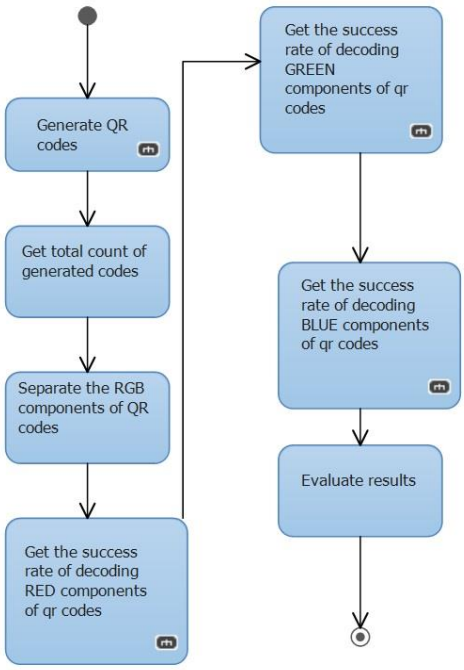


Figure 3.3: First testing phase activity diagram

The figure above shows the process of the first testing phase. The test is about the selection of the best filter for the decoding of the QR code. First is to generate the QR code. There are many QR code generators that can be found in the internet. The group used a QR code generator website such as QR Code 2D Barcode Generator from Raco Industries to generate QR codes. This website provides customization on the QR code including the encoding type and the level of error correction. After generating the QR codes, count the number of the generated QR codes. Separate the red, green and blue components in the QR code. This can be done using a software program inside the Arduino. Decode the QR codes per filter. In the figure, the group test the decoder using red filters. After all the QR codes were scanned, the group computes the success rate of the test by getting the number of matches over the total number of QR codes. This is done also using green and red filters. When all the filters

were tested, the group evaluates the results and selects the filter to be used in further testing of the hardware and the software.

Generating the QR codes:

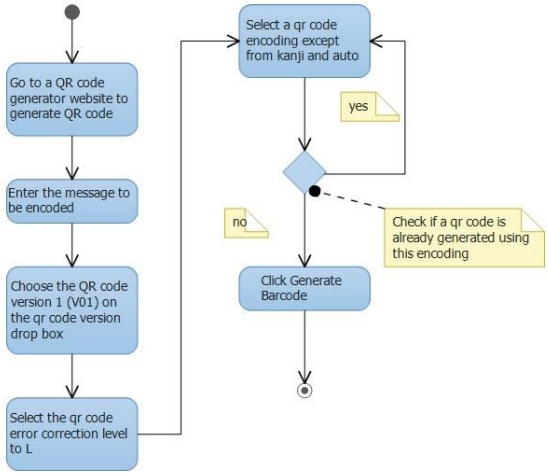


Figure 3.4: Generating the QR code

The figure above shows the process in generating the QR code. The website used in generating the QR code was mentioned in the previous flowchart explanation. To generate a QR code, go to a QR code generator website first then enter the message to be encoded in the QR code. Select the version of the QR code to be used. In our case, all of the QR codes that we generated from the mentioned website are in version 1. Next is to select the level of error correction. In our case, all QR codes that we generated from the website are generated using the lowest level of error correction L. Then select the QR code encoding. It should be numeric, alphanumeric, or binary. Kanji and Auto should not be selected. Generate the QR code by clicking on the generate barcode button located below the page after selecting the proper options. The generated barcode is displayed on the screen and can be downloaded to the local drive of the PC. In our case, all barcodes that we generated in this website are downloaded and

saved to be used in further tests. A total of six different QR codes with different character encoding can be generated, two QR codes for each character encoding criteria. The encoded message is also changed for each QR code that is generated. The generated QR code will be used in further tests.

Getting the Success rate of QR codes:

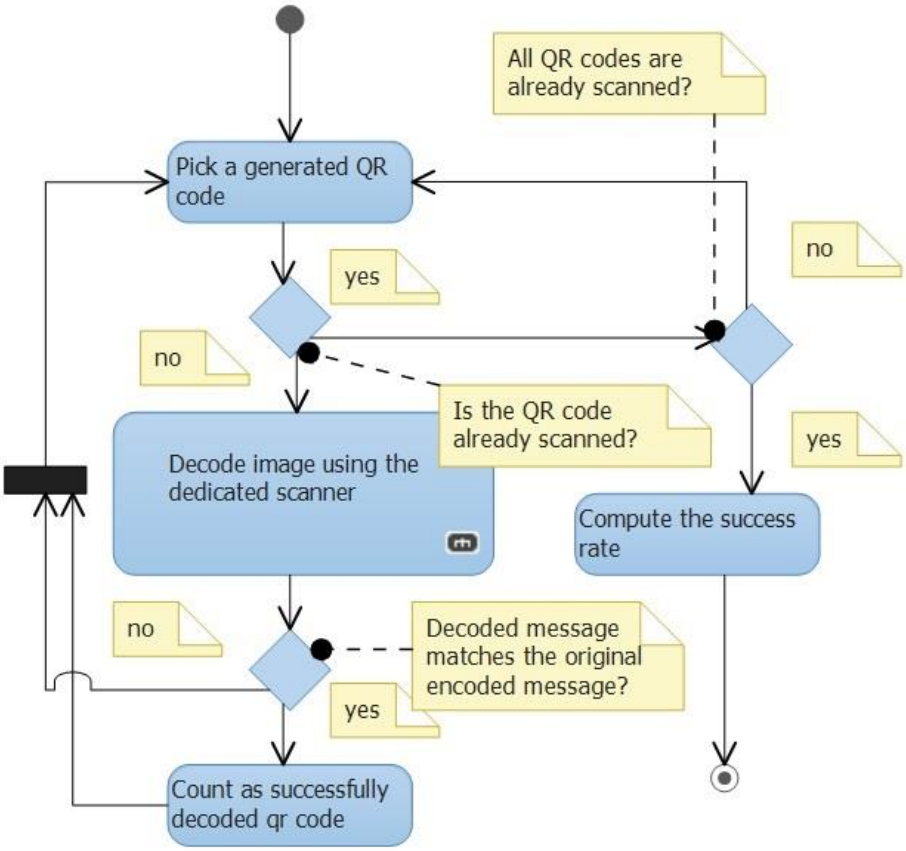


Figure 3.5: Activity Diagram for getting the success rate

In the figure above is the flowchart for getting the success rate of the QR codes. This is an important part of the test. First is to select a QR code to be tested from the pool of QR codes that we generated from the previous step. Next, after selecting the QR code, scan the QR code using the Dedicated MCU based QR code scanner. This is done first, by placing the QR

code image in front of the camera. Adjust the distance of the camera from the QR code image until the QR code image can be clearly displayed on the LCD. Press the trigger button located at the top right portion of the Arducam shield. The device will try to decode the scanned image afterwards. When the device successfully decodes the image, it will display the result on its LCD display. This is where the group compares the original message to the decoded message. If the decoded message matches the encoded message, it is counted as a success otherwise it is not. If the device didn't show any result after it process the image, it also means that the device was not successful on decoding the QR code. After all the QR codes were scanned, the group will compute for the success rate mentioned previously in the methodology.

Arduino System Activity Diagram

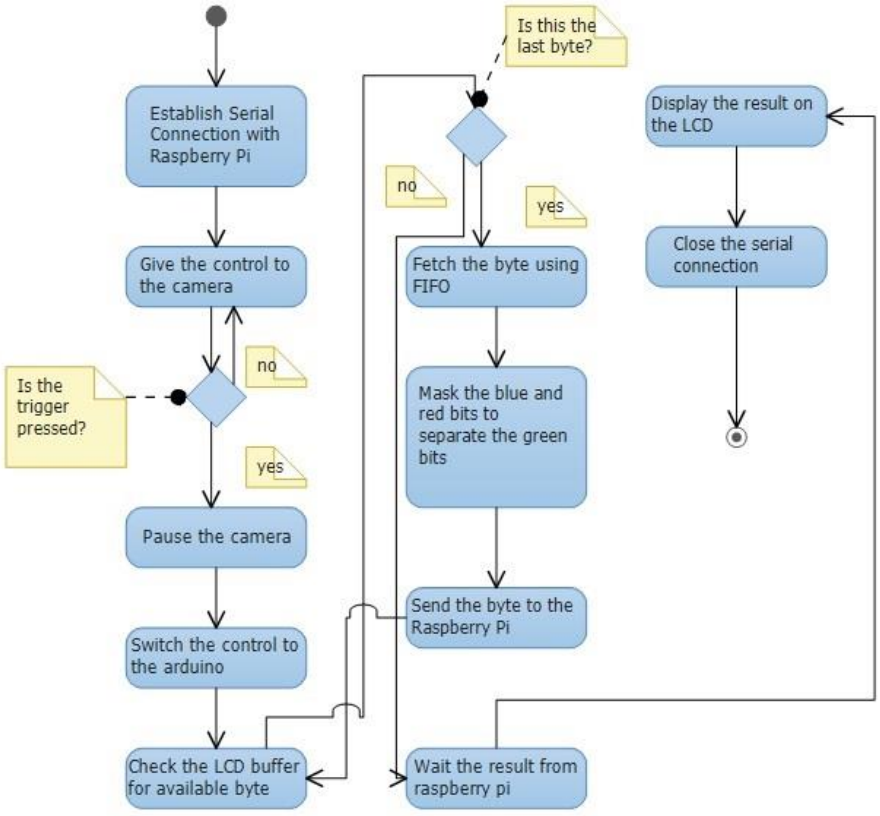


Figure 3.6: Arduino System Activity Diagram

The Arduino interfaced with the Arducam Shield Rev. B captures and processes the image. The image processing that is used is RGB filtering. This separates the red, green, and blue bits of the 16-bit color bit. The green filter is used in this research because among the three filters, only the green filter can be recognized by the scanner because it has the least amount of distortion. The Arduino fetches the image from the LCD buffer byte by byte using FIFO algorithm and processed each byte to separate the green component. After the byte is filtered, it is sent to the Raspberry Pi for QR code decoding. The Arduino waits for the Raspberry Pi to send the decoded data and displays it to the Arducam's LCD.

Raspberry PI System Activity Diagram

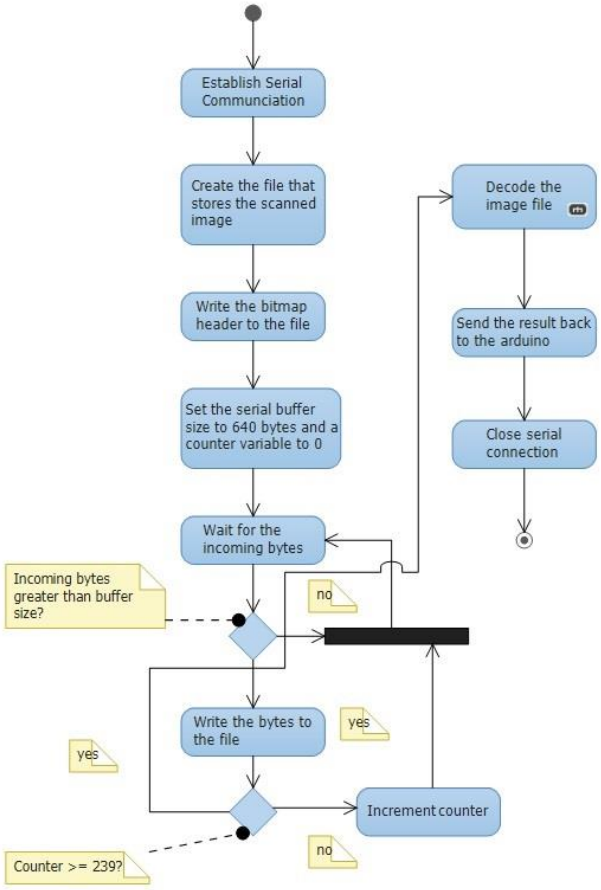


Figure 3.7: Raspberry Pi System Activity Diagram

The Raspberry pi is responsible for the implementation of the decoding algorithm of QR codes specified in the ISO/IEC 18004:2006. First is that the Raspberry PI establishes the connection to the Arduino. Each time the Raspberry Pi fills its 640 byte buffer with bytes coming from its serial port sent by Arduino, it writes the bytes to the file named "QR.bmp". The counter variable ensures that the file will not exceed the 240 by 320 pixel resolution of the bitmap image. The file is decoded by implementing the functions from the zbartools library, specifically the zbarimg. It is capable of displaying the decoded message as well as the time it takes to decode the message. The result is saved to a file named result.txt. The textfile is read by the program, and sends it back to the Arduino microcontroller. The files, QR.bmp and result.txt will be preserved in the Raspberry Pi until a new image was captured again from the Arduino microcontroller.

Image Decoding Algorithm Activity Diagram

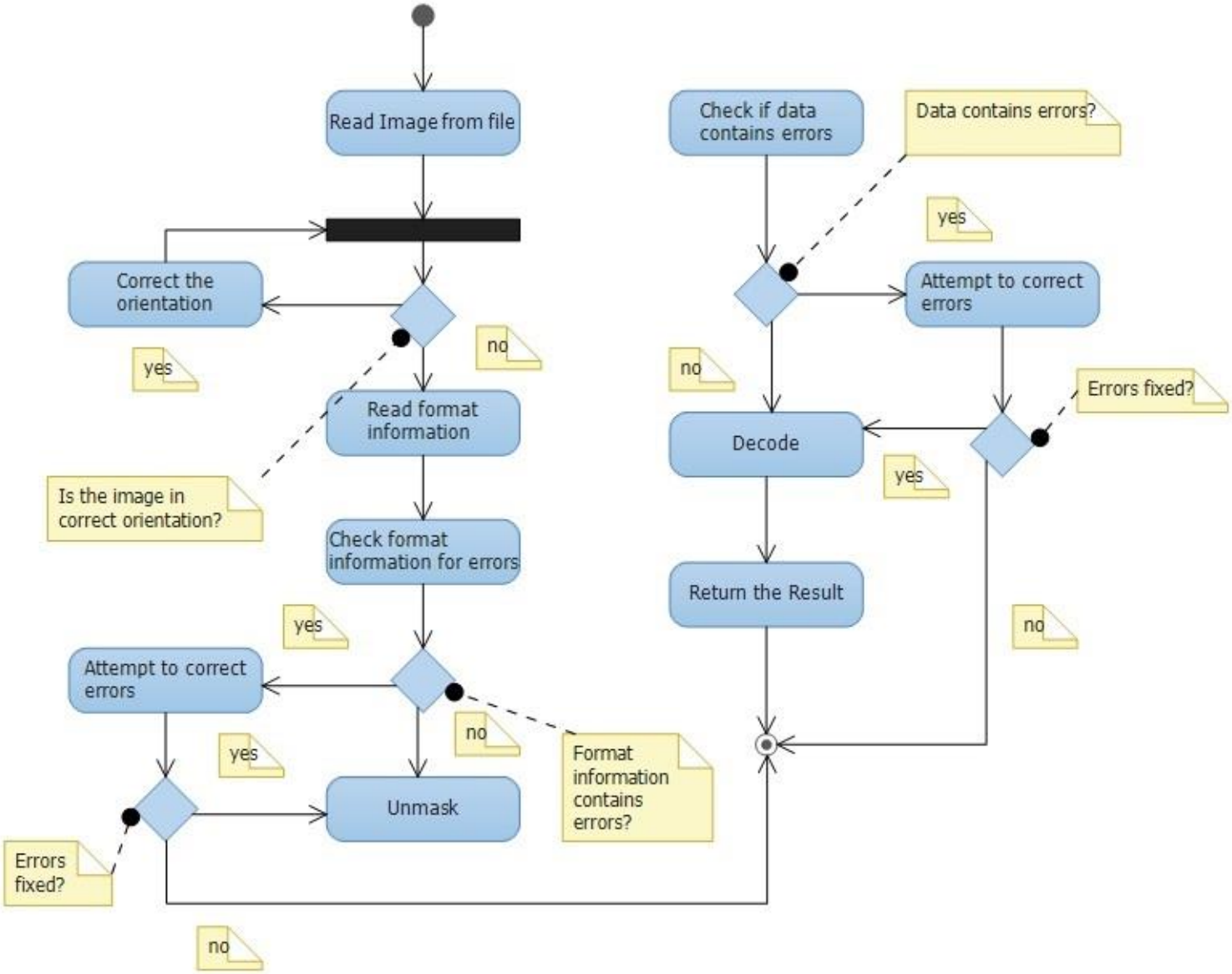


Figure 3.8: Image Decoding Algorithm Flowchart

First, image is checked whether it is in the correct orientation. The QR code has 3 locator patterns which allows the decoder to recognize whether it is in the correct orientation or not, rotating the image if necessary. Next, the decoder will read the format information of the QR code. The format information is checked for errors using the BCH error correction. The BCH error correction undergoes a series of division-like algorithm to the format information to check whether the encoded information was changed. The BCH error correction algorithm will result to 0 if there are no errors, and non-zero if there are errors. If it contains errors, the decoder will try to fix it by finding the nearest format information pattern that it matches. Once the format information is known, the type of masking is also known. The decoder will unmask the QR code by XORing the mask pattern that the encoder uses. After the QR code is unmasked, the data bits are read. Data bits are also checked for errors and corrected using the Reed-Solomon error correction. After fixing all errors, the decoded message is displayed.

Hardware Block Diagram

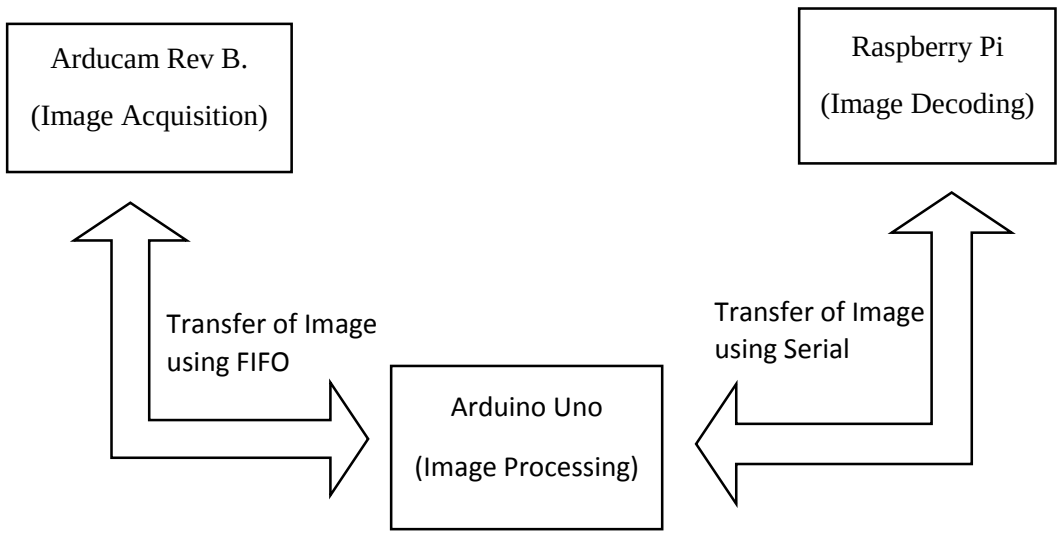


Figure 3.9: Block Diagram of the Dedicated MCU-based QR Code scanner

Arducam Rev. B. is an Arduino shield that can be directly attached to Arduino boards like Arduino Uno. It can capture an image using the camera module attached to it, and send the image to the Arduino microcontroller. The Arduino processes the image byte by byte. Each byte contains the RGB 565 information of the image which are modified by using simple operators such as module division and bit shifting operations. The result will be either the Red, Green, or Blue component of the acquired image. This image is passed to the Raspberry pi through a serial port, and then decodes the processed QR code image and sends back the result or the decoded message to the Arduino. The Arduino then sends the message to the Arducam shield's LCD display which shows the result to the user.

