**Vetscan Hub QR Testing**

**Table of Contents**

[1 Overview 2](#_Toc80368733)

[1.1 Scope of this Document 2](#_Toc80368734)

[1.2 Test Equipment 2](#_Toc80368735)

[1.3 Documentation 2](#_Toc80368736)

[1.4 QR Specification 3](#_Toc80368737)

[1.4.1 Provisional Specification of Zoetis Vetscan Ecosystem Consumable QR Code 3](#_Toc80368738)

[1.5 Test Labels 4](#_Toc80368739)

[1.6 Test Label Generation 4](#_Toc80368740)

[1.6.1 QR code maximum data capacity 5](#_Toc80368741)

[1.7 Test Label Printing 6](#_Toc80368742)

[2 Test 7](#_Toc80368743)

[2.1 Test Setup 7](#_Toc80368744)

[2.2 Glare with Varying LED Configurations 7](#_Toc80368745)

[2.3 Fixed Focus vs Autofocus 8](#_Toc80368746)

[3 Test Instructions 9](#_Toc80368747)

[3.1 Web App 9](#_Toc80368748)

[3.2 Galaxy S10 10](#_Toc80368749)

[3.3 Python App 11](#_Toc80368750)

[4 Test Results 12](#_Toc80368751)

[4.1 Galaxy S10 Cell Phone 12](#_Toc80368752)

[4.2 Laptop Web App 12](#_Toc80368753)

[4.3 Laptop Python App 12](#_Toc80368754)

[4.4 Vetscan Hub Mockup Web App 12](#_Toc80368755)

[4.5 Vetscan Hub Mockup Python App - Autofocus Off 12](#_Toc80368756)

[4.6 Vetscan Hub Mockup Python App - Autofocus On 12](#_Toc80368757)

[4.7 UPC Scanning 13](#_Toc80368758)

[4.8 “Data Matrix” decoding timing using pylibdmtx library 13](#_Toc80368759)

[4.8.1 Laptop - “Data Matrix” decoding times 14](#_Toc80368760)

[4.8.1 Vetscan Hub Mockup - “Data Matrix” decoding times 14](#_Toc80368761)

[5 Conclusions 15](#_Toc80368762)

[5.1 Python app versus web app 15](#_Toc80368763)

[5.2 Comparison of results 15](#_Toc80368764)

[5.3 Autofocus 16](#_Toc80368765)

[5.3.1 Autofocus Time to Scan 16](#_Toc80368766)

[5.3.2 Fixed Focus Time to Scan 17](#_Toc80368767)

[5.3.3 Graph of Autofocus vs Fixed Focus 18](#_Toc80368768)

[5.3.4 Autofocus Time to Scan – run with image rotated 19](#_Toc80368769)

[5.3.5 Fixed Focus Time to Scan