Schematic Diagram

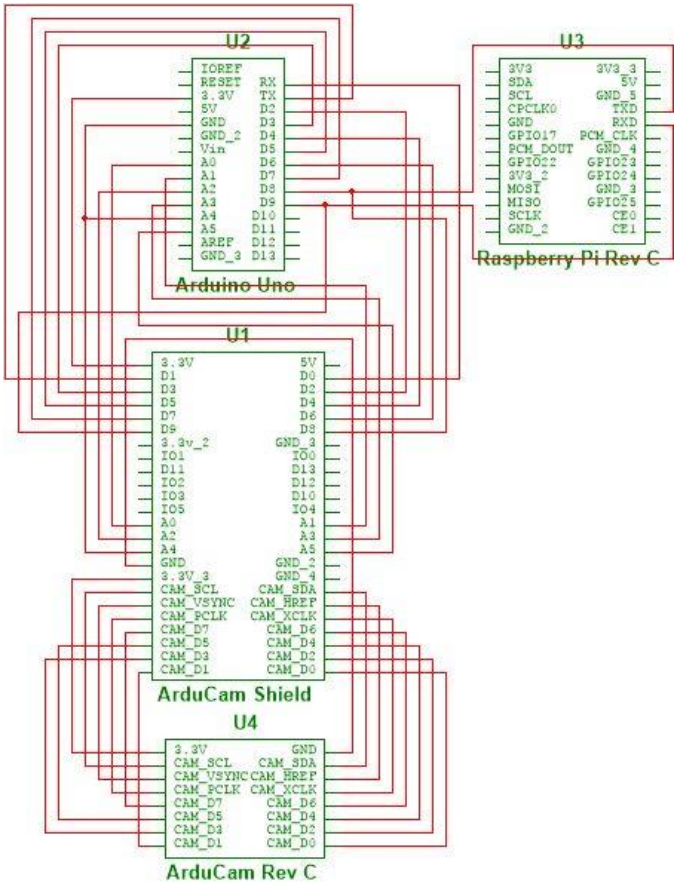


Figure 3.10: Schematic Diagram of the Prototype

In the schematic above, there are four components that are assembled for the thesis prototype: ArduCAM rev C, which is a 2-megapixel camera module that captures images containing QR codes; ArduCAM shield, where the ArduCAM module is attached. This shield also has display where the camera viewfinder and the results are displayed; Arduino UNO microcontroller, where the images were processed, separating colors using filters coming from the camera. It is also transfers and receives data to and from the Raspberry Pi and the ArduCAM shield; and Raspberry Pi, where the images coming from the camera are decoded and transfers back the results to the microcontroller. The ports D8 and D9 of Arduino serves as the data transfer and receiver to the TXD and RXD ports of the Raspberry Pi. The RX and TX ports of Arduino serves as the data transfer and receiver to ports D0 and D1 of the ArduCAM shield. The camera module is placed on top of the ArduCAM shield. The Raspberry Pi microcomputer also serves as the power source of the camera module, the shield and the microcontroller.

Testing

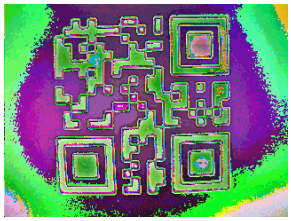
The original QR codes are already generated using an online QR code generator. The encoder is set to encode QR codes in version 1, with a level of error correction L. The mode of character encoding and the length of the message being encoded is different for each test.

Objective 1: Capture the image using the camera of the device and apply image processing to make the image practical for scanning. Scan the generated QR code using the dedicated scanner. Extract the original image captured by the hardware and capture the image again this time by using RGB filters. Modify the program to change the filters to red, green or blue.

Decode the images then write down the decoded message. Calculate the success rate in each tables and then select the best type of filter for decoding the QR code image to proceed to the next test.

In this test, the tables are separated by each color filter: Red, Green and Blue. This was done in that order. Before starting the test, the program should be modified first to proceed with this test. First is to modify the program so that the image extracted is original. The results are shown in the table below. The Generated QR code is in the left column and the captured image is in the right column of the table.

Table 3.1: Generated and Captured QR Code

Generated QR Code	Original Captured Image
	

The generated QR code in the left contains the 0123456789 message. It is encoded with QR code version 1, configured to the lowest error correction level, L and the character encoding is numeric encoding. The QR code in the right is the raw image extracted from the hardware. The image is the usual output of the camera. The program is based on the open source program included in the Arduino microcontroller. The next steps are to use RGB filters in each QR codes in the table.

First in the RGB filter test, use the red filter by modifying the program in the Arduino IDE then upload it to the microcontroller. There are six QR code image to be scanned and decoded. After scanning all the QR codes, check whether the results match the encoded message. Compute for its success rate by using the equation:

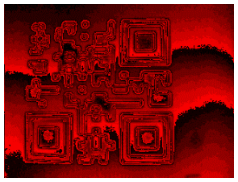
$$SUCCESS\ RATE = \left[\frac{N_{success}}{N_{total}}\right] x100\%$$

Where,

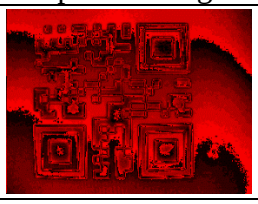
$N_{success}$ is the number of successfully decoded images

N_{total} is the total number of encoded image

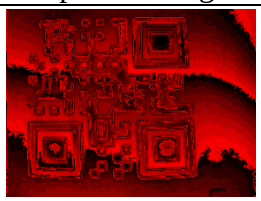
Table 3.2: Extracted Images using Red Filter:

QR # 1	Captured Image:
	
Encoded Message: Hello	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains "Hello" in alphanumeric encoding. This code is captured by the camera as seen in the right side of the table. This captured image cannot be decoded because the QR code cannot be detected clearly in the image on the right. The test was unsuccessful.

QR # 2	Captured Image:
	
Encoded Message: Hello World	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains "Hello World" in alphanumeric encoding. The image on the right side of the table is what the camera has processed. The QR code cannot be detected in the captured image because it was distorted. There is no decoded message as a result of this test.

QR # 3	Captured Image:
	
Encoded Message: Hi	Decoded Message: NONE
Encoding: Binary	
	Result: FAILED

The QR code on the left contains “Hi” encoded in binary. The image on the right side is the captured image using the camera. As seen on the captured image, the QR code can’t be decoded properly because of its distorted features. This test was failure as a result.

QR # 4	Captured Image:
	
Encoded Message: Hi World	Decoded Message: NONE
Encoding: Binary	
	Result: FAILED

The QR code on the left contains “Hi World” using binary encoding. The captured image is in the right side of the table. That image cannot be decoded because the scanner cannot detect the QR code properly. The QR code is still present but it is not properly processed. As a result, the scanner returns no decoded message.

QR # 5	Captured Image:
	
Encoded Message: 12345	Decoded Message: NONE
Encoding: Numeric	
	Result: FAILED

The QR code on the left contains the word “12345” encoded in numeric. In the right side is the captured image. The captured image was distorted and not very detailed. The scanner cannot properly detect the QR code on the captured image. As a result, this test was a failure.

QR # 6	Captured Image:
	
Encoded Message: 0123456789	Decoded Message: NONE
Encoding: Numeric	
	Result: FAILED

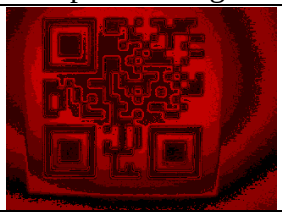
The QR code on the left side contains “0123456789” encoded in numeric. The one on the right side is the captured image using the scanner. When the decoder was trying to detect the QR code, it failed to do so because the image was distorted. As a result of this test, nothing was decoded in that image.

QR # 7	Captured Image:
	
Encoded Message: ####Sample####	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains the message “####Sample####” encoded in alphanumeric. The image on the right side is the captured image. The QR code was scanned and decoded by the camera. There was no message returned to the display. This means that the camera doesn’t recognize the code from the image properly.

QR # 8	Captured Image:
	
Encoded Message: @qrcode#sample%	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains the message "@qrcode#sample%" encoded in alphanumeric. The image on the right side is the captured image. The camera scans the image for QR code and tried to decode it. The camera fails to find the QR code image. The QR code in the image has some unnecessary noise in it that it cannot be decoded properly.

QR # 9	Captured Image:
	
Encoded Message: ~Message Text!@	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains the message "~Message Text!@"encoded in alphanumeric. The image on the right side is the captured image. It is then scanned for QR codes then it will be decoded. The QR code is visible from the captured image but the decoder cannot decode the message in there. The QR code has some noise in it that the data inside it cannot be retrieved.

QR # 10	Captured Image:
	
Encoded Message: 00012345000	Decoded Message: NONE
Encoding: Numeric	
	Result: FAILED

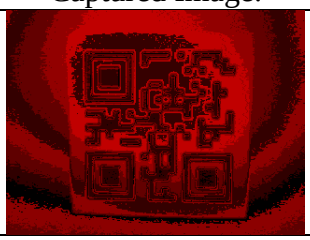
The QR code on the left contains the message “00012345000”encoded in binary. The image on the right side is the captured image. The camera scans for QR codes. As seen in the captured image, the QR code is visible but there are noises in there. The camera fails to decode the QR code. The noises affects the QR codes and as a result of it, the test fails.

QR # 11	Captured Image:
	
Encoded Message: 246813579	Decoded Message: NONE
Encoding: Numeric	
	Result: FAILED

The QR code on the left contains the message “246813579”encoded in numeric. The image on the right side is the captured image. The image has noises in it. These affects the QR code information. The camera tries to scan the image and decode the QR code in it. There is no message returned to the display. The test fails as a result.

QR # 12	Captured Image:
	
Encoded Message: 98124758392	Decoded Message: NONE
Encoding: Numeric	
	Result: FAILED

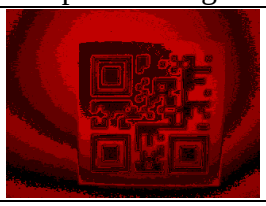
The QR code on the left contains the message “98124758392” encoded in numeric. The image on the right side is the captured image. The image was scanned by the camera for QR codes. The QR code was detected and decoded but there is no message returned. The image has some noises in it that affects the whole QR code. As a result, this test was a failure.

QR # 13	Captured Image:
	
Encoded Message: 110101011101	Decoded Message: NONE
Encoding: Binary	
	Result: FAILED

The QR code on the left contains “110101011101” in binary encoding. This code is captured by the camera as seen in the right side of the table. This captured image cannot be decoded because the QR code cannot be detected clearly in the image on the right. The test was unsuccessful.

QR # 14	Captured Image:
	
Encoded Message: 0000101010011100101010	Decoded Message: NONE
Encoding: Binary	
	Result: FAILED

The QR code on the left contains “0000101010011100101010” in binary encoding. The image on the right side of the table is what the camera has processed. The QR code cannot be detected in the captured image because it was distorted. There is no decoded message as a result of this test.

QR # 15	Captured Image:
	
Encoded Message: ABCDefgh	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains “ABCDefgh” encoded in alphanumeric. The image on the right side is the captured image using the camera. As seen on the captured image, the QR code can’t be decoded properly because of its distorted features. This test was failure as a result.

QR # 16	Captured Image:
	
Encoded Message: Dario Ty	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

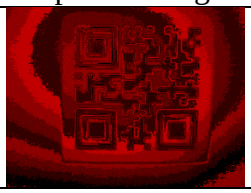
The QR code on the left contains “Dario Ty” using alphanumeric encoding. The captured image is in the right side of the table. That image cannot be decoded because the scanner cannot detect the QR code properly. The QR code is still present but it is not properly processed. As a result, the scanner returns no decoded message.

QR # 17	Captured Image:
	
Encoded Message: facebook.com/home	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

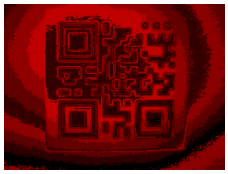
The QR code on the left contains the message “facebook.com/home” encoded in alphanumeric. The image on the right side is the captured image. The QR code was scanned and decoded by the camera. There was no message returned to the display. This means that the camera doesn’t recognize the code from the image properly.

QR # 18	Captured Image:
	
Encoded Message: google.com.ph	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left side contains “google.com.ph” encoded in alphanumeric. The one on the right side is the captured image using the scanner. When the decoder was trying to detect the QR code, it failed to do so because the image was distorted. As a result of this test, nothing was decoded in that image.

QR # 19	Captured Image:
	
Encoded Message: Hello Operator1	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

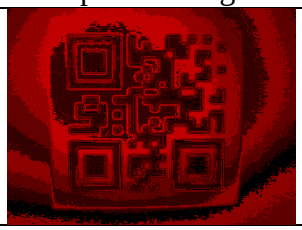
The QR code on the left contains the message “Hello Operator1” encoded in alphanumeric. The image on the right side is the captured image. The camera scans the image for QR code and tried to decode it. The camera fails to find the QR code image. The QR code in the image has some unnecessary noise in it that it cannot be decoded properly.

QR # 20	Captured Image:
	
Encoded Message: hello00910	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains the message “hello00910”encoded in alphanumeric. The image on the right side is the captured image. It is then scanned for QR codes then it will be decoded. The QR code is visible from the captured image but the decoder cannot decode the message in there. The QR code has some noise in it that the data inside it cannot be retrieved.

QR # 21	Captured Image:
	
Encoded Message: Mapua Institute	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains the message “Mapua Institute” encoded in alphanumeric. The image on the right side is the captured image. The camera scans for QR codes. As seen in the captured image, the QR code is visible but there are noises in there. The camera fails to decode the QR code. The noises affects the QR codes and as a result of it, the test fails.

QR # 22	Captured Image:
	
Encoded Message: Mapua Intramuros	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains the message “Mapua Intramuros” encoded in alphanumeric. The image on the right side is the captured image. The image has noises in it.

These affects the QR code information. The camera tries to scan the image and decode the QR code in it. There is no message returned to the display. The test fails as a result.

QR # 23	Captured Image:
	
Encoded Message: Mark Domingo	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

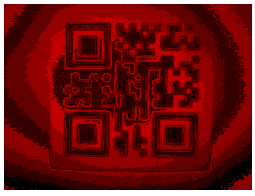
The QR code on the left contains the message “Mark Domingo” encoded in alphanumeric. The image on the right side is the captured image. The image was scanned by the camera for QR codes. The QR code was detected and decoded but there is no message returned. The image has some noises in it that affects the whole QR code. As a result, this test was a failure.

QR # 24	Captured Image:
	
Encoded Message: Patrick Jove	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains “Patrick Jove” in alphanumeric encoding. This code is captured by the camera as seen in the right side of the table. This captured image cannot be decoded because the QR code cannot be detected clearly in the image on the right. The test was unsuccessful.

QR # 25	Captured Image:
	
Encoded Message: qr@raco.com!	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left side contains “qr@raco.com!” encoded in alphanumeric. The one on the right side is the captured image using the scanner. When the decoder was trying to detect the QR code, it failed to do so because the image was distorted. As a result of this test, nothing was decoded in that image.

QR # 26	Captured Image:
	
Encoded Message: readablecode#30	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains “readablecode#30” using alphanumeric encoding. The captured image is in the right side of the table. That image cannot be decoded because the scanner cannot detect the QR code properly. The QR code is still present but it is not properly processed. As a result, the scanner returns no decoded message.

QR # 27	Captured Image:
	
Encoded Message: sampleQRCODE	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

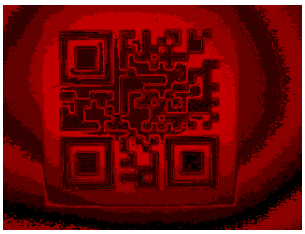
The QR code on the left contains “sampleQRCODE” in alphanumeric encoding. The image on the right side of the table is what the camera has processed. The QR code cannot be detected in the captured image because it was distorted. There is no decoded message as a result of this test.

QR # 28	Captured Image:
	
Encoded Message: !@#$%^&*()_+|	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains the message “!@#$%^&*()_+|” encoded in alphanumeric. The image on the right side is the captured image. The QR code was scanned and decoded by the camera. There was no message returned to the display. This means that the camera doesn’t recognize the code from the image properly.

QR # 29	Captured Image:
	
Encoded Message: Thesis Sample	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains the message “Thesis Sample “encoded in alphanumeric. The image on the right side is the captured image. The image has noises in it. These affects the QR code information. The camera tries to scan the image and decode the QR code in it. There is no message returned to the display. The test fails as a result.

QR # 30	Captured Image:
	
Encoded Message: wikipedia.org	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left contains "wikipedia.org" using alphanumeric encoding. The captured image is in the right side of the table. That image cannot be decoded because the scanner cannot detect the QR code properly. The QR code is still present but it is not properly processed. As a result, the scanner returns no decoded message.

$$SUCCESS\ RATE = \left[\frac{0}{30}\right] x100\% = 0\ \%$$

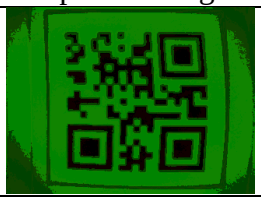
Table 3.3: Chi-Square Table for the Red Filter:

	OBSERVED (O)	EXPECTED (E)	(O-E)	$(O-E)^2$	$\frac{(O-E)^2}{2}$
DETECTED	0	2	-2	4	2
NOT DETECTED	30	28	2	4	0.142857
TOTAL	30	30			**2.142857**

Based on the results, the QR code image scanned cannot be recognized by the decoder program of the microcontroller. It is observed that the scanned images are somehow distorted and makes the microcontroller having a hard time to recognize the whole QR code image. Overall, the test proves that the red filter was not viable for decoding, giving a 0% success rate. The chi-square value is 2.142857. The degrees of freedom is 1 and the level of significance is 0.05. The value obtained lies between 2.07 and 3.84 or 2.07<P<3.84. As a result of this, we accept the null hypothesis for red filters which is that at least 95% of the tested QR codes will not be successfully decoded

The next test is to use the green filter by modifying the program in the Arduino IDE then upload it to the microcontroller. There are six QR code image to be scanned and decoded. After scanning all the QR codes, check whether the results match the encoded message.

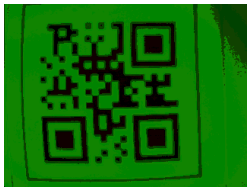
Table 3.4: Extracted Images using Green Filter:

QR # 1	Captured Image:
	
Encoded Message: Hello	Decoded Message: Hello
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “Hello”. The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The camera decoded the QR code properly and as a result, it returned the same message from the original.

QR # 2	Captured Image:
	
Encoded Message: Hello World	Decoded Message: Hello World
Encoding: Alphanumeric	
	Result: SUCCESS

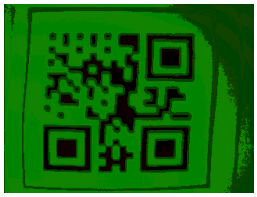
The QR code on the left is encoded with the word “Hello World”. The character encoding of the QR code is in alphanumeric mode. The decoded message from the camera matches the encoded QR code which is in the right side. This test was a success because the QR code captured was recognized by the decoder inside the camera.

QR # 3	Captured Image:
	
Encoded Message: Hi	Decoded Message: Hi
Encoding: Binary	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “Hi” encoded in binary. The decoded message from the camera matches the encoded QR code. This test was a success because the captured image on the right of the table shows that the structure of the QR code is still clear and can be recognized and detected by the camera.

QR # 4	Captured Image:
	
Encoded Message: Hi World	Decoded Message: Hi World
Encoding: Binary	
	Result: SUCCESS

The QR code on the left is encoded with the word “Hi World” encoded in binary. The image on the right was the captured image. The QR code was clearly present in the captured image and as a result, the decoder returned the original message. The test was a success.

QR # 5	Captured Image:
	
Encoded Message: 12345	Decoded Message: 12345
Encoding: Numeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “12345” encoded in numeric. The captured image is in the right side. This was decoded by the camera. The decoded message from the camera matches the encoded QR code because the captured image shows a QR code properly.

QR # 6	Captured Image:
	
Encoded Message: 0123456789	Decoded Message: 0123456789
Encoding: Numeric	
	Result: SUCCESS

The QR code on the left side contains “0123456789” encoded in numeric. The QR code on the right side is the captured image using the camera. This captured image was decoded using the decoder inside the camera. The decoded message displayed the same as the original message. This was a success as a result of this.

QR # 7	Captured Image:
	
Encoded Message: ####Sample####	Decoded Message: ####Sample####
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “####Sample####”. The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The QR code is visible from the captured image. The camera decoded the QR code and it returned the exact message as the original. This test was a success.

QR # 8	Captured Image:
	
Encoded Message: @qrcode#sample%	Decoded Message: @qrcode#sample%
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left is encoded with the word "@qrcode#sample%". The character encoding of the QR code is in alphanumeric mode. The QR code from the captured image was decoded by the camera. The camera returned the message the same as from the generated QR code. As a result, this test was a success.

QR # 9	Captured Image:
	
Encoded Message: ~Message Text!@	Decoded Message: ~Message Text!@
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word "~Message Text!@" encoded in alphanumeric. The captured image was scanned and decode the QR code from it. The camera returns the same message stored in the original. As a result of this test, it was recorded as a success.

QR # 10	Captured Image:
	
Encoded Message: 00012345000	Decoded Message: 00012345000
Encoding: Numeric	
	Result: SUCCESS

The QR code on the left is encoded with the word “00012345000” encoded in numeric. The image on the right was the captured image. The QR code was clearly present in the captured image and as a result, the decoder returned the original message. The test was a success.

QR # 11	Captured Image:
	
Encoded Message: 246813579	Decoded Message: 246813579
Encoding: Numeric	
	Result: SUCCESS

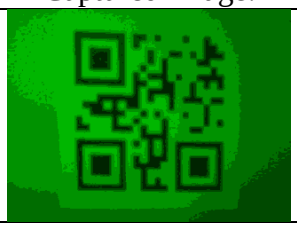
The QR code on the left of the table is encoded with the word “246813579” encoded in numeric. The captured image is in the right side. This was decoded by the camera. The decoded message from the camera matches the encoded QR code because the captured image shows a QR code properly.

QR # 12	Captured Image:
	
Encoded Message: 98124758392	Decoded Message: 98124758392
Encoding: Numeric	
	Result: SUCCESS

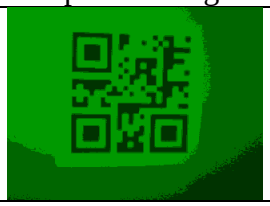
The QR code on the left side contains “98124758392” encoded in numeric. The QR code on the right side is the captured image using the camera. This captured image was decoded using the decoder inside the camera. The decoded message displayed the same as the original message. This was a success as a result of this.

QR # 13	Captured Image:
	
Encoded Message: 110101011101	Decoded Message: 110101011101
Encoding: Binary	
	Result: SUCCESS

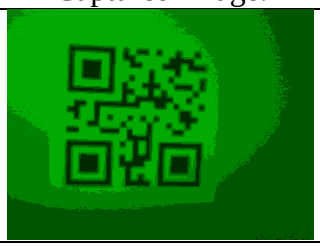
The QR code on the left of the table is encoded with the word “110101011101”. The character encoding of the QR code is in binary mode. The image on the right is the captured QR code. The camera decoded the QR code properly and as a result, it returned the same message from the original.

QR # 14	Captured Image:
	
Encoded Message: 000010101001110010101O	Decoded Message: 000010101001110010101O
Encoding: Binary	
	Result: SUCCESS

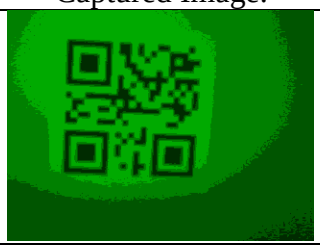
The QR code on the left is encoded with the word “000010101001110010101O”. The character encoding of the QR code is in binary mode. The decoded message from the camera matches the encoded QR code which is in the right side. This test was a success because the QR code captured was recognized by the decoder inside the camera.

QR # 15	Captured Image:
	
Encoded Message: ABCDefgh	Decoded Message: ABCDefgh
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “ABCDefgh” encoded in alphanumeric. The decoded message from the camera matches the encoded QR code. This test was a success because the captured image on the right of the table shows that the structure of the QR code is still clear and can be recognized and detected by the camera.

QR # 16	Captured Image:
	
Encoded Message: Dario Ty	Decoded Message: Dario Ty
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left is encoded with the word “Dario Ty” encoded alphanumeric. The image on the right was the captured image. The QR code was clearly present in the captured image and as a result, the decoder returned the original message. The test was a success.

QR # 17	Captured Image:
	
Encoded Message: facebook.com/home	Decoded Message: facebook.com/home
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word facebook.com/home encoded in alphanumeric. The captured image was scanned and decode the QR code from it. The camera returns the same message stored in the original. As a result of this test, it was recorded as a success.

QR # 18	Captured Image:
	
Encoded Message: google.com.ph	Decoded Message: google.com.ph
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left is encoded with the word “google.com.ph”. The character encoding of the QR code is in alphanumeric mode. The QR code from the captured image was decoded by the camera. The camera returned the message the same as from the generated QR code. As a result, this test was a success.

QR # 19	Captured Image:
	
Encoded Message: Hello Operator1	Decoded Message: Hello Operator1
Encoding: Alphanumeric	
	Result: SUCCESS

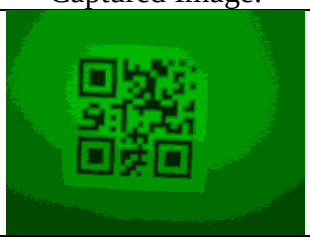
The QR code on the left of the table is encoded with the word “Hello Operator1”. The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The QR code is visible from the captured image. The camera decoded the QR code and it returned the exact message as the original. This test was a success.

QR # 20	Captured Image:
	
Encoded Message: hello00910	Decoded Message: hello00910
Encoding: Alphanumeric	
	Result: SUCCESS

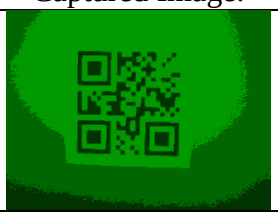
The QR code on the left side contains “hello00910” encoded in alphanumeric. The QR code on the right side is the captured image using the camera. This captured image was decoded using the decoder inside the camera. The decoded message displayed the same as the original message. This was a success as a result of this.

QR # 21	Captured Image:
	
Encoded Message: Mapua Institute	Decoded Message: Mapua Institute
Encoding: Alphanumeric	
Result: SUCCESS	

The QR code on the left of the table is encoded with the word “Mapua Institute” encoded in alphanumeric. The captured image is in the right side. This was decoded by the camera. The decoded message from the camera matches the encoded QR code because the captured image shows a QR code properly.

QR # 22	Captured Image:
	
Encoded Message: Mapua Intramuros	Decoded Message: Mapua Intramuros
Encoding: Alphanumeric	
Result: SUCCESS	

The QR code on the left of the table is encoded with the word “Mapua Intramuros”. The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The camera decoded the QR code properly and as a result, it returned the same message from the original.

QR # 23	Captured Image:
	
Encoded Message: Mark Domingo	Decoded Message: Mark Domingo
Encoding: Alphanumeric	
	Result: SUCCESS

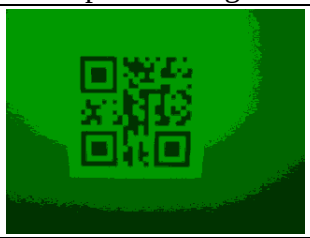
The QR code on the left is encoded with the word “Mark Domingo”. The character encoding of the QR code is in alphanumeric mode. The decoded message from the camera matches the encoded QR code which is in the right side. This test was a success because the QR code captured was recognized by the decoder inside the camera.

QR # 24	Captured Image:
	
Encoded Message: Patrick Jove	Decoded Message: Patrick Jove
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “Patrick Jove” encoded in alphanumeric. The decoded message from the camera matches the encoded QR code. This test was a success because the captured image on the right of the table shows that the structure of the QR code is still clear and can be recognized and detected by the camera.

QR # 25	Captured Image:
	
Encoded Message: qr@raco.com!	Decoded Message: qr@raco.com!
Encoding: Alphanumeric	
	Result: SUCCESS

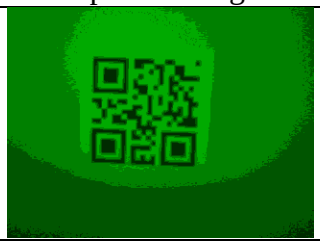
The QR code on the left side contains “qr@raco.com!” encoded in alphanumeric. The QR code on the right side is the captured image using the camera. This captured image was decoded using the decoder inside the camera. The decoded message displayed the same as the original message. This was a success as a result of this.

QR # 26	Captured Image:
	
Encoded Message: readablecode#30	Decoded Message: readablecode#30
Encoding: Alphanumeric	
	Result: SUCCESS

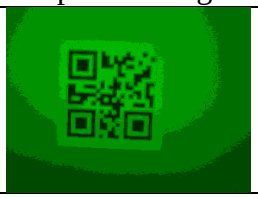
The QR code on the left of the table is encoded with the word “readablecode#30”. The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The QR code is visible from the captured image. The camera decoded the QR code and it returned the exact message as the original. This test was a success.

QR # 27	Captured Image:
	
Encoded Message: sampleQRCODE	Decoded Message: sampleQRCODE
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “sampleQRCODE”.encoded in alphanumeric. The captured image was scanned and decode the QR code from it. The camera returns the same message stored in the original. As a result of this test, it was recorded as a success.

QR # 28	Captured Image:
	
Encoded Message: !@#$%^&*()_+|	Decoded Message: !@#$%^&*()_+|
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left is encoded with the word "!@#$%^&*()_+|". The character encoding of the QR code is in alphanumeric mode. The QR code from the captured image was decoded by the camera. The camera returned the message the same as from the generated QR code. As a result, this test was a success.

QR # 29	Captured Image:
	
Encoded Message: Thesis Sample	Decoded Message: Thesis Sample
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word "Thesis Sample". The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The QR code is visible from the captured image. The camera decoded the QR code and it returned the exact message as the original. This test was a success.

QR # 30	Captured Image:
	
Encoded Message: wikipedia.org	Decoded Message: wikipedia.org
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word "wikipedia.org". The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The camera decoded the QR code properly and as a result, it returned the same message from the original.

$$SUCCESS\ RATE = \left[\frac{30}{30}\right] x100\% = 100\ \%$$

Table 3.5: Chi-Square Table for the Green Filter:

	OBSERVED (O)	EXPECTED (E)	(O-E)	$(O-E)^2$	$\frac{(O-E)^2}{2}$
DETECTED	30	28	2	4	0.142857
NOT DETECTED	0	2	-2	4	2
TOTAL	30	30			**2.142857**

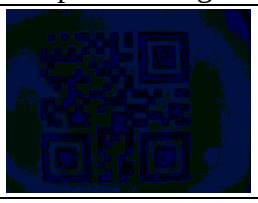
Based on the results, some QR code image scanned can be recognized by the decoder program of the microcontroller. The important features of the QR code is very clear in the scanned QR codes. This makes the microcontroller recognize the whole QR code image in the whole test. This gives the test a 100% success rate. The chi-square value is 2.142857. The degrees of freedom is 1 and the level of significance is 0.05. The value obtained lies between 2.07 and 3.84 or 2.07<P<3.84. As a result of this, we accept the null hypothesis for green filters which is that at least 95% of the tested QR codes will be successfully decoded

The next test is to use the blue filter by modifying the program in the Arduino IDE then upload it to the microcontroller. There are six QR code image to be scanned and decoded. After scanning all the QR codes, check whether the results match the encoded message.

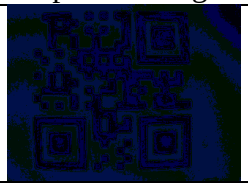
Table 3.6: Extracted Images using Blue Filter:

QR # 1	Captured Image:
	
Encoded Message: Hello	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

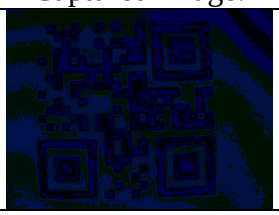
The QR code on the left is encoded with the word “Hello” encoded in alphanumeric mode. The decoded message from the camera which is in the right side of the table does not match the encoded QR code. This test was a failure because the captured image was too dark to be recognized by the camera.

QR # 2	Captured Image:
	
Encoded Message: Hello World	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

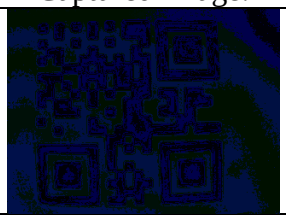
The QR code on the left is encoded with the word “Hello World” encoded in alphanumeric mode. The image in the right side is the captured image from the camera. The camera can’t identify the QR code in the captured image because it was dark and distorted. This test was a failure.

QR # 3	Captured Image:
	
Encoded Message: Hi	Decoded Message: NONE
Encoding: Binary	
	Result: FAILED

The QR code on the left is encoded with the word “Hi” encoded in binary. The image on the right side of the table was scanned to decode a QR code. The camera doesn’t find any QR code in the image even though it was barely visible there. This test was a failure.

QR # 4	Captured Image:
	
Encoded Message: Hi World	Decoded Message: NONE
Encoding: Binary	
	Result: FAILED

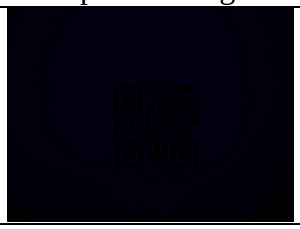
The QR code on the left is encoded with the word “Hi World” encoded in binary mode. The image on the right was scanned and tried to decode but the camera fails to do so. The QR code was barely visible from the captured image. This test was a failure as a result.

QR # 5	Captured Image:
	
Encoded Message: 12345	Decoded Message: NONE
Encoding: Numeric	
	Result: FAILED

The QR code on the left is encoded with the word “12345” encoded in numeric mode. This QR code was scanned using the camera programmed with blue filters. The camera couldn’t identify the QR code in the captured image because it was barely recognizable. This test was a failure as a result.

QR # 6	Captured Image:
	
Encoded Message: 0123456789	Decoded Message: NONE
Encoding: Numeric	
	Result: FAILED

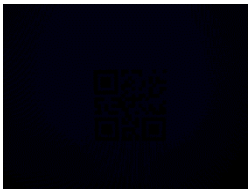
The QR code on the left is encoded with the word “0123456789” encoded in numeric mode. This QR code was scanned using the camera programmed with blue filters. The image on the right side is the captured image and as seen there, the image was dark and the QR code was unnoticeable. There is no returned message and this test was a failure.

QR # 7	Captured Image:
	
Encoded Message: ####Sample####	Decoded Message: NONE
Encoding: Alphanumeric	
	Result:FAILED

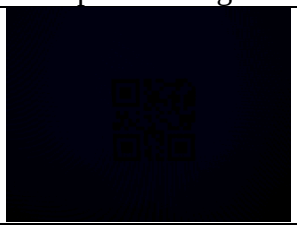
The QR code on the left is encoded with the word “####Sample####” encoded in alphanumeric mode. The decoded message from the camera which is in the right side of the table does not match the encoded QR code. This test was a failure because the captured image was too dark that the QR code can be barely seen.

QR # 8	Captured Image:
	
Encoded Message: @qrcode#sample%	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

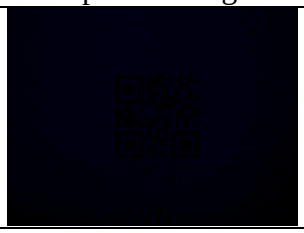
The QR code on the left is encoded with the word “@qrcode#sample%” encoded in alphanumeric mode. The image in the right side is the captured image from the camera. The camera can’t identify the QR code in the captured image because it was dark and the code was barely visible. This test was a failure.

QR # 9	Captured Image:
	
Encoded Message: ~Message Text!@	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

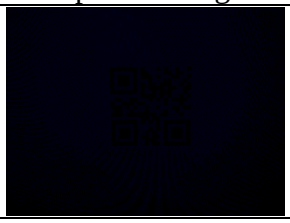
The QR code on the left is encoded with the word “~Message Text!@” encoded in alphanumeric. The image on the right side of the table was scanned to decode a QR code. The camera doesn’t find any QR code in the image even though it was barely visible there. This test was a failure.

QR # 10	Captured Image:
	
Encoded Message: 00012345000	Decoded Message: NONE
Encoding: Numeric	
	Result: FAILED

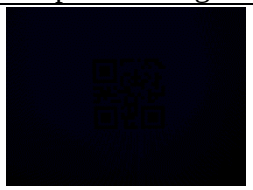
The QR code on the left side contains “00012345000” encoded in numeric. The one on the right side is the captured image using the scanner. When the decoder was trying to detect the QR code, it failed to do so because the QR code was not visible enough. As a result of this test, nothing was decoded in that image.

QR # 11	Captured Image:
	
Encoded Message: 246813579	Decoded Message: NONE
Encoding: Numeric	
	Result: FAILED

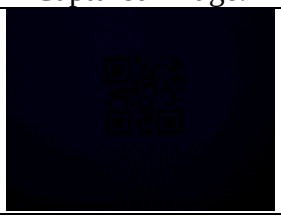
The QR code on the left side contains “246813579” encoded in numeric. The one on the right side is the captured image using the scanner. When the decoder was trying to detect the QR code, it failed to do so because the code was barely visible. As a result of this test, nothing was decoded in that image.

QR # 12	Captured Image:
	
Encoded Message: 98124758392	Decoded Message: NONE
Encoding: Numeric	
	Result: FAILED

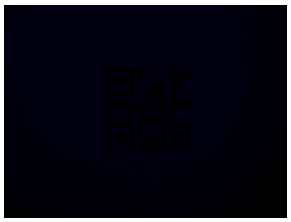
The QR code on the left is encoded with the word “98124758392” encoded in numeric mode. The decoded message from the camera which is in the right side of the table does not match the encoded QR code. This test was a failure because the captured image was too dark to be recognized by the camera.

QR # 13	Captured Image:
	
Encoded Message: 110101011101	Decoded Message: NONE
Encoding: Binary	
	Result: FAILED

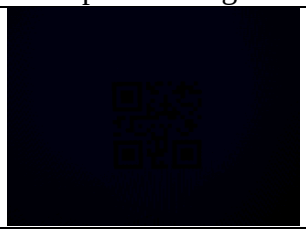
The QR code on the left is encoded with the word “110101011101” encoded in binary mode. This QR code was scanned using the camera programmed with blue filters. The camera couldn’t identify the QR code in the captured image because it was barely recognizable. This test was a failure as a result.

QR # 14	Captured Image:
	
Encoded Message: 0000101010011100101010	Decoded Message: NONE
Encoding: Binary	
	Result: FAILED

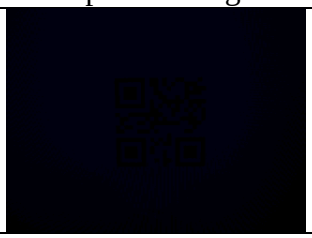
The QR code on the left contains “0000101010011100101010” encoded in binary. The one on the right is the captured image using the camera. The scanner cannot decode the message because the image on the right is dark it cannot see the QR code. As a result, there is no decoded message in the image.

QR # 15	Captured Image:
	
Encoded Message: ABCDefgh	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

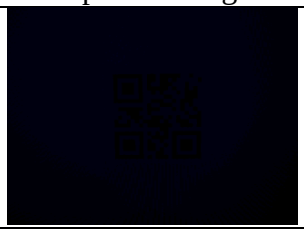
The QR code on the left contains “ABCDefgh” encoded in alphanumeric mode. The one on the right is the captured image using the camera. The scanner cannot decode the message because the image on the right is distorted it cannot see the QR code. As a result, there is no decoded message in the image.

QR # 16	Captured Image:
	
Encoded Message: Dario Ty	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

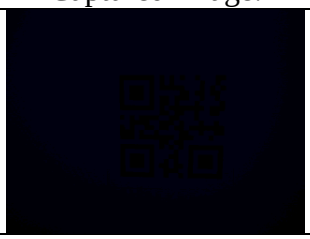
The QR code on the left is encoded with the word “Dario Ty” encoded in alphanumeric mode. The image in the right side is the captured image from the camera. The camera can’t identify the QR code in the captured image because it was dark. This test was a failure.

QR # 17	Captured Image:
	
Encoded Message: facebook.com/home	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

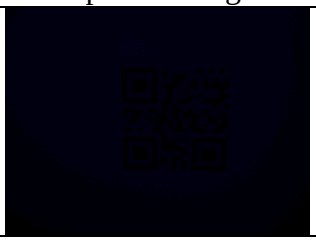
The QR code on the left is encoded with the word “Hi” encoded in facebook.com/home. The image on the right side of the table was scanned to decode a QR code. The camera doesn’t find any QR code in the image even though it was barely visible there. This test was a failure.

QR # 18	Captured Image:
	
Encoded Message: google.com.ph	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left is encoded with the word “google.com.ph” encoded in binary mode. The image on the right was scanned and tried to decode but the camera fails to do so. The QR code was barely visible from the captured image. This test was a failure as a result.

QR # 19	Captured Image:
	
Encoded Message: Hello Operator1	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

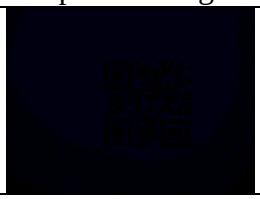
The QR code on the left contains “Hello Operator1” in alphanumeric encoding. This code is captured by the camera as seen in the right side of the table. This captured image cannot be decoded because the QR code cannot be detected clearly in the image on the right. The test was unsuccessful.

QR # 20	Captured Image:
	
Encoded Message: hello00910	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

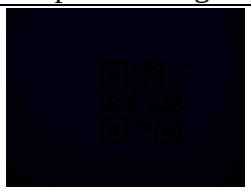
The QR code on the left contains “hello00910” in alphanumeric encoding. The image on the right side of the table is what the camera has processed. The QR code cannot be detected in the captured image because it was barely visible. There is no decoded message as a result of this test.

QR # 21	Captured Image:
	
Encoded Message: Mapua Institute	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

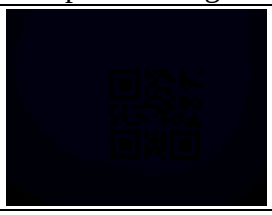
The QR code on the left side contains "Mapua Institute" encoded in numeric. The one on the right side is the captured image using the scanner. When the decoder was trying to detect the QR code, it failed to do so because the image was too dark to be decoded. As a result of this test, nothing was decoded in that image.

QR # 22	Captured Image:
	
Encoded Message: Mapua Intramuros	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

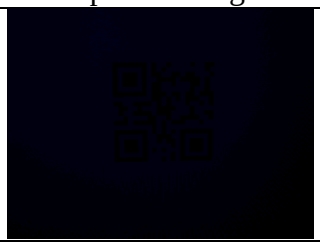
The QR code on the left contains the word "Mapua Intramuros" encoded in alphanumeric. In the right side is the captured image. The captured image was too dark and not very detailed. The scanner cannot properly detect the QR code on the captured image. As a result, this test was a failure.

QR # 23	Captured Image:
	
Encoded Message: Mark Domingo	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

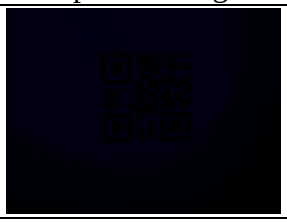
The QR code on the left contains “Mark Domingo” using binary encoding. The captured image is in the right side of the table. That image cannot be decoded because the scanner cannot detect the QR code properly. The QR code is still present but it is not entirely visible. As a result, the scanner returns no decoded message.

QR # 24	Captured Image:
	
Encoded Message: Patrick Jove	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

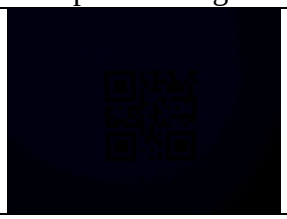
The QR code on the left is encoded with the word “Patrick Jove” encoded in alphanumeric mode. The decoded message from the camera which is in the right side of the table does not match the encoded QR code. This test was a failure because the captured image was too dark to be recognized by the camera.

QR # 25	Captured Image:
	
Encoded Message: qr@raco.com!	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

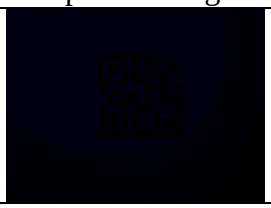
The QR code on the left is encoded with the word “qr@raco.com!” encoded in alphanumeric mode. The image in the right side is the captured image from the camera. The camera can’t identify the QR code in the captured image because it was dark and the QR code was difficult to find. This test was a failure.

QR # 26	Captured Image:
	
Encoded Message: readablecode#30	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left is encoded with the word “readablecode#30” encoded in alphanumeric mode. The image on the right was scanned and tried to decode but the camera fails to do so. The QR code was barely visible from the captured image. This test was a failure as a result.

QR # 27	Captured Image:
	
Encoded Message: sampleQRCODE	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left is encoded with the word “sampleQRCODE” encoded in alphanumeric mode. This QR code was scanned using the camera programmed with blue filters. The camera couldn’t identify the QR code in the captured image because it was barely recognizable. This test was a failure as a result.

QR # 28	Captured Image:
	
Encoded Message: !@#$%^&*()_+|	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left is encoded with the word "!@#$%^&*()_+|" encoded in alphanumeric mode. This QR code was scanned using the camera programmed with blue filters. The image on the right side is the captured image and as seen there, the image was dark and the QR code was unnoticeable. There is no returned message and this test was a failure.

QR # 29	Captured Image:
	
Encoded Message: Thesis Sample	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left is encoded with the word "Thesis Sample" encoded in alphanumeric mode. The decoded message from the camera which is in the right side of the table does not match the encoded QR code. This test was a failure because the captured image was too dark to be recognized by the camera.

QR # 30	Captured Image:
	
Encoded Message: wikipedia.org	Decoded Message: NONE
Encoding: Alphanumeric	
	Result: FAILED

The QR code on the left is encoded with the word "wikipedia.org" encoded in alphanumeric mode. The image in the right side is the captured image from the camera. The camera can't identify the QR code in the captured image because it was dark and distorted. This test was a failure.

$$SUCCESS\ RATE = \left[\frac{0}{30}\right] x100\% = 0\ \%$$

Table 3.7: Chi-Square Table for the Blue Filter:

	OBSERVED (O)	EXPECTED (E)	(O-E)	$(O-E)^2$	$\frac{(O-E)^2}{2}$
DETECTED	0	2	-2	4	2
NOT DETECTED	30	28	2	4	0.142857
TOTAL	30	30			**2.142857**

Based on the results, just like the test during the red filter, the QR code image scanned cannot be recognized by the decoder program of the microcontroller. It is observed that the scanned images are somehow distorted and also the QR codes are barely visible and makes the microcontroller having a hard time to recognize the whole QR code image. This gives the test a 0% success rate. The chi-square value is 2.142857. The degrees of freedom is 1 and the level of significance is 0.05. The value obtained lies between 2.07 and 3.84 or 2.07<P<3.84. As a result of this, we accept the null hypothesis for blue filters which is that at least 95% of the tested QR codes will not be successfully decoded

Based on the observations, the success rates of the RGB filters are 0 % for red, 100 % for green and 0 % for blue filter. Among the three filters that has been applied in the QR Code image, the Green filter was the one that was used in decoding the QR Code. It is the only image that is much detailed than the Red and Blue filters which can be described as distorted and dark.

Objective 3: Measure the success rate of decoding error free QR codes. Scan the generated QR code using the dedicated scanner. Compare the decoded message to the original encoded message. If the decoded message matches the encoded message, it is counted as a success. Record the time on table 3 and compute the success rate.

The QR code image on the left is encoded with QR code version 1, Level L error correction and various character encoding methods except Kanji. The scanned images from the camera are placed in the right side of the table. The green filter was used in running this test as the result of the image filters test previously. Each of the QR code image on the left were scanned using the hardware. The decoded message will be displayed after some time. The group extracted the scanned image from the hardware and placed it on the table below. The results are also placed along with it. To find the success rate of this test, use the equation:

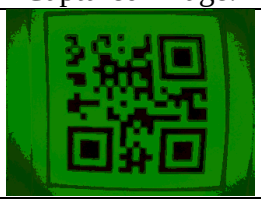
$$SUCCESS\ RATE = \left[\frac{N_{success}}{N_{total}}\right] x100\%$$

Where,

$N_{success}$ is the number of successfully decoded images

N_{total} is the total number of encoded images

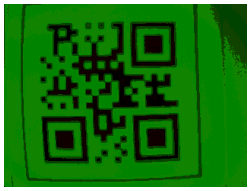
Table 3.8: Decoding of QR code images

QR # 1	Captured Image:
	
Encoded Message: Hello	Decoded Message: Hello
Encoding: Alphanumeric	
	Result: SUCCESS

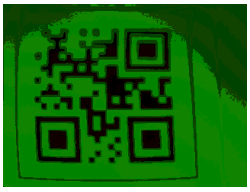
The QR code on the left of the table is encoded with the word “Hello”. The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The camera decoded the QR code properly and as a result, it returned the same message from the original.

QR # 2	Captured Image:
	
Encoded Message: Hello World	Decoded Message: Hello World
Encoding: Alphanumeric	
	Result: SUCCESS

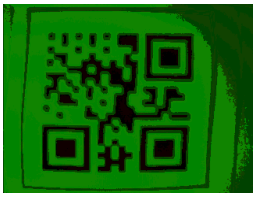
The QR code on the left is encoded with the word “Hello World”. The character encoding of the QR code is in alphanumeric mode. The decoded message from the camera matches the encoded QR code which is in the right side. This test was a success because the QR code captured was recognized by the decoder inside the camera.

QR # 3	Captured Image:
	
Encoded Message: Hi	Decoded Message: Hi
Encoding: Binary	
	Result: SUCCESS

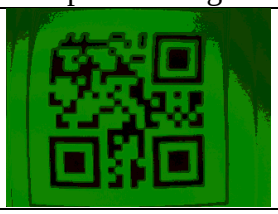
The QR code on the left of the table is encoded with the word “Hi” encoded in binary. The decoded message from the camera matches the encoded QR code. This test was a success because the captured image on the right of the table shows that the structure of the QR code is still clear and can be recognized and detected by the camera.

QR # 4	Captured Image:
	
Encoded Message: Hi World	Decoded Message: Hi World
Encoding: Binary	
	Result: SUCCESS

The QR code on the left is encoded with the word “Hi World” encoded in binary. The image on the right was the captured image. The QR code was clearly present in the captured image and as a result, the decoder returned the original message. The test was a success.

QR # 5	Captured Image:
	
Encoded Message: 12345	Decoded Message: 12345
Encoding: Numeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “12345” encoded in numeric. The captured image is in the right side. This was decoded by the camera. The decoded message from the camera matches the encoded QR code because the captured image shows a QR code properly.

QR # 6	Captured Image:
	
Encoded Message: 0123456789	Decoded Message: 0123456789
Encoding: Numeric	
	Result: SUCCESS

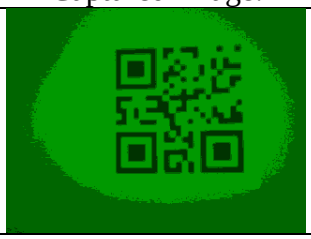
The QR code on the left side contains “0123456789” encoded in numeric. The QR code on the right side is the captured image using the camera. This captured image was decoded using the decoder inside the camera. The decoded message displayed the same as the original message. This was a success as a result of this.

QR # 7	Captured Image:
	
Encoded Message: ####Sample####	Decoded Message: ####Sample####
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word "####Sample####". The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The QR code is visible from the captured image. The camera decoded the QR code and it returned the exact message as the original. This test was a success.

QR # 8	Captured Image:
	
Encoded Message: @qrcode#sample%	Decoded Message: @qrcode#sample%
Encoding: Alphanumeric	
	Result: SUCCESS

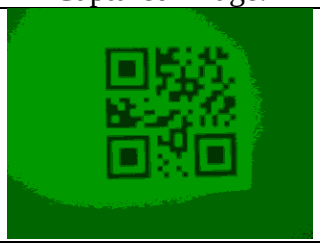
The QR code on the left is encoded with the word "@qrcode#sample%". The character encoding of the QR code is in alphanumeric mode. The QR code from the captured image was decoded by the camera. The camera returned the message the same as from the generated QR code. As a result, this test was a success.

QR # 9	Captured Image:
	
Encoded Message: ~Message Text!@	Decoded Message: ~Message Text!@
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word "~Message Text!@" encoded in alphanumeric. The captured image was scanned and decode the QR code from it. The camera returns the same message stored in the original. As a result of this test, it was recorded as a success.

QR # 10	Captured Image:
	
Encoded Message: 00012345000	Decoded Message: 00012345000
Encoding: Numeric	
	Result: SUCCESS

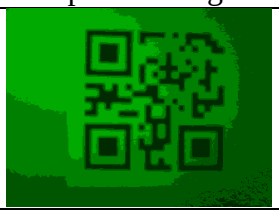
The QR code on the left is encoded with the word "00012345000" encoded in numeric. The image on the right was the captured image. The QR code was clearly present in the captured image and as a result, the decoder returned the original message. The test was a success.

QR # 11	Captured Image:
	
Encoded Message: 246813579	Decoded Message: 246813579
Encoding: Numeric	
	Result: SUCCESS

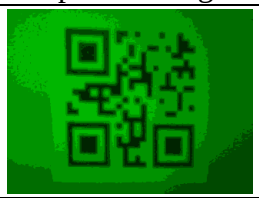
The QR code on the left of the table is encoded with the word "246813579" encoded in numeric. The captured image is in the right side. This was decoded by the camera. The decoded message from the camera matches the encoded QR code because the captured image shows a QR code properly.

QR # 12	Captured Image:
	
Encoded Message: 98124758392	Decoded Message: 98124758392
Encoding: Numeric	
	Result: SUCCESS

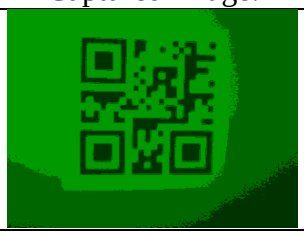
The QR code on the left side contains “98124758392” encoded in numeric. The QR code on the right side is the captured image using the camera. This captured image was decoded using the decoder inside the camera. The decoded message displayed the same as the original message. This was a success as a result of this.

QR # 13	Captured Image:
	
Encoded Message: 110101011101	Decoded Message: 110101011101
Encoding: Binary	
	Result: SUCCESS

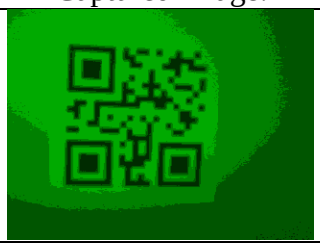
The QR code on the left of the table is encoded with the word “110101011101”. The character encoding of the QR code is in binary mode. The image on the right is the captured QR code. The camera decoded the QR code properly and as a result, it returned the same message from the original.

QR # 14	Captured Image:
	
Encoded Message: 000010101001110010101O	Decoded Message: 000010101001110010101O
Encoding: Binary	
	Result: SUCCESS

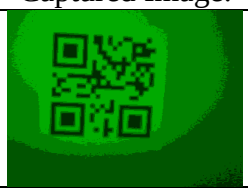
The QR code on the left is encoded with the word “00001010100111001010 10”. The character encoding of the QR code is in binary mode. The decoded message from the camera matches the encoded QR code which is in the right side. This test was a success because the QR code captured was recognized by the decoder inside the camera.

QR # 15	Captured Image:
	
Encoded Message: ABCDefgh	Decoded Message: ABCDefgh
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “ABCDefgh” encoded in alphanumeric. The decoded message from the camera matches the encoded QR code. This test was a success because the captured image on the right of the table shows that the structure of the QR code is still clear and can be recognized and detected by the camera.

QR # 16	Captured Image:
	
Encoded Message: Dario Ty	Decoded Message: Dario Ty
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left is encoded with the word “Dario Ty” encoded alphanumeric. The image on the right was the captured image. The QR code was clearly present in the captured image and as a result, the decoder returned the original message. The test was a success.

QR # 17	Captured Image:
	
Encoded Message: facebook.com/home	Decoded Message: facebook.com/home
Encoding: Alphanumeric	
	Result: SUCCESS

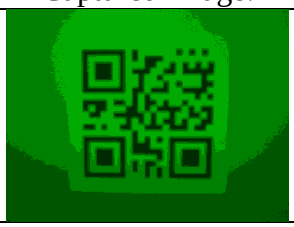
The QR code on the left of the table is encoded with the word facebook.com/home encoded in alphanumeric. The captured image was scanned and decode the QR code from it. The camera returns the same message stored in the original. As a result of this test, it was recorded as a success.

QR # 18	Captured Image:
	
Encoded Message: google.com.ph	Decoded Message: google.com.ph
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left is encoded with the word “google.com.ph”. The character encoding of the QR code is in alphanumeric mode. The QR code from the captured image was decoded by the camera. The camera returned the message the same as from the generated QR code. As a result, this test was a success.

QR # 19	Captured Image:
	
Encoded Message: Hello Operator1	Decoded Message: Hello Operator1
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “Hello Operator1”. The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The QR code is visible from the captured image. The camera decoded the QR code and it returned the exact message as the original. This test was a success.

QR # 20	Captured Image:
	
Encoded Message: hello00910	Decoded Message: hello00910
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left side contains “hello00910” encoded in alphanumeric. The QR code on the right side is the captured image using the camera. This captured image was decoded using the decoder inside the camera. The decoded message displayed the same as the original message. This was a success as a result of this.

QR # 21	Captured Image:
	
Encoded Message: Mapua Institute	Decoded Message: Mapua Institute
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “Mapua Institute” encoded in alphanumeric. The captured image is in the right side. This was decoded by the camera. The decoded message from the camera matches the encoded QR code because the captured image shows a QR code properly.

QR # 22	Captured Image:
	
Encoded Message: Mapua Intramuros	Decoded Message: Mapua Intramuros
Encoding: Alphanumeric	
	Result: SUCCESS

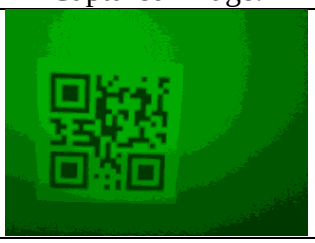
The QR code on the left of the table is encoded with the word “Mapua Intramuros”. The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The camera decoded the QR code properly and as a result, it returned the same message from the original.

QR # 23	Captured Image:
	
Encoded Message: Mark Domingo	Decoded Message: Mark Domingo
Encoding: Alphanumeric	
	Result: SUCCESS

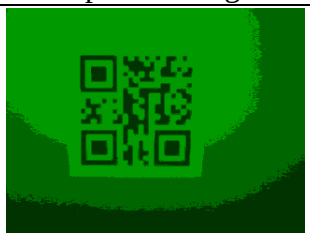
The QR code on the left is encoded with the word “Mark Domingo”. The character encoding of the QR code is in alphanumeric mode. The decoded message from the camera matches the encoded QR code which is in the right side. This test was a success because the QR code captured was recognized by the decoder inside the camera.

QR # 24	Captured Image:
	
Encoded Message: Patrick Jove	Decoded Message: Patrick Jove
Encoding: Alphanumeric	
	Result: SUCCESS

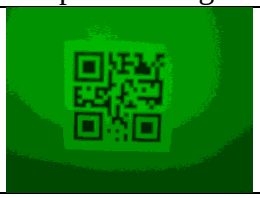
The QR code on the left of the table is encoded with the word “Patrick Jove” encoded in alphanumeric. The decoded message from the camera matches the encoded QR code. This test was a success because the captured image on the right of the table shows that the structure of the QR code is still clear and can be recognized and detected by the camera.

QR # 25	Captured Image:
	
Encoded Message: qr@raco.com!	Decoded Message: qr@raco.com!
Encoding: Alphanumeric	
	Result: SUCCESS

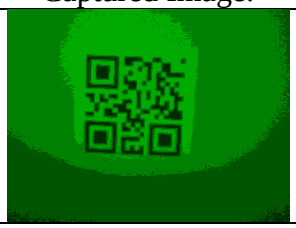
The QR code on the left side contains “qr@raco.com!” encoded in alphanumeric. The QR code on the right side is the captured image using the camera. This captured image was decoded using the decoder inside the camera. The decoded message displayed the same as the original message. This was a success as a result of this.

QR # 26	Captured Image:
	
Encoded Message: readablecode#30	Decoded Message: readablecode#30
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “readablecode#30”. The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The QR code is visible from the captured image. The camera decoded the QR code and it returned the exact message as the original. This test was a success.

QR # 27	Captured Image:
	
Encoded Message: sampleQRCODE	Decoded Message: sampleQRCODE
Encoding: Alphanumeric	
	Result: SUCCESS

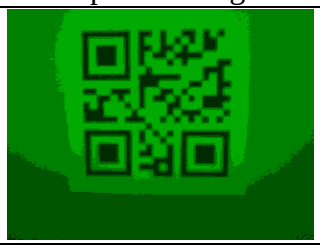
The QR code on the left of the table is encoded with the word "sampleQRCODE".encoded in alphanumeric. The captured image was scanned and decode the QR code from it. The camera returns the same message stored in the original. As a result of this test, it was recorded as a success.

QR # 28	Captured Image:
	
Encoded Message: !@#$%^&*()_+|	Decoded Message: !@#$%^&*()_+|
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left is encoded with the word "!@#$%^&*()_+|". The character encoding of the QR code is in alphanumeric mode. The QR code from the captured image was decoded by the camera. The camera returned the message the same as from the generated QR code. As a result, this test was a success.

QR # 29	Captured Image:
	
Encoded Message: Thesis Sample	Decoded Message: Thesis Sample
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “Thesis Sample”. The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The QR code is visible from the captured image. The camera decoded the QR code and it returned the exact message as the original. This test was a success.

QR # 30	Captured Image:
	
Encoded Message: wikipedia.org	Decoded Message: wikipedia.org
Encoding: Alphanumeric	
	Result: SUCCESS

The QR code on the left of the table is encoded with the word “wikipedia.org”. The character encoding of the QR code is in alphanumeric mode. The image on the right is the captured QR code. The camera decoded the QR code properly and as a result, it returned the same message from the original.

$$SUCCESS\ RATE = \left[\frac{30}{30}\right] x100\% = 100\ \%$$

Table 3.9: Chi-Square Table for the QR code images:

	OBSERVED (O)	EXPECTED (E)	(O-E)	$(O-E)^2$	$\frac{(O-E)^2}{2}$
DECODED	30	28	2	4	0.142857
NOT DECODED	0	2	-2	4	2
TOTAL	30	30			**2.142857**

Based on the results, 30 out of 30 of the QR code images scanned in the table can be recognized by the decoder program of the hardware. The important features of the QR code such as the finder pattern, format information, and especially the data and error correction are very clear in the scanned image and makes the hardware decode the message. The time was

also recorded for use in the next tests. With these results, the test has achieved a 100% success rate. The chi-square value is 2.142857. The degrees of freedom is 1 and the level of significance is 0.05. The value obtained lies between 2.07 and 3.84 or 2.07<P<3.84. As a result of this, we accept the null hypothesis for unmodified QR codes which is that at least 95% of the tested QR codes will be successfully decoded

Objective 4: Measure the success rate of decoding QR codes introduced with errors in different attack scenarios. Intentionally add errors on the generated QR codes on part 1. Scan it again using the dedicated scanner. Compare the results from the original encoded data and record the time on table 3 and compute the success rate.

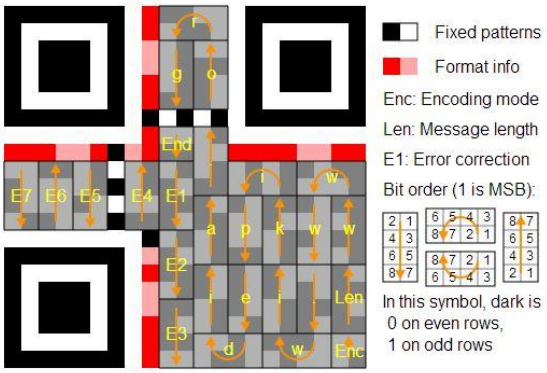


Figure 3.11: QR Code structure

The figure shows the QR code structure, in this test the areas that should be modified. The format info contains the mask pattern (mask area) and the error correction level. The gray area is where the data and the error correction are placed. In the bottom right corner of the QR code, the character encoding is labeled with 'Enc'. Above this is the character length which is labeled with 'Len'.

The QR code image on the left of the table below is encoded with QR code version 1, Level L error correction and various character encoding methods except Kanji. The scanned images from the camera are placed in the right side of the table. The green filter was used in

running this test as the result of the image filters test previously. The test starts by generating QR codes given in the website. The group then modify the QR code image depending on the attack area given in the table. The group chose to modify the Mask, Character Encoding, Character Area and the Data and Error Correction area of the QR Code. The group uses an image manipulation program in a computer. After modifying, the group scanned the image, the result will appear in the display for some time. Extract the QR code from the hardware and place it in the rightmost column of the table along with the displayed results. To find the success rate of this test, use the equation:

$$SUCCESS\ RATE = \left[\frac{N_{success}}{N_{total}}\right] x100\%$$

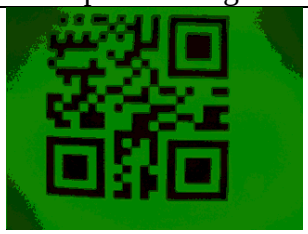
Where,

$N_{success}$ is the number of successfully decoded images

N_{total} is the total number of encoded images

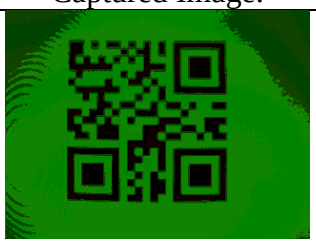
Table 3.10: Decoding of modified QR code images

The QR codes was encoded with the message "Hello World". QR codes were modified depending on the attack area and labeled to identify the changes made in the original.

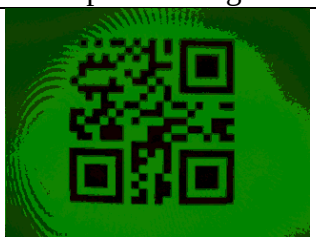
QR # 31	Modified QR:	Captured Image:
		
Encoded Message: Hello World	Attack Area: Mask Pattern	Decoded Message: Hello World
		Result: SUCCESS

The QR code on the left contains "Hello World" which is generated from the QR code generator web application. In the center is the modified QR code. The red boxes in the center

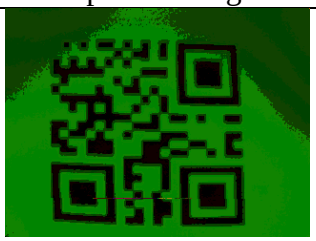
image is where the mask area is located. In comparison with the left image, the mask area is filled with an additional black pixels each side using an image manipulation program. This changes the mask pattern of the whole QR code. When the group scanned the modified QR code, the results shows “Hello World”, this is the same as the content of the original QR code image. This test was counted as a success.

QR # 32	Modified QR:	Captured Image:
		
Encoded Message: Hello World	Attack Area: Char. Encoding	Decoded Message: Hello World
		Result: SUCCESS

Same as the first test, the QR code on the left contains “Hello World”. In the center QR code image, the red box on the lower right corner means that it is the character encoding area, the area to be modified. Compared with the original image, the character encoding area is fully filled with pixels. This changed the character encoding type of the QR code from alphanumeric to invalid. The group scanned the modified QR code image and the results shown is “Hello World”, same as the original code. The contents are unaffected regardless of the modification. This test was counted as a success.

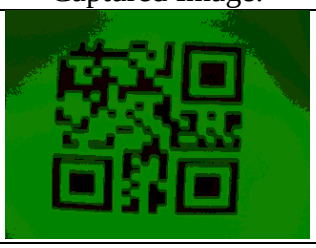
QR # 33	Modified QR:	Captured Image:
		
Encoded Message: Hello World	Attack Area: Char. Count	Decoded Message: Hello World
		Result: SUCCESS

The QR code used in this test is contains “Hello World”. The red box in the lower left of the QR code image in the center is the character count area. The original image was modified using an image manipulation program. The modified QR code in the center changed its character count by removing the pixels in that area. This changes the character count from a different one.The groupp scanned the center image and got a matching result: “Hello World”. The contents of the code are unaffected and the test was counted as a success.

QR # 34	Modified QR:	Captured Image:
		
Encoded Message: Hello World	Attack Area: Data & Error Correction	Decoded Message: NONE
		Result: FAILED

The”Hello World” QR code in this test was used. In the center image is the modified QR code. The QR code was modified using the same techniques used in the first 3 tests. There are four red boxes in the center image. These are the areas of attack in this test. In boxes # 1, 2, and 4, the pixels was removed thus modified the error correction area of the QR code. In box # 4, the pixels was also removed. This time, the data of the QR code was modified. The group scanned the modified QR code and didn’t get any results. The contents of the modified QR code cannot be recovered anymore because of the modification, making this test marked as a failure.

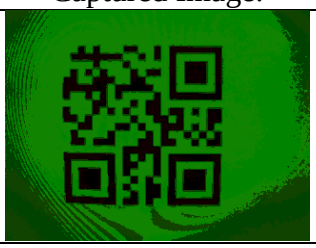
The QR codes was encoded with the message “0123456789”. QR codes were modified depending on the attack area and labeled to identify the changes made in the original.

QR # 35	Modified QR:	Captured Image:
		
Encoded Message: 0123456789	Attack Area: Mask Pattern	Decoded Message: 0123456789
		Result: SUCCESS

In this test, the QR code on the left contains, “0123456789”. Same as the test in QR code # 7, the mask area was modified. The red boxes in the center image was the mask pattern area. Pixels were added in the mask area using an image manipulation program making the mask pattern of the QR code change. The modified QR code was scanned and the result matches with the original QR code, “0123456789”. This test was registered as a success.

QR # 36	Modified QR:	Captured Image:
		
Encoded Message: 0123456789	Attack Area: Char. Encoding	Decoded Message: 0123456789
		Result: SUCCESS

Same as QR code # 11, the QR code image on the left contains “0123456789”. This time, the character encoding area was modified. As seen in the center image, the red box in the image was filled with black pixels making the encoding change from numeric mode to an invalid one. When the group scanned the center image, the result is still the same as the original QR code image, “0123456789”. The contents were unaffected and makes this QR test counted as a success.

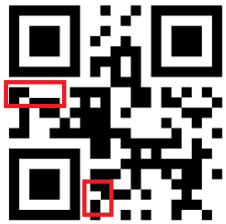
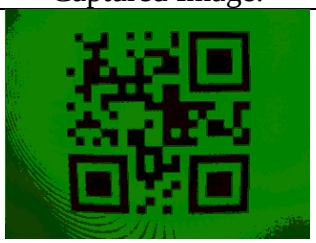
QR # 37	Modified QR:	Captured Image:
		
Encoded Message: 0123456789	Attack Area: Char. Count	Decoded Message: 0123456789
		Result: SUCCESS

The QR code in the left side still contains “0123456789”. In the center image is the modified QR code. The red box is where the character count area is. The original image was modified by removing the pixels in the character count area. Compared with the original, this changes the character count of the whole QR code. The group scanned the modified QR code image and the scanned image still shows the encoded QR code, “0123456789”. This makes the test counted as a success.

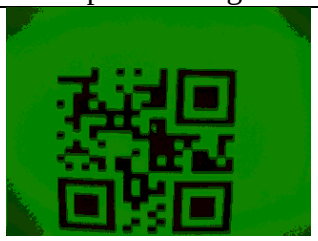
QR # 38	Modified QR:	Captured Image:
		
Encoded Message: 0123456789	Attack Area: Data & Error Correction	Decoded Message: NONE
		Result: FAILED

This is the last test for the QR code that contains “0123456789”. In the center of the table is the modified QR code. There are six red boxes where the pixels was modified. Compared with the original image, boxes # 1 and 4 was added with a single black pixel.This changed the error correction area of the QR code. In boxes # 2 and 5, the pixels was removed in it and in boxes # 3 and 6, the additional pixels were added changing the data of the QR code. The test was a failure because the QR code image in the center was heavily modified and makes the content of it unrecognizable.

The QR codes was encoded with the message “Hi World”. QR codes were modified depending on the attack area and labeled to identify the changes made in the original.

QR # 39	Modified QR:	Captured Image:
		
Encoded Message: Hi World	Attack Area: Mask Pattern	Decoded Message: Hi World
		Result: SUCCESS

The QR code in the left contains “Hi World”. This QR code is also used in the next 3 tests. The mask area is the target of modification as seen in the center image of the table. The red boxes are where the modification was done. Observe that the pixels was removed in the QR code thus making the mask pattern different than the original. The modified QR code was scanned and the results shows that the content of the QR code retained. This test was counted as a success.

QR # 40	Modified QR:	Captured Image:
		
Encoded Message: Hi World	Attack Area: Char. Encoding	Decoded Message: Hi World
		Result: SUCCESS

In this test, the QR code “Hi World” was modified. The character encoding area was modified this time. The red box in the center image is where the modification was done and this is also where the character encoding area is. The orignal pixel was removed and added some pixels above it. This makes character encoding of the QR code was changed from

Alphanumeric mode to a structured append mode. The modified QR code was scanned and the modified QR code still contains “Hi World” regardless of the modifications done in the image. The test was counted as a success.

QR # 41	Modified QR:	Captured Image:
		
Encoded Message: Hi World	Attack Area: Char. Count	Decoded Message: Hi World
		Result: SUCCESS

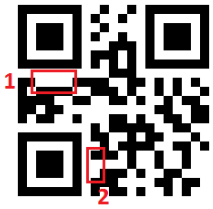
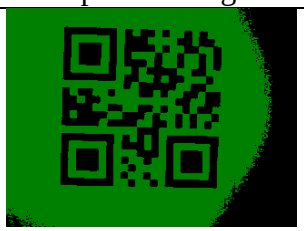
QR code # 17 contains the word “Hi World”. This time, the character count was modified in this test. Observe the red box in the middle image above, this is where the character count is located. Compared with the original image on the left, the pixels were removed from the QR code making the character count changed to a different one. The modified QR code was scanned and the contents are still the same as the original, “Hi World”. The test was counted as a success.

QR # 42	Modified QR:	Captured Image:
		
Encoded Message: Hi World	Attack Area: Data & Error Correction	Decoded Message: NONE
		Result: FAILED

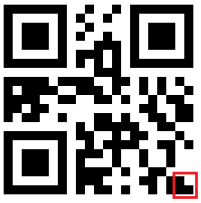
The last QR code on the left still contains “Hi World”. The data and error correction area of this original QR code was modified as seen in the center image. There are 3 boxes in the center image where the modification was done. In box # 2, the error correction was

modified by removing a single pixel and in boxes # 1 and 3, the data was modified by removing the pixels. Compared with the original image, the QR code in the middle was heavily modified. This was reflected in the results of the scan and makes the test unsuccessful. This is the third unsuccessful run.

The QR codes was encoded with the message “246813579”. QR codes were modified depending on the attack area and labeled to identify the changes made in the original.

QR # 43	Modified QR:	Captured Image:
		
Encoded Message: 246813579	Attack Area: Mask Pattern	Decoded Message: 246813579
		Result: SUCCESS

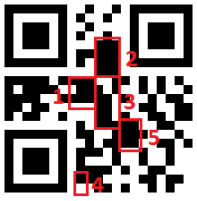
The red boxes in the center image is where the mask area is located. In comparison with the left image, some pixels were removedeach side using an image manipulation program. This changes the mask pattern of the whole QR code. When the camera scanes and decodes the modified QR code, the results shows the same as the content of the original QR code image. This test was counted as a success.

QR # 44	Modified QR:	Captured Image:
		
Encoded Message: 246813579	Attack Area: Char. Encoding	Decoded Message: 246813579
		Result: SUCCESS

The red box in the center image is where the modification was done and this is also where the character encoding area is. The character encoding of the QR code was changed from numeric mode to an unidentified or invalid one. The modified QR code was scanned and the modified QR code still contains “246813579” regardless of the modifications done in the image. The test was counted as a success.

QR # 45	Modified QR:	Captured Image:
		
Encoded Message: 246813579	Attack Area: Char. Count	Decoded Message: 246813579
		Result: SUCCESS

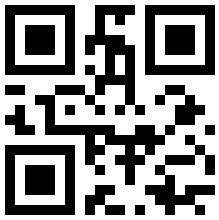
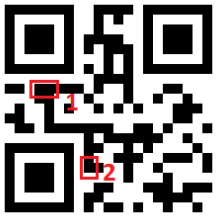
The red box is where the character count area is. The original image was modified by removing the pixels in the character count area. Compared with the original, this changes the character count of the whole QR code. The camera scanned for the modified QR code image and the image still shows the encoded QR code, “246813579”. This makes the test counted as a success.

QR # 46	Modified QR:	Captured Image:
		
Encoded Message: 246813579	Attack Area: Data & Error Correction	Decoded Message: NONE
		Result: FAILED

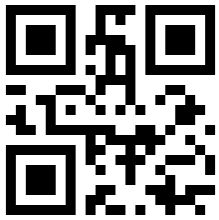
Boxes labeled 4 was the error correction area. The error correction area of the QR code changed from level Q to level L. Boxes labeled 1 to 3 and 5 are the data area to be modified.

The test was a failure because the QR code image in the center was heavily modified and makes the content of it unrecognizable.

The QR codes was encoded with the message “Dario Ty”. QR codes were modified depending on the attack area and labeled to identify the changes made in the original.

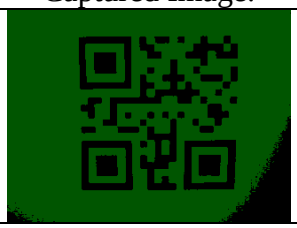
QR # 47	Modified QR:	Captured Image:
		
Encoded Message: Dario Ty	Attack Area: Mask Pattern	Decoded Message: Dario Ty
		Result: SUCCESS

The red boxes labeled 1 and 2 in the center image was the mask pattern area. Pixels were added in the mask area using an image manipulation program making the mask pattern of the QR code change. The modified QR code was scanned and the result matches with the original QR code. This test was registered as a success.

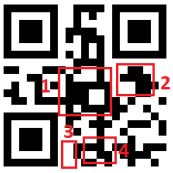
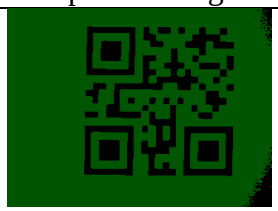
QR # 48	Modified QR:	Captured Image:
		
Encoded Message: Dario Ty	Attack Area: Char. Encoding	Decoded Message: Dario Ty
		Result: SUCCESS

The character encoding area was modified. The red box in the center image is where the character encoding area is. The character encoding of the QR code was changed to an unidentified or an invalid one. The modified QR code was scanned and the modified QR code

still contains the same message fom the original regardless of the modifications done in the image. The test was counted as a success.

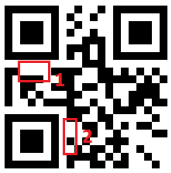
QR # 49	Modified QR:	Captured Image:
		
Encoded Message: Dario Ty	Attack Area: Char. Count	Decoded Message: Dario Ty
		Result: SUCCESS

The red box is where the character count area is. The original image was modified by removing the pixels in the character count area. Compared with the original, this changes the character count of the whole QR code. The camera scanned for the modified QR code image and the image still shows the encoded QR code, “Dario Ty”. This makes the test counted as a success.

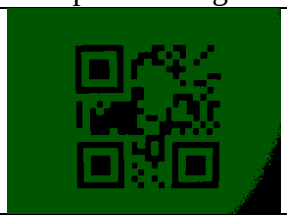
QR # 50	Modified QR:	Captured Image:
		
Encoded Message: Dario Ty	Attack Area: Data & Error Correction	Decoded Message: NONE
		Result: FAILED

The red boxes labeled 1, 2 and 4 are the data area and the red box labeled 3 are the error correction area. The error correction level was changed to M. The decoder scans the QR code and decoded it. There is no returned message to the display. The data is damaged from the modification. As a result, this test was a failure.

The QR codes was encoded with the message “Mark Domingo”. QR codes were modified depending on the attack area and labeled to identify the changes made in the original.

QR # 51	Modified QR:	Captured Image:
		
Encoded Message: Mark Domingo	Attack Area: Mask Pattern	Decoded Message: Mark Domingo
		Result: SUCCESS

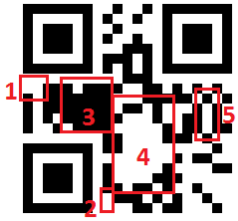
The boxes marked 1 and 2 are where the mask pattern is located. In comparison with the original, the mask pattern was filled with additional pixels that changes the mask pattern completely. The decoder scans the QR code and decoded it. The message was still the same as the original and as a result, this was a success.

QR # 52	Modified QR:	Captured Image:
		
Encoded Message: Mark Domingo	Attack Area: Char. Encoding	Decoded Message: Mark Domingo
		Result: SUCCESS

The red box in the center image is where the modification was done and this is also where the character encoding area is. The character encoding of the QR code was changed from numeric mode to an unidentified or invalid one. The modified QR code was scanned and the modified QR code still contains “Mark Domingo” regardless of the modifications done in the image. The test was counted as a success.

QR # 53	Modified QR:	Captured Image:
		
Encoded Message: Mark Domingo	Attack Area: Char. Count	Decoded Message: Mark Domingo
		Result: SUCCESS

The character count was changed by adding additional pixels from that area. With that, the character count changes completely. The decoder scans the QR code and decoded it. The message was still the same as the original and as a result, this was a success.

QR # 54	Modified QR:	Captured Image:
		
Encoded Message: Mark Domingo	Attack Area: Data & Error Correction	Decoded Message: NONE
		Result: FAILED

Boxes labeled 1 and 2 was the error correction area. The error correction area of the QR code changed from level Q to level M. Boxes labeled 3 to 5 are the data area to be modified. The test was a failure because the QR code image in the center was heavily modified and makes the content of it unrecognizable.

The QR codes was encoded with the message “Patrick Jove”. QR codes were modified depending on the attack area and labeled to identify the changes made in the original.

QR # 55	Modified QR:	Captured Image:
		
Encoded Message: Patrick Jove	Attack Area: Mask Pattern	Decoded Message: Patrick Jove
		Result: SUCCESS

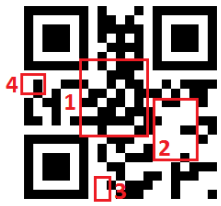
The boxes marked 1 and 2 are where the mask pattern is located. In comparison with the original, the mask pattern was filled with additional pixels that changes the mask pattern completely. The decoder scans the QR code and decoded it. The message was still the same as the original and as a result, this was a success.

QR # 56	Modified QR:	Captured Image:
		
Encoded Message: Patrick Jove	Attack Area: Char. Encoding	Decoded Message: Patrick Jove
		Result: SUCCESS

The character encoding area, marked by a red box, was filled with additional pixels that changes the encoding mode to an invalid one. The decoder scans the QR code and decoded it. The message was still the same as the original and as a result, this was a success.

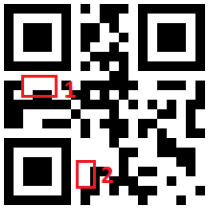
QR # 57	Modified QR:	Captured Image:
		
Encoded Message: Patrick Jove	Attack Area: Char. Count	Decoded Message: Patrick Jove
		Result: SUCCESS

The character count was changed by adding additional pixels from that area. With that, the character count changes completely. The decoder scans the QR code and decoded it. The message was still the same as the original and as a result, this was a success.

QR # 58	Modified QR:	Captured Image:
		
Encoded Message: Patrick Jove	Attack Area: Data & Error Correction	Decoded Message: NONE
		Result: FAILED

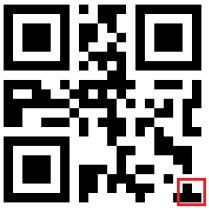
The red boxes labeled 1 and 2 are the data area and the red boxes labeled 3 and 4 are the error correction area. The error correction level was changed to M in box 3 and level L in box 4. The decoder scans the QR code and decoded it. There is no returned message to the display. The data is damaged from the modification. As a result, this test was a failure.

The QR codes was encoded with the message "Thesis Sample". QR codes were modified depending on the attack area and labeled to identify the changes made in the original.

QR # 59	Modified QR:	Captured Image:
		
Encoded Message: Thesis Sample	Attack Area: Mask Pattern	Decoded Message: Thesis Sample
		Result: SUCCESS

The red boxes labeled 1 and 2 are the mask pattern area. Pixels was removed from the original QR code that completely changes the mask pattern of it. The decoder scans the QR

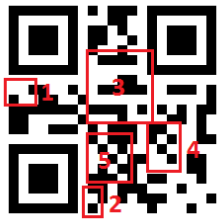
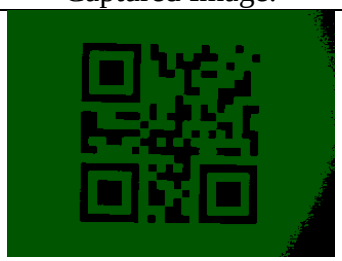
code and decoded it. The message was still the same as the original and as a result, this was a success.

QR # 60	Modified QR:	Captured Image:
		
Encoded Message: Thesis Sample	Attack Area: Char. Encoding	Decoded Message: Thesis Sample
		Result: SUCCESS

The character encoding area, marked by a red box, was filled with additional pixels that changes the encoding mode from alphanumeric mode to an invalid one. The decoder scans the QR code and decoded it. The message was still the same as the original and as a result, this was a success

QR # 61	Modified QR:	Captured Image:
		
Encoded Message: Thesis Sample	Attack Area: Char. Count	Decoded Message: Thesis Sample
		Result: SUCCESS

The character count was changed by adding additional pixels from that area. With that, the character count changes completely. The decoder scans the QR code and decoded it. The message was still the same as the original and as a result, this was a success.

QR # 62	Modified QR:	Captured Image:
		
Encoded Message: Thesis Sample	Attack Area: Data & Error Correction	Decoded Message: NONE
		Result: FAILED

The red boxes labeled 1 and 2 are the error correction area and the red boxes labeled 3 to 5 are the data area. The error correction level was changed to M in box 1 and level H in box 4. The decoder scans the QR code and decoded it. There is no returned message to the display. The data is damaged from the modification. As a result, this test was a failure.

$$SUCCESS\ RATE = \left[\frac{24}{32}\right] x100\% = 75\ \%$$

Table 3.11: Chi-Square Table for the modified QR code images:

	OBSERVED (O)	EXPECTED (E)	(O-E)	$(O-E)^2$	$\frac{(O-E)^2}{2}$
DECODED	24	30	-6	36	1.2
NOT DECODED	8	2	6	36	18
TOTAL	32	32			**19.2**

Based on the results, the decoder program can still be recognized a modified QR code except those with altered data and error correction areas. The data and error correction area as explained in the previous chapters, are the largest part of the QR code. If someone modifies a data or the error correction area in the QR code, the damage is significant and the data cannot be recovered no matter what. Therefore, as long as these areas are still intact or has minimal damage, the QR code can still be recognized. This gives the test a 75% success rate as expected. The chi-square value is 19.2. The degrees of freedom used is 1 and the level of significance used is 0.05. Based on the results, the chi-square value is larger than 3.84 or P>3.84. The null

hypothesis is rejected and the alternative hypothesis is accepted which is at least 5% of the test in modified QR codes will pass.

Objective 2: Measure the total time it takes to capture, process, decode, and display the result of the scanned QR code with respect to varying character length and data type encoding. Observe and record the delay of decoding a QR code in different scenarios using the decoding time inside the program and a stopwatch.

The group recorded the time in the previous tests in tables 3.5 and 3.6. The total time is the sum of the scanning time and the decoding time. There are total of 18 trials in this test. The equation below is to get the scanning time by getting the difference between the total time and the decoding time:

$$Scanning\ time = Total\ Time - Decoding\ Time$$

To get the arithmetic mean for each of decoding, scanning and the total time, use the equation:

$$Ave_{TIME} = \frac{1}{n}\sum_{i=1}^{n} a_i$$

Where,

Ave_{time} is the average time of decoding, scanning or the total time.

n is the total number of the QR codes tested

a_i is the decoding, scanning or total time recorder in each QR code.

Table 3.12: Measurement of Time

QR #	**Message**	**Encoding**	**Type of Error**	**Decoding Time**	**Scanning Time**	**Total Time (in sec)**
1	Hello	Alphanumeric	None	0.39 s	40.00 s	40.39
2	Hello World	Alphanumeric	None	0.38 s	40.07 s	40.46
3	Hi	Binary	None	0.39 s	39.88 s	40.27

4	Hi World	Binary	None	0.39 s	39.95 s	40.34
5	12345	Numeric	None	0.41 s	39.99 s	40.40
6	0123456789	Numeric	None	0.39 s	39.73 s	40.12
7	####Sample####	Alphanumeric	None	0.41 s	39.95 s	40.36
8	@qrcode#sample%	Alphanumeric	None	0.40 s	40.10 s	40.50
9	~Message Text!@	Alphanumeric	None	0.40 s	40.06 s	40.46
10	00012345000	Numeric	None	0.38 s	39.90 s	40.28
11	246813579	Numeric	None	0.40 s	39.94 s	40.34
12	98124758392	Numeric	None	0.38 s	39.99 s	40.37
13	110101011101	Binary	None	0.37 s	39.97 s	40.34
14	0000101010011100101010	Binary	None	0.38 s	40.03 s	40.41
15	ABCDefgh	Alphanumeric	None	0.39 s	40.00 s	40.39
16	Dario Ty	Alphanumeric	None	0.38 s	40.05 s	40.43
17	facebook.com/home	Alphanumeric	None	0.39 s	40.08 s	40.47
18	google.com.ph	Alphanumeric	None	0.39 s	39.96 s	40.35
19	Hello Operator1	Alphanumeric	None	0.40 s	39.92 s	40.32
20	hello00910	Alphanumeric	None	0.38 s	40.02 s	40.40
21	Mapua Institute	Alphanumeric	None	0.39 s	39.93 s	40.32
22	Mapua Intramuros	Alphanumeric	None	0.41 s	40.02 s	40.43
23	Mark Domingo	Alphanumeric	None	0.39 s	39.96 s	40.35
24	Patrick Jove	Alphanumeric	None	0.40 s	40.01 s	40.41
25	qr@raco.com!	Alphanumeric	None	0.40 s	40.10 s	40.50
26	readablecode#30	Alphanumeric	None	0.40 s	40.06 s	40.46
27	sampleQRCODE	Alphanumeric	None	0.39 s	39.96 s	40.35
28	!@#$%^&*()_+|	Alphanumeric	None	0.41 s	40.13 s	40.54
29	Thesis Sample	Alphanumeric	None	0.40 s	39.91 s	40.31
30	wikipedia.org	Alphanumeric	None	0.39 s	39.99 s	40.38
31	Hello World	Alphanumeric	Mask	0.38 s	40.29 s	40.67
32	Hello World	Alphanumeric	Char Enc	0.38 s	40.00 s	40.38
33	Hello World	Alphanumeric	CharCount	0.39 s	40.18 s	40.57
34	Hello World	Alphanumeric	Data+E.C	0.39 s	40.15 s	40.54
35	0123456789	Numeric	Mask	0.38 s	40.18 s	40.56
36	0123456789	Numeric	Char Enc	0.39 s	40.09 s	40.48
37	0123456789	Numeric	CharCount	0.42 s	40.08 s	40.50
38	0123456789	Numeric	Data+E.C	0.38 s	40.19 s	40.57
39	Hi World	Binary	Mask	0.39 s	40.28 s	40.67
40	Hi World	Binary	Char Enc	0.38 s	40.47 s	40.85
41	Hi World	Binary	CharCount	0.42 s	40.21 s	40.63
42	Hi World	Binary	Data+E.C	0.38 s	39.74 s	40.12
43	246813579	Numeric	Mask	0.38 s	40.17 s	40.55
44	246813579	Numeric	Char Enc	0.39 s	40.09 s	40.48
45	246813579	Numeric	CharCount	0.41 s	40.13 s	40.54
46	246813579	Numeric	Data+E.C	0.39 s	40.13 s	40.52

47	Dario Ty	Alphanumeric	Mask	0.39 s	40.15 s	40.54
48	Dario Ty	Alphanumeric	Char Enc	0.39 s	40.09 s	40.48
49	Dario Ty	Alphanumeric	CharCount	0.40 s	40.08 s	40.48
50	Dario Ty	Alphanumeric	Data+E.C	0.38 s	40.17 s	40.55
51	Mark Domingo	Alphanumeric	Mask	0.39 s	40.12 s	40.51
52	Mark Domingo	Alphanumeric	Char Enc	0.38 s	40.19 s	40.57
53	Mark Domingo	Alphanumeric	CharCount	0.42 s	40.06 s	40.48
54	Mark Domingo	Alphanumeric	Data+E.C	0.39 s	40.21 s	40.60
55	Patrick Jove	Alphanumeric	Mask	0.38 s	40.16 s	40.54
56	Patrick Jove	Alphanumeric	Char Enc	0.38 s	40.25 s	40.63
57	Patrick Jove	Alphanumeric	CharCount	0.42 s	40.17 s	40.59
58	Patrick Jove	Alphanumeric	Data+E.C	0.38 s	40.22 s	40.60
59	Thesis Sample	Alphanumeric	Mask	0.38 s	40.16 s	40.54
60	Thesis Sample	Alphanumeric	Char Enc	0.39 s	40.17 s	40.56
61	Thesis Sample	Alphanumeric	CharCount	0.41 s	39.99 s	40.40
62	Thesis Sample	Alphanumeric	Data+E.C	0.38 s	40.07 s	40.45

The leftmost column is the QR code number and the number are indicated in each generated QR code images in the previous tests in tables 3.5 and 3.6. The second column is the original message encoded in the QR code of the respective number. The third and fourth column is the configuration of the QR code of the respective number such as the encoding and the type of attack used in the QR code. The decoding time indicates how long does the QR code can be decoded by the program. This is displayed on the results from the terminal only. The scanning time indicates how long the hardware and the software find the QR code and the total time is the sum of both the decoding time and the scanning time. This was measured by the stopwatch. The scanning time was taken from the equation given previously. After all data was recorded, the group computes the average time of decoding, scanning and the total time.

Based on the computed results, the average decoding time was 0.3919 seconds. The average scanning time is 40.0718 seconds. Lastly, the total time is 40.4613 seconds. The count was started after pressing the trigger of the camera and ended when the results are displayed. This is where the total time was recorded. The decoding time is shown in the results displayed

on the terminal. With the data that has been acquired, the time differs depending on the quality of the image captured and the data that has to be decoded by the program.

By observing the tests conducted, the chi-square values from the three filters are all 2.142857. With these result, we can conclude that the null hypothesis for Red, Green, Blue filters and also in the unmodified QR codes are accepted while in the modified QR codes test, which has a chi-square value of 19.2, rejects the null hypothesis and accepts the alternative hypothesis. 95% of the tested QR codes using Red and Blue filters failed to decode the QR code while 95% of the tested QR codes when testing using Green filters and decoding unmodified QR codes are successfully decoded. In modified QR codes, at least 5% of the test codes successfully decoded QR codes.

Among the three filters, the Green filter only achieved 100% success rate. This was selected for testing modified and unmodified QR codes. In unmodified QR codes, all of the QR codes tested were successfully detected while in modified QR codes, only 75% of the QR codes were successfully detected. The data and error correction areas of modification affected the QR code that makes the test failed in modified QR codes.

There are several factors to consider when decoding a QR code image. The success rate will depend on the distance of the camera from the QR code such as the quality of the captured image, the brightness that surrounds that area and the orientation of the QR code or the camera angle. As long as the camera and the decoder program can recognize the important features of a QR code such as the Position detection pattern, timing pattern and the black pixel it can successfully decode the QR code image. Based on the results, the group has successfully implemented the dedicated QR code scanner.

Chapter IV

CONCLUSION

Quick Response code or also popularly known as the QR Code is a two-dimensional code that stores more information than the one-dimensional code. Initially designed for automotive industry, the QR codes has now used in vast applications such as product tracking, item identification, marketing, commercial applications and much more. Compared to the Universal Product Code, which is a one-dimensional, QR codes has greater capacity and can store alphanumeric characters, kanji and bit strings. Recently, the decoders for the QR code are mobile phones and tablets. In this project, the group has implemented a dedicated QR code scanner that is powered by Arduino and Raspberry Pi. Multiple tests were conducted to satisfy the objectives. The ArduCAM shield attached to the Arduino microcontroller is connected to the Raspberry Pi. The ArduCAM and Arduino handles the capturing of the image and the Raspberry Pi handles the decoding.

In the first test, the group selected the best image filter to decode the QR code image. Using the RGB filters, the group compute the success rate by matching the generated QR code and the decoded result taken from the camera. Only the Green filter has achieved 100% success rate. The group selected the Green filter as it can be recognized by the QR decoder program more than the Red and Blue filters. The Red and Blue filters are distorted that it is unreadable to the decoder.

Next are scanning the QR code samples both error free and with various errors. The group observed that the decoder program successfully recognized and decoded the captured QR code image in the error-free QR code tests thus the success rate of the test is 100%. The

QR codes were then modified depending on the area of attack. The group observed that the decoder program successfully decoded the QR code image when the mask pattern, character encoding and character count was modified. The decoding fails only when the data and error correction were modified. With this results, the success rate is 75% of 32 images scanned.

During the scans, the group also recorded the time starting from with the capturing of image until display of the results. The group observed that the average scanning time is 40.46 seconds. This includes the decoding time inside the decoder program which has an average of 0.39 seconds.

By observing the tests conducted, the chi-square values from the three filters are all 2.142857. With these result, we can conclude that the null hypothesis for Red, Green, Blue filters and also in the unmodified QR codes are accepted while in the modified QR codes test, which has a chi-square value of 19.2, rejects the null hypothesis and accepts the alternative hypothesis. 95% of the tested QR codes using Red and Blue filters failed to decode the QR code while 95% of the tested QR codes when testing using Green filters and decoding unmodified QR codes are successfully decoded. In modified QR codes, at least 5% of the test codes successfully decoded QR codes.

There are several factors to consider when decoding a QR code image. The success rate will depend on the distance of the camera from the QR code such as the quality of the captured image, the brightness that surrounds that area and the orientation of the QR code or the camera angle. As long as the camera and the decoder program can recognize the important features of a QR code such as the Position detection pattern, timing pattern and the black pixel it can successfully decode the QR code image. Based on the results, the group has successfully implemented the dedicated QR code scanner.

Chapter V

RECOMMENDATION

As the group successfully implemented the dedicated QR code scanner using a microcontroller, there are recommendation that should be considered to improve both the hardware and software involved in this project. To make the QR code scanner a viable product, future researchers should consider the performance of the hardware and the software. Researchers can use different microcontrollers to create their own dedicated QR code scanner. It is important that the QR codes should be decoded in less time and should be compared to the performance of a smartphone-based QR code scanner applications. The camera quality should also be considered. The external environment, such as brightness and distance can affect the image quality seen by the camera module. Mentioned in the limitations of the project, the hardware and software is only limited to scanning and decoding QR code version 1 images only. Further improvements in both hardware and software by future researchers should be necessary to make the QR code scanner capable of scanning QR code version 2 images and above. Researchers can also implement a dedicated two-dimensional scanner aside from QR codes such as the Aztec code, Data Matrix and NexCode. The potential of a QR code scanner is huge, future researchers can contribute to the improvement of the QR code scanner.

Bibliography

ISO/IEC 18004 (2006). Information technology – Automatic Identification and Data Capture techniques – QR Code 2005 bar code symbology specification, ISO/IEC, Switzerland

Acharya T., and Ray A. K. (2006). Image Processing – Principles and Applications, John Wiley & Sons, Inc., Hoboken, New Jersey.

Sandberg, K., (2011). Introduction to Image Processing in Matlab. Digital Image Processing.

A-Lin H., et al, (2011). QR code image detection using run-length coding. International Conference on Computer Science and Network Technology, 2130 – 2134.

Sun M., et al. Identification of QR Code Based on Pattern Recognition. 397 – 401

Belussi L.F.F., et al, Fast QR Code Detection in Arbitrarily Acquired Images. 1 – 8

Kieseberg, P., et al, (2012). Malicious Pixels Using QR Codes as Attack Vector, Khalil I., and Mantoro, T. Trustworthy Ubiquitous Computing, Atlantis Press.

Heath S., (2003). What is an embedded system? Embedded Systems Design, Newnes. Burlington MA.

IC Imaging Control C++: RGB555/RGB565 | Technical Documentation | Image Acquisition Components, Imaging Control® The Imaging Source, [Online], http://www.imagingcontrol.com/en_US/support/documentation/class/PixelformatRGB565.htm [Accessed: 12 December, 2013]

Massimo Banzi, (2009), Getting Started with Arduino First Edition, MakeBooks, Sebastopol, California, USA pages 1-3, 19-24

Arduino – ArduinoBoardUNO, [online], http://arduino.cc/en/Main/arduinoBoardUno, [Accessed: 12 December 2013]

Introduction | Arduino Based Camera, [Online, 15 March, 2013], http://www.arducam.com/introduction/#more-444, [Accessed: 12 December 2013]

ArduCAM – An Arduino Camera Shield, [Online, 18 September, 2012], http://www.arducam.com/arducam/, [Accessed: 12 December 2013]

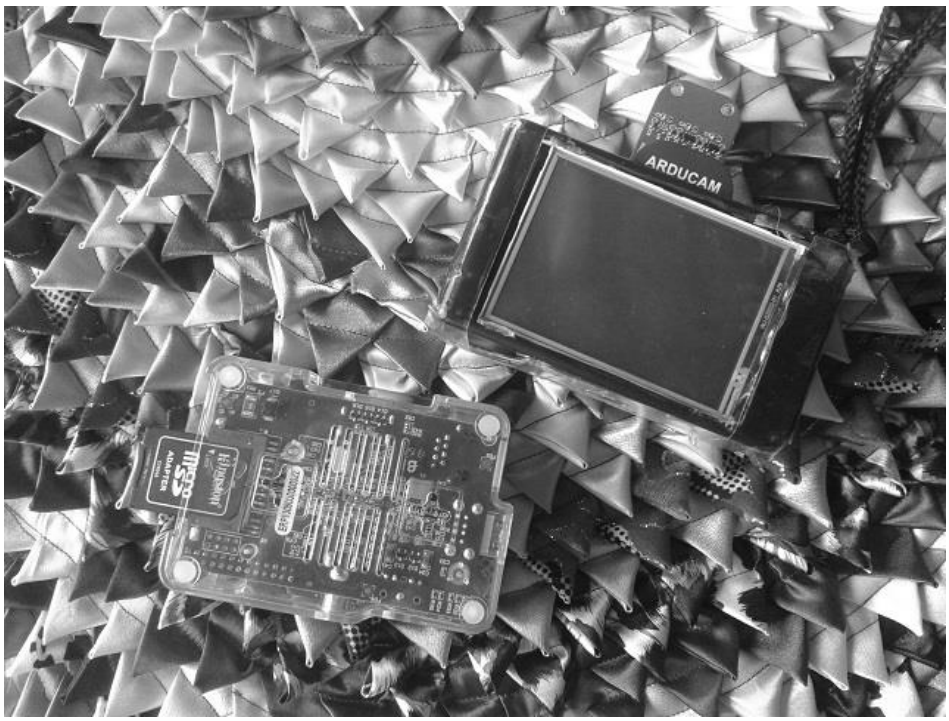
MT9D11 Specification Sheet, (2004), Micron Technology. Page 1

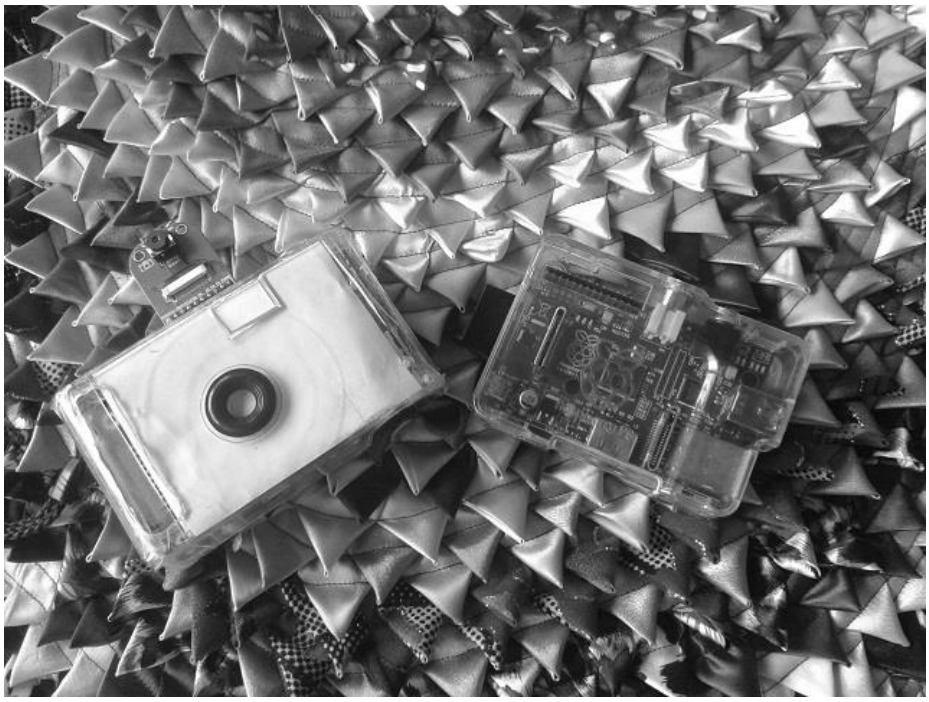
Richardson M. and Wallace S., (2013), Getting Started with Raspberry Pi, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Zbar Bar Code Reader Library Documentation, [Online, 2008], Jeff Brown, http://zbar.sourceforge.net/api/index.html

APPENDIX

Pictures of the Prototype






ARDUCAM


ARDUCAM

Datasheets

Arduino UNO

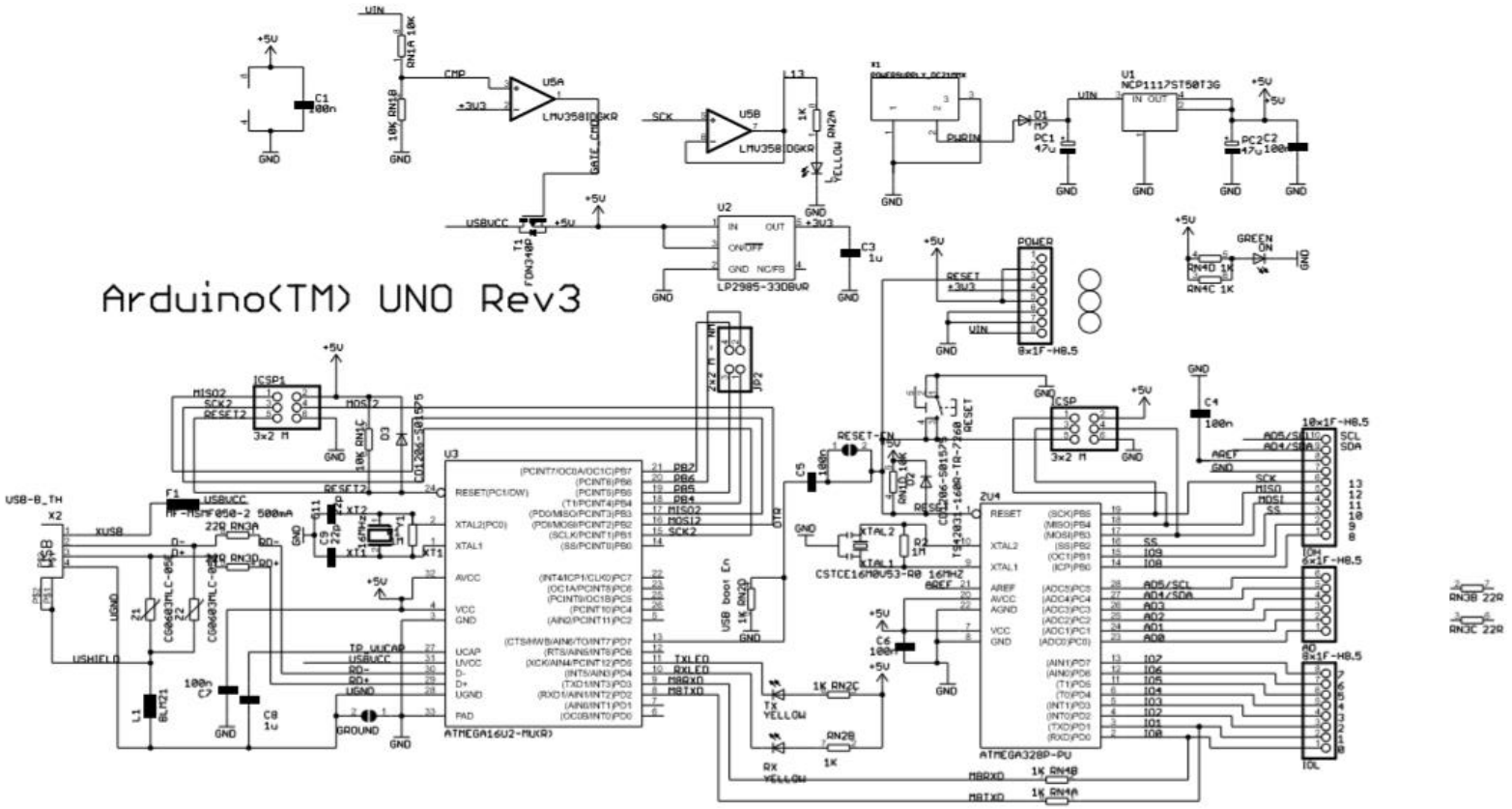
Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48PA/88PA/168PA/328P)
  - 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
  - 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 20 MHz @ 1.8 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 µA
  - Power-save Mode: 0.75 µA (Including 32 kHz RTC)

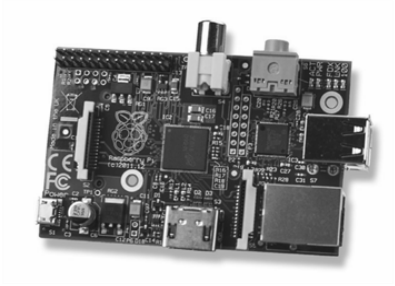
8-bit AVR® Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash

ATmega48PA
ATmega88PA
ATmega168PA
ATmega328P

Rev. 8161D–AVR–10/09



Raspberry Pi



MODEL B

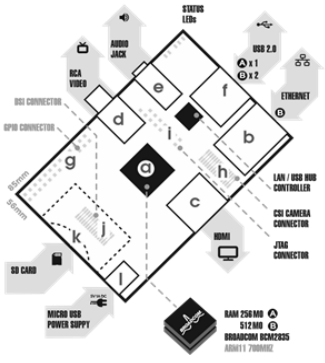
Product Name	**Raspberry Pi Model B**
Product Description	The Raspberry Pi is a small, powerful and lightweight ARM based computer which can do many of the things a desktop PC can do. The powerful graphics capabilities and HDMi video output make it ideal for multimedia applications such as media centres and narrowcasting solutions. The Raspberry Pi is based on a Broadcom BCM2835 chip. It does not feature a built-in hard disk or solid-state drive, instead relying on an SD card for booting and long-term storage.
RS Part Number	**756-8308**

Specifications

Chip	Broadcom BCM2835 SoC (a)
Core architecture	ARM11
CPU	700 MHz Low Power ARM1176JZFS Applications Processor
GPU	Dual Core VideoCore IV® Multimedia Co-Processor Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
Memory	512MB SDRAM
Operating System	Boots from SD card, running a version of the Linux operating system
Dimensions	85.6 x 53.98 x 17mm
Power	Micro USB socket 5V, 1.2A (l)

Connectors:

Ethernet	10/100 BaseT Ethernet socket (b)
Video Output	HDMI (rev 1.3 & 1.4) (c); Composite RCA (PAL and NTSC) (d)
Audio Output	3.5mm jack (e), HDMI
USB 2.0	Dual USB Connector (f)
GPIO Connector	26-pin 2.54 mm (100 mil) expansion header: 2x13 strip. Providing 8 GPIO pins plus access to I²C, SPI and UART as well as +3.3 V, +5 V and GND supply lines (g)
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2) (h)
JTAG	Not populated (i)
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane (j)
Memory Card Slot	SDIO (k)



Accessories

▲ Camera Module
775-7731

▲ International power supply
765-3311

▲ 8GB SD card pre-programmed with NOOBS - **779-6770**

▲ Expansion board
772-2974

▲ WiFi dongle
760-3621

▲ 10400mAh Li-Ion battery pack
775-7517

▲ Raspberry Pi user guide
768-6686

RS

www.rs-components.com/raspberrypi

ArduCAM (MT9D111)

1/3.2-Inch System-On-A-Chip (SOC) CMOS Digital Image Sensor

MT9D111

Features

- DigitalClarity™ CMOS Imaging Technology
- Superior low-light performance
- Ultra-low-power, low-cost
- Internal master clock generated by on-chip phase-locked loop oscillator (PLL)
- Electronic rolling shutter (ERS), progressive scan
- Integrated image flow processor (IFP) for single-die camera module
- Automatic image correction and enhancement, including lens shading correction
- Arbitrary image decimation with anti-aliasing
- Integrated real-time JPEG encoder
- Integrated microcontroller for flexibility
- Two-wire serial interface providing access to registers and microcontroller memory
- Selectable output data format: ITU-R BT.601 (YCbCr), 565RGB, 555RGB, 444RGB, JPEG 4:2:2, JPEG 4:2:0, and raw 10-bit
- Output FIFO for data rate equalization
- Programmable I/O slew rate
- Xenon and LED flash support with fast exposure adaptation
- Flexible support for external auto focus, optical zoom, and mechanical shutter

Applications

- Cellular phones
- PC cameras
- PDAs

Table 1: Key Performance Parameters

Parameter		Value
Optical format		1/3.2-inch (4:3)
Full resolution		1,600 x 1,200 pixels (UXGA)
Pixel size		2.8µm x 2.8µm
Active pixel array area		4.73mm x 3.52mm
Shutter type		Electronic rolling shutter (ERS) with global reset
Maximum frame rate		15 fps at full resolution, 30 fps in preview mode, (800 x 600)
Maximum data rate/ master clock		80 MB/s 6 MHz to 80 MHz
Supply voltage	Analog	2.5V–3.1V
	Digital	1.7V–1.95V
	I/O	1.7V–3.1V
	PLL	2.5V–3.1V
ADC resolution		10-bit, on-die
Responsivity		1.0/lux-sec (550nm)
Dynamic range		71dB
SNRMAX		42.3dB
Power consumption		348mW at 15 fps, full resolution
		223mW at 30 fps, preview mode
Operating temperature		-30°C to +70°C
Package		Bare die, 64-ball iCSP

Ordering Information

Table 2: Available Part Numbers

Part Number	Description
MT9D111	64-ball iCSP
MT9D111D00STCK15LC1	Bare die
MT9D111W00STCK15LC1	Wafer

Dedicated MCU-Based QR Code Scanner using Image Processing

Mark Angelo B. Domingo, Patrick Jovit A. Jove, Dario Lemuel D. Ty.

School of Electrical, Electronics and Computer Engineering, Mapua Institute of Technology
Intramuros, Manila, Philippines

***Abstract*— The study entitled "Dedicated MCU-based QR Code Scanner Using Image Processing" aims to implement hardware dedicated QR code scanner to translate version 1 QR code to its text equivalent. The device involves a camera interfaced to a microcontroller with the output seen on a LCD display. Testing of the device involves selection of proper filters for the camera, and four possible attack scenarios for the position of the QR code and successfully recognizing the value embedded in the code using the standard decoding algorithm of QR codes.**

Index Terms— **QR code, Microcontroller, Camera, Arduino, Raspberry Pi**

INTRODUCTION

Barcodes are symbols that can be read by optical machines. The barcode contains the data that is readable in those machines. Originally, the data inside the barcode are given with varying widths and spaces. These barcodes are usually referred to as a one-dimensional kind. Then, we use various shapes such as dots and rectangles in two-dimensional bar codes for the symbol. Barcode readers are used to scan barcodes and are usually seen in commercial establishments, it also presently used in more devices such as smartphones and tablets. Since 2D Barcodes such as the QR code stores more information than 1D Barcodes, it is most commonly used in advertisements where the URL of a certain company is encoded into the QR code, or even a QR equivalent conversion of a business card. QR codes may be used in a lot of possible applications such as Identification (replacement for the RFID tags), Data Storage, Advertisements, Business Cards, etc.

In early years barcodes were used. One dimensional barcodes are universally used as a product identification representation. This kind of barcodes is usually seen in books, magazines, appliances, medicines, postal main and food packaging. This allows merchants and suppliers to keep track of inventory both coming into stores or suppliers and being sold or transferred. It was present in almost all commercial merchandise. Decoders for the one-dimensional barcodes are dedicated. As one-dimensional barcodes are only scanned horizontally the capacity of the stored information is limited. The 2D barcode solves this kind of problem. In contrast with the one-dimensional barcode, 2D barcode such as QR Codes can be scanned horizontally and vertically and as a result, this kind of barcode has larger capacity than 1D barcode. It can process the image even when not properly aligned. Given its obvious advantage, it is a better choice to use this technology. This kind of technology is usually decoded by using mobile devices such as smartphones and tablets. Translators and decoders are integrated in the gadgets. And because of that, QR codes are popularly used for mobile marketing.

There are limitations though, such as, the QR code scanners are only available on smartphones and web applications (such as online QR code scanners, readers, encoders and decoders). This limits the potential of the QR code and also the 2D barcode as a replacement for 1D barcode, a product identification symbol, business card identification, and a container of short messages. As of now, there are no available QR code scanners yet in the market with a dedicated hardware for decoding the QR codes.

The aim of this study is to implement a dedicated microprocessor-based QR code scanner. The following objectives are needed in this study: (1) Capture the image using the camera of the device and apply image processing to make the image practical for scanning (2) Measure the total time it takes to capture, process, decode, and display the result of the scanned QR code with respect to varying character length and data type encoding (3) Measure the success rate of decoding error free QR codes (4) Measure the success rate of decoding QR codes introduced with errors in different attack scenarios

METHODOLOGY

The MCU-based QR code scanner is used to decode different QR codes with various character encodings specifically: Numeric, Alphanumeric, and Binary. Errors are introduced at different attack scenarios to test the systems capability of correcting it. The Original QR Codes are generated using the online QR code generator by Raco Industries. Their online QR code generator is very manageable that it allows us to choose the version of the QR code, level of error correction, and the mode of character encoding that we want to use. The QR code is captured using a camera interfaced into a microcontroller, specifically, the Arduino Uno interfaced with Arducam Shield Rev. B respectively. The image is fetched from the image buffer of the LCD attached to Arducam byte-per-byte using FIFO (First in, First Out) algorithm, and then processed by the Arduino to produce an image practical for QR code decoding. The processed image is then sent to the Raspberry Pi that runs on Linux via Serial Communication. The Raspberry Pi uses the Zbartools library to decode the image, and then sends the result back to the Arduino using Serial Communication. The Arduino then displays the result on the LCD attached to the Arducam shield. The original encoded data is compared to the decoded message. The total time of the whole process, from capturing the image to displaying the result is measured using an ordinary stopwatch, while the decoding time is displayed on the

Raspberry Pi's terminal. The QR codes used in this research are encoded using an online QR code generator. From the code text box, we entered the original message, then we set the level of error correction to L or Low, in which 7% of the original message can be restored, then we set the character encoding to Alphanumeric, and then left the other fields unchanged, then clicked on the generate button. We repeat these processes while giving variation to the character encoding of the QR code. When the codes are scanned using our system, it might result to a success, or a failure. The result is considered a success if the decoded message by the system matches the original encoded message. Therefore, the ratio of success is calculated using the formula:

$$SUCCESS\ RATE = \left[\frac{Number\ of\ Codes\ Successfully\ decoded}{Total\ number\ of\ Encoded\ codes}\right] x100\%$$

There are four possible attack scenarios that could possibly change or corrupt the message encoded in the QR code. These errors are fixed by the scanner in order for it to display the original encoded message. We intentionally introduced errors on these areas and scanned it using the dedicated scanner, then compare it to the encoded message. If the decoded message matches the original encoded message, it is counted as one of the successfully decoded codes.

The statistical test used in the following tests is the Chi-Square Test. This kind of test was chosen because it was simple and it is commonly used for the goodness of fit, which is referred to how the results observed close to what it is expected based on the hypothesis given. The formula for the chi-square test is:

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

Where:

O is the observed data in each category
E is the expected data based on the hypothesis
$\sum$ is the sum of calculations for each category
X2 is the chi-square value.

In every test conducted, there is a chi-square value that will be computed. In testing with red filters, the null hypothesis is that there will at least 95% will fail the test and the alternative hypothesis is that at least 5% of the tested QR code will be successfully decoded. The same hypotheses apply to the blue filters. In the test using green filter, the null hypothesis is that at least 95% of the QR codes will be decoded successfully and the alternative hypothesis is that at least 5% of the QR codes will pass. These hypotheses also apply in the testing of unmodified and modified QR codes. The degree of freedom used is 1 and the level of significance used in the test is 0.05.

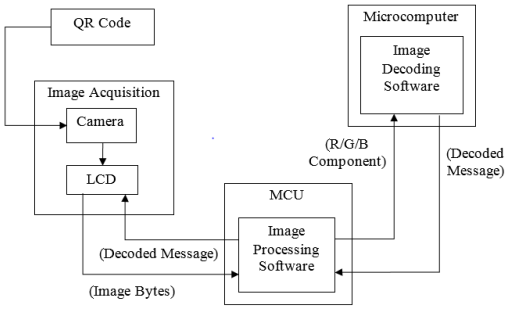


Fig.1. Conceptual Framework

QR Code image will be acquired using a camera module. Image coming from the camera is displayed on the LCD. The LCD display will temporarily store the image from the camera, and then sends the image byte by byte to the MCU which processes the image to separate the RGB component. The Red, Green, or Blue component of the image will be used by the decoder installed on a microcomputer to decode the image and send the result back to the MCU. The MCU will control the LCD to display the decoded message. Depending on the success rate of decoding the image, the group will choose the best component of the image that will be used for the experiment.

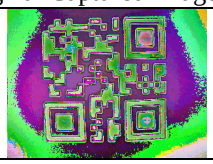
RESULTS AND DISCUSSION

The original QR codes are already generated using an online QR code generator. The encoder is set to encode QR codes in version 1, with a level of error correction L. The mode of character encoding and the length of the message being encoded are different for each test.

Objective 1: Capture the image using the camera of the device and apply image processing to make the image practical for scanning. Scan the generated QR code using the dedicated scanner. Extract the original image captured by the hardware and capture the image again this time by using RGB filters. Modify the program to change the filters to red, green or blue. Decode the images then write down the decoded message. Calculate the success rate in each table and then select the best type of filter for decoding the QR code image to proceed to the next test.

In this test, the tables are separated by each color filter: Red, Green and Blue. This was done in that order. Before starting the test, the program should be modified first to proceed with this test. First is to modify the program so that the image extracted is original. The results are shown in the table below. The Generated QR code is in the left column and the captured image is in the right column of the table.

GENERATED AND CAPTURED QR CODE

Generated QR Code	Original Captured Image
	

The generated QR code in the left contains the 0123456789 message. It is encoded with QR code version 1, configured to the lowest error correction level, L and the character encoding is Numeric encoding. The QR code in the right is the raw image extracted from the hardware. The image is the usual output of the camera. The program is based on the open source program included in the Arduino microcontroller. The next steps are to use RGB filters in each QR codes in the table.

First in the RGB filter test, use the red filter by modifying the program in the Arduino IDE then upload it to the microcontroller. There are thirty QR code images to be scanned and decoded. After scanning all the QR codes, check whether the results match the encoded message.

EXTRACTED IMAGES USING RED FILTER:

	OBSERVED (O)	EXPECTED (E)	(O-E)	(O-E)2	$\frac{(O-E)^2}{2}$
DETECTED	0	2	-2	4	2
NOT DETECTED	30	28	2	4	0.142857
TOTAL	30	30			2.142857

$$SUCCESS\ RATE = \left[\frac{0}{30}\right] x100\% = 0\ \%$$

Based on the results, the QR code image scanned cannot be recognized by the decoder program of the microcontroller. It is observed that the scanned images are somehow distorted and makes the microcontroller having a hard time to recognize the whole QR code image. Overall, the test proves that the red filter was not viable for decoding, giving a 0% success rate. The chi-square value is 2.142857. The degree of freedom is 1 and the level of significance is 0.05. The value obtained lies between 2.07 and 3.84 or 2.07<P<3.84. As a result of this, we accept the null hypothesis for red filters which is that at least 95% of the tested QR codes will not be successfully decoded

The next test is to use the green filter by modifying the program in the Arduino IDE then uploads it to the microcontroller. There are six QR code image to be scanned and decoded. After scanning all the QR codes, check whether the results match the encoded message.

EXTRACTED IMAGES USING GREEN FILTER:

	OBSERVED (O)	EXPECTED (E)	(O-E)	(O-E)2	$\frac{(O-E)^2}{2}$
DETECTED	30	28	2	4	0.142857
NOT DETECTED	0	2	-2	4	2
TOTAL	30	30			2.142857

$$SUCCESS\ RATE = \left[\frac{30}{30}\right] x100\% = 100\ \%$$

Based on the results, some QR code image scanned can be recognized by the decoder program of the microcontroller. The important features of the QR code are very clear in the scanned QR codes. This makes the microcontroller recognize the whole QR code image in the whole test. This gives the test a 100% success rate. The chi-square value is 2.142857. The degree of freedom is 1 and the level of significance is 0.05. The value obtained lies between 2.07 and 3.84 or 2.07<P<3.84. As a result of this, we accept the null hypothesis for green filters which is that at least 95% of the tested QR codes will be successfully decoded

The next test is to use the blue filter by modifying the program in the Arduino IDE then uploads it to the microcontroller. There are six QR code image to be scanned and decoded. After scanning all the QR codes, check whether the results match the encoded message.

EXTRACTED IMAGES USING BLUE FILTER:

	OBSERVED (O)	EXPECTED (E)	(O-E)	(O-E)2	$\frac{(O-E)^2}{2}$
DETECTED	0	2	-2	4	2
NOT DETECTED	30	28	2	4	0.142857
TOTAL	30	30			2.142857

$$SUCCESS\ RATE = \left[\frac{0}{30}\right] x100\% = 0\ \%$$

Based on the results, just like the test during the red filter, the QR code image scanned cannot be recognized by the decoder program of the microcontroller. It is observed that the scanned images are somehow distorted and also the QR codes are barely visible and makes the microcontroller having a hard time to recognize the whole QR code image. This gives the test a 0% success rate. The chi-square value is 2.142857. The degree of freedom is 1 and the level of significance is 0.05. The value obtained lies between 2.07 and 3.84 or 2.07<P<3.84. As a result of this, we accept the null hypothesis for blue filters which is that at least 95% of the tested QR codes will not be successfully decoded

Based on the observations, the success rates of the RGB filters are 0 % for red, 100 % for green and 0 % for blue filter. Among the three filters that has been applied in the QR Code image, the Green filter was the one that was used in decoding the QR Code. It is the only image that is much detailed than the Red and Blue filters which can be described as distorted and dark.

Objective 3: Measure the success rate of decoding error free QR codes. Scan the generated QR code using the dedicated scanner. Compare the decoded message to the original encoded message. If the decoded message matches the encoded message, it is counted as a success. Record the time on table 3 and compute the success rate.

The QR code image on the left is encoded with QR code version 1, Level L error correction and various character encoding methods except Kanji. The scanned images from the camera are

placed in the right side of the table. The green filter was used in running this test as the result of the image filters test previously. Each of the QR code image on the left were scanned using the hardware. The decoded message will be displayed after some time. The group extracted the scanned image from the hardware and placed it on the table below. The results are also placed along with it.

DECODING OF QR CODE IMAGES

	OBSERVED (O)	EXPECTED (E)	(O-E)	(O-E)2	$\frac{(O-E)^2}{2}$
DECODED	30	28	2	4	0.142857
NOT DECODED	0	2	-2	4	2
TOTAL	30	30			2.142857

$$SUCCESS\ RATE = \left[\frac{30}{30}\right] x100\% = 100\ \%$$

Based on the results, 30 out of 30 of the QR code images scanned in the table can be recognized by the decoder program of the hardware. The important features of the QR code such as the finder pattern, format information, and especially the data and error correction are very clear in the scanned image and make the hardware decode the message. The time was also recorded for use in the next tests. With these results, the test has achieved a 100% success rate. The chi-square value is 2.142857. The degree of freedom is 1 and the level of significance is 0.05. The value obtained lies between 2.07 and 3.84 or 2.07<P<3.84. As a result of this, we accept the null hypothesis for unmodified QR codes which is that at least 95% of the tested QR codes will be successfully decoded

Objective 4: Measure the success rate of decoding QR codes introduced with errors in different attack scenarios. Intentionally add errors on the generated QR codes on part 1. Scan it again using the dedicated scanner. Compare the results from the original encoded data and record the time on table 3 and compute the success rate.

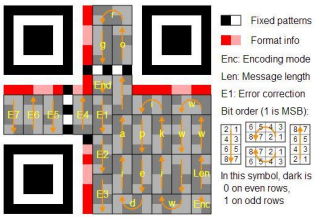


Fig.3.8 QR Code structure

The figure shows the QR code structure, in this test the areas that should be modified. The format info contains the mask pattern (mask area) and the error correction level. The gray area is where the data and the error correction are placed. In the bottom right corner of the QR code, the character encoding is labeled with 'Enc'. Above this is the character length which is labeled with 'Len'.

The QR code image on the left of the table below is encoded with QR code version 1, Level L error correction and various character encoding methods except Kanji. The scanned images from the camera are placed in the right side of the table.

The green filter was used in running this test as the result of the image filters test previously. The test starts by generating QR codes given in the website. The group then modifies the QR code image depending on the attack area given in the table. The group chose to modify the Mask, Character Encoding, Character Area and the Data and Error Correction area of the QR Code. The group uses an image manipulation program in a computer. After modifying, the group scanned the image; the result will appear in the display for some time. Extract the QR code from the hardware and place it in the rightmost column of the table along with the displayed results.

DECODING OF MODIFIED QR CODE IMAGES

The QR codes were encoded with the message "Hello World". QR codes were modified depending on the attack area and labeled to identify the changes made in the original.

QR # 32	Modified QR:	Captured Image:
		
Encoded Message: Hello World	Attack Area: Char. Encoding	Decoded Message: Hello World
		Result: SUCCESS

The QR code on the left contains "Hello World" which is generated from the QR code generator web application. In the center is the modified QR code. The red boxes in the center image are where the mask area is located. In comparison with the left image, the mask area is filled with an additional black pixels each side using an image manipulation program. This changes the mask pattern of the whole QR code. When the group scanned the modified QR code, the results shows "Hello World", this is the same as the content of the original QR code image. This test was counted as a success.

QR # 31	Modified QR:	Captured Image:
		
Encoded Message: Hello World	Attack Area: Mask Pattern	Decoded Message: Hello World
		Result: SUCCESS

Same as the first test, the QR code on the left contains "Hello World". In the center QR code image, the red box on the lower right corner means that it is the character encoding area, the area

to be modified. Compared with the original image, the character encoding area is fully filled with pixels. This changed the character encoding type of the QR code from alphanumeric to invalid. The group scanned the modified QR code image and the result shown is "Hello World", same as the original code. The contents are unaffected regardless of the modification. This test was counted as a success.

QR # 33	Modified QR:	Captured Image:
		
Encoded Message: Hello World	Attack Area: Char. Count	Decoded Message: Hello World
		Result: SUCCESS

The QR code used in this test is contains "Hello World". The red box in the lower left of the QR code image in the center is the character count area. The original image was modified using an image manipulation program. The modified QR code in the center changed its character count by removing the pixels in that area. This changes the character count from a different one. The group scanned the center image and got a matching result: "Hello World". The contents of the code are unaffected and the test was counted as a success.

QR # 34	Modified QR:	Captured Image:
		
Encoded Message: Hello World	Attack Area: Data & Error Correction	Decoded Message: NONE
		Result: FAILED

The "Hello World" QR code in this test was used. In the center image is the modified QR code. The QR code was modified using the same techniques used in the first 3 tests. There are four red boxes in the center image. These are the areas of attack in this test. In boxes # 1, 2, and 4, the pixels was removed thus modified the error correction area of the QR code. In box # 4, the pixels were also removed. This time, the data of the QR code was modified. The group scanned the modified QR code and didn't get any results. The contents of the modified QR code cannot be recovered anymore because of the modification, making this test marked as a failure.

	OBSERVED (O)	EXPECTED (E)	(O-E)	(O-E)2	$\frac{(O-E)^2}{2}$
DECODED	24	30	-6	36	1.2
NOT DECODED	8	2	6	36	18
TOTAL	32	32			19.2

$$SUCCESS\ RATE = \left[\frac{24}{32}\right] x100\% = 75\ \%$$

Based on the results, the decoder program can still recognized a modified QR code except those with altered data and error correction areas. The data and error correction area as explained in the previous chapters are the largest part of the QR code. If someone modifies a data or the error correction area in the QR code, the damage is significant and the data cannot be recovered no matter what. Therefore, as long as these areas are still intact or have minimal damage, the QR code can still be recognized. This gives the test a 75% success rate as expected. The chi-square value is 19.2. The degree of freedom used is 1 and the level of significance used is 0.05. Based on the results, the chi-square value is larger than 3.84 or P>3.84. The null hypothesis is rejected and the alternative hypothesis is accepted which is at least 5% of the test in modified QR codes will pass.

Objective 2: Measure the total time it takes to capture, process, decode, and display the result of the scanned QR code with respect to varying character length and data type encoding. Observe and record the delay of decoding a QR code in different scenarios using the decoding time inside the program and a stopwatch.

The group recorded the time in the previous tests in tables 3.5 and 3.6. The total time is the sum of the scanning time and the decoding time. There are total of 18 trials in this test. The equation below is to get the scanning time by getting the difference between the total time and the decoding time:

$$Scanning\ time = Total\ Time - Decoding\ Time$$

MEASUREMENT OF TIME

QR #	Message	Encoding	Type of Error	Decoding Time	Scanning Time	Total Time (in sec)
1	Hello	Alphanumeric	None	0.39 s	40.00 s	40.39
2	Hello World	Alphanumeric	None	0.38 s	40.07 s	40.46
3	Hi	Binary	None	0.39 s	39.88 s	40.27
4	Hi World	Binary	None	0.39 s	39.95 s	40.34
5	12345	Numeric	None	0.41 s	39.99 s	40.40
6	0123456789	Numeric	None	0.39 s	39.73 s	40.12
7	####Sample####	Alphanumeric	None	0.41 s	39.95 s	40.36
8	@qrcode #sample %	Alphanumeric	None	0.40 s	40.10 s	40.50
9	~Message Text!@	Alphanumeric	None	0.40 s	40.06 s	40.46
10	00012345000	Numeric	None	0.38 s	39.90 s	40.28
11	246813579	Numeric	None	0.40 s	39.94 s	40.34
12	98124758392	Numeric	None	0.38 s	39.99 s	40.37
13	110101011101	Binary	None	0.37 s	39.97 s	40.34
14	000010101001111001010	Binary	None	0.38 s	40.03 s	40.41
15	ABCDefgh	Alphanumeric	None	0.39 s	40.00 s	40.39

16	Dario Ty	Alphanumeric	None	0.38 s	40.05 s	40.43
17	facebook.com/home	Alphanumeric	None	0.39 s	40.08 s	40.47
18	google.com.ph	Alphanumeric	None	0.39 s	39.96 s	40.35
19	Hello Operator 1	Alphanumeric	None	0.40 s	39.92 s	40.32
20	hello00910	Alphanumeric	None	0.38 s	40.02 s	40.40
21	Mapua Institute	Alphanumeric	None	0.39 s	39.93 s	40.32
22	Mapua Intramuros	Alphanumeric	None	0.41 s	40.02 s	40.43
23	Mark Domingo	Alphanumeric	None	0.39 s	39.96 s	40.35
24	Patrick Jove	Alphanumeric	None	0.40 s	40.01 s	40.41
25	qr@raco.com!	Alphanumeric	None	0.40 s	40.10 s	40.50
26	readablecode#30	Alphanumeric	None	0.40 s	40.06 s	40.46
27	sampleQRCODE	Alphanumeric	None	0.39 s	39.96 s	40.35
28	!@#$%^&*()_+|	Alphanumeric	None	0.41 s	40.13 s	40.54
29	Thesis Sample	Alphanumeric	None	0.40 s	39.91 s	40.31
30	wikipedia.org	Alphanumeric	None	0.39 s	39.99 s	40.38
31	Hello World	Alphanumeric	Mask	0.38 s	40.29 s	40.67
32	Hello World	Alphanumeric	Char Enc	0.38 s	40.00 s	40.38
33	Hello World	Alphanumeric	CharCount	0.39 s	40.18 s	40.57
34	Hello World	Alphanumeric	Data+E.C	0.39 s	40.15 s	40.54
35	0123456789	Numeric	Mask	0.38 s	40.18 s	40.56
36	0123456789	Numeric	Char Enc	0.39 s	40.09 s	40.48
37	0123456789	Numeric	CharCount	0.42 s	40.08 s	40.50
38	0123456789	Numeric	Data+E.C	0.38 s	40.19 s	40.57
39	Hi World	Binary	Mask	0.39 s	40.28 s	40.67
40	Hi World	Binary	Char Enc	0.38 s	40.47 s	40.85
41	Hi World	Binary	CharCount	0.42 s	40.21 s	40.63
42	Hi World	Binary	Data+E.C	0.38 s	39.74 s	40.12
43	246813579	Numeric	Mask	0.38 s	40.17 s	40.55
44	246813579	Numeric	Char Enc	0.39 s	40.09 s	40.48
45	246813579	Numeric	CharCount	0.41 s	40.13 s	40.54
46	246813579	Numeric	Data+E.C	0.39 s	40.13 s	40.52
47	Dario Ty	Alphanumeric	Mask	0.39 s	40.15 s	40.54
48	Dario Ty	Alphanumeric	Char Enc	0.39 s	40.09 s	40.48
49	Dario Ty	Alphanumeric	CharCount	0.40 s	40.08 s	40.48
50	Dario Ty	Alphanumeric	Data+E.C	0.38 s	40.17 s	40.55
51	Mark Domingo	Alphanumeric	Mask	0.39 s	40.12 s	40.51
52	Mark Domingo	Alphanumeric	Char Enc	0.38 s	40.19 s	40.57
53	Mark Domingo	Alphanumeric	CharCount	0.42 s	40.06 s	40.48
54	Mark Domingo	Alphanumeric	Data+E.C	0.39 s	40.21 s	40.60
55	Patrick Jove	Alphanumeric	Mask	0.38 s	40.16 s	40.54
56	Patrick Jove	Alphanumeric	Char Enc	0.38 s	40.25 s	40.63
57	Patrick Jove	Alphanumeric	CharCount	0.42 s	40.17 s	40.59
58	Patrick Jove	Alphanumeric	Data+E.C	0.38 s	40.22 s	40.60
59	Thesis Sample	Alphanumeric	Mask	0.38 s	40.16 s	40.54
60	Thesis Sample	Alphanumeric	Char Enc	0.39 s	40.17 s	40.56
61	Thesis Sample	Alphanumeric	CharCount	0.41 s	39.99 s	40.40
62	Thesis Sample	Alphanumeric	Data+E.C	0.38 s	40.07 s	40.45

The leftmost column is the QR code number and the numbers are indicated in each generated QR code images in the previous tests in tables 3.5 and 3.6. The second column is the original message encoded in the QR code of the respective number. The third and fourth column is the configuration of the QR code of the respective number such as the encoding and the type of attack used in the QR code. The decoding time indicates how long does the QR code can be decoded by the program. This is displayed on the results from the terminal only. The scanning time indicates how long the hardware and the software find the QR code and the total time is the sum of both the decoding time and the scanning time. This was measured by the stopwatch. The scanning time was taken from the equation given previously. After all data was recorded, the group computes the average time of decoding, scanning and the total time.

Based on the computed results, the average decoding time was 0.3919 seconds. The average scanning time is 40.0718 seconds. Lastly, the total time is 40.4613 seconds. The count was started after pressing the trigger of the camera and ended when the results are displayed. This is where the total time was recorded. The decoding time is shown in the results displayed on the terminal. With the data that has been acquired, the time differs depending on the quality of the image captured and the data that has to be decoded by the program.

By observing the tests conducted, the chi-square values from the three filters are all 2.142857. With these result, we can conclude that the null hypothesis for Red, Green, Blue filters and also in the unmodified QR codes are accepted while in the modified QR codes test, which has a chi-square value of 19.2, rejects the null hypothesis and accepts the alternative hypothesis. 95% of the tested QR codes using Red and Blue filters failed to decode the QR code while 95% of the tested QR codes when

testing using Green filters and decoding unmodified QR codes are successfully decoded. In modified QR codes, at least 5% of the test codes successfully decoded QR codes.

Conclusion and Recommendation

Conclusion

Quick Response code or also popularly known as the QR Code is a two-dimensional code that stores more information than the one-dimensional code. Initially designed for automotive industry, the QR codes has now used in vast applications such as product tracking, item identification, marketing, commercial applications and much more. Compared to the Universal Product Code, which is a one-dimensional, QR codes has greater capacity and can store alphanumeric characters, kanji and bit strings. Recently, the decoders for the QR code are mobile phones and tablets. In this project, the group has implemented a dedicated QR code scanner that is powered by Arduino and Raspberry Pi. Multiple tests were conducted to satisfy the objectives. The ArduCAM shield attached to the Arduino microcontroller is connected to the Raspberry Pi. The ArduCAM and Arduino handle the capturing of the image and the Raspberry Pi handles the decoding.

In the first test, the group selected the best image filter to decode the QR code image. Using the RGB filters, the group computes the success rate by matching the generated QR code and the decoded result taken from the camera. Only the Green filter has achieved 100% success rate. The group selected the Green filter as it can be recognized by the QR decoder program more than the Red and Blue filters. The Red and Blue filters are distorted that it is unreadable to the decoder.

Next are scanning the QR code samples both error free and with various errors. The group observed that the decoder program successfully recognized and decoded the captured QR code image in the error-free QR code tests thus the success rate of the test is 100%. The QR codes were then modified depending on the area of attack. The group observed that the decoder program successfully decoded the QR code image when the mask pattern, character encoding and character count was modified. The decoding fails only when the data and error correction were modified. With these results, the success rate is 75% of 32 images scanned.

During the scans, the group also recorded the time starting from with the capturing of image until display of the results. The group observed that the average scanning time is 40.46 seconds. This includes the decoding time inside the decoder program which has an average of 0.39 seconds.

By observing the tests conducted, the chi-square values from the three filters are all 2.142857. With these result, we can conclude that the null hypothesis for Red, Green, Blue filters and also in the unmodified QR codes are accepted while in the modified QR codes test, which has a chi-square value of 19.2, rejects the null hypothesis and accepts the alternative hypothesis. 95% of the tested QR codes using Red and Blue filters failed to decode the QR code while 95% of the tested QR codes when testing using Green filters and decoding unmodified QR codes are successfully decoded. In modified QR codes, at least 5% of the test codes successfully decoded QR codes.

There are several factors to consider when decoding a QR code image. The success rate will depend on the distance of the camera from the QR code such as the quality of the captured image, the brightness that surrounds that area and the orientation of the QR code or the camera angle. As long as the camera and the decoder program can recognize the important features of a QR code such as the Position detection pattern, timing pattern and the black pixel it can successfully decode the QR code image. Based on the results, the group has successfully implemented the dedicated QR code scanner.

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads—the template will do that for you.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar.

Recommendation

As the group successfully implemented the dedicated QR code scanner using a microcontroller, there are recommendation that should be considered to improve both the hardware and software involved in this project. To make the QR code scanner a viable product, future researchers should consider the performance of the hardware and the software. Researchers can use different microcontrollers to create their own dedicated QR code scanner. It is important that the QR codes should be decoded in less time and should be compared to the performance of a Smartphone-based QR code scanner application. The camera quality should also be considered. The external environment, such as brightness and distance can affect the image quality seen by the camera module. Mentioned in the limitations of the project, the hardware and software is only limited to scanning and decoding QR code version 1 images only. Further improvements in both hardware and software by future researchers should be necessary to make the QR code scanner capable of scanning QR code version 2 images and above. Researchers can also implement a dedicated two-dimensional scanner aside from QR codes such as the Aztec code, Data Matrix and NexCode. The potential of a QR code scanner is huge, future researchers can contribute to the improvement of the QR code scanner.

ACKNOWLEDGMENT

Special people who have given their efforts, assistance and guidance to this thesis paper were behind the efficacious completion of this hard-earned work. Having shorn of them, this project would not have been accomplished.

First of all, the researchers would like to express their heartfelt gratitude to our Lord Almighty for all the blessings bestowed upon them, giving them the talents, perseverance, and knowledge to do the paper and prototype entirely. Every paragraph written, every module tested successfully, every application implemented, and every conclusion formulated, are all lifted onto our Lord's glory.

The researchers are also grateful to Engr. Jesus Martinez Jr., the thesis adviser, for shining a light of initial interest in the presented topic and for the unceasing guidance throughout the course of documentations and prototype. They highlighted the need, purpose, and significance of the course, giving out clear, direct, and well-organized thoughts and contributions in addition to the researchers' work.

To each member of the group, each one is thankful for the dedication, contribution, perseverance, wit, and positivity that have been offered. Thankful for the healthy brain storming, decision making, and conflict resolving that seldom arise.

Finally, having been thankful of our families for the love, understanding, and undying support, they have provided the group with financial aid, and the emotional support that drives the group towards their goal.

REFERENCES

[1] ISO/IEC 18004 (2006). Information technology – Automatic Identification and Data Capture techniques – QR Code 2005 bar code symbology specification, ISO/IEC, Switzerland

[2] Acharya T., and Ray A. K. (2006). Image Processing – Principles and Applications, John Wiley & Sons, Inc., Hoboken, New Jersey.

[3] Sandberg, K., (2011). Introduction to Image Processing in Matlab. Digital Image Processing.

[4] A-Lin H., et al, (2011). QR code image detection using run-length coding. International Conference on Computer Science and Network Technology, 2130 – 2134.

[5] Sun M., et al. Identification of QR Code Based on Pattern Recognition. 397 – 401

[6] Belussi L.F.F., et al, Fast QR Code Detection in Arbitrarily Acquired Images. 1 – 8

[7] Kieseberg, P., et al, (2012). Malicious Pixels Using QR Codes as Attack Vector, Khalil I., and Mantoro, T. Trustworthy Ubiquitous Computing, Atlantis Press.

[8] Heath S., (2003). What is an embedded system? Embedded Systems Design, Newnes. Burlington MA.

[9] IC Imaging Control C++: RGB555/RGB565 | Technical Documentation | Image Acquisition Components, Imaging Control® The Imaging Source, [Online], http://www.imagingcontrol.com/en_US/support/documentation/class/PixelformatRGB565.htm [Accessed: 12 December, 2013]

[10] Massimo Banzi, (2009), Getting Started with Arduino First Edition, MakeBooks, Sebastopol, California, USA pages 1-3, 19-24

[11] Arduino – Arduino Board UNO, [online], http://arduino.cc/en/Main/arduinoBoardUno, [Accessed: 12 December 2013]

[12] Introduction | Arduino Based Camera, [Online, 15 March, 2013], http://www.arducam.com/introduction/#more-444, [Accessed: 12 December 2013]

[13] ArduCAM – An Arduino Camera Shield, [Online, 18 September, 2012], http://www.arducam.com/arducam/, [Accessed: 12 December 2013]

[14] MT9D11 Specification Sheet, (2004), Micron Technology. Page 1

[15] Richardson M. and Wallace S., (2013), Getting Started with Raspberry Pi, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

[16] Zbar Bar Code Reader Library Documentation, [Online, 2008], Jeff Brown, http://zbar.sourceforge.net/api/index.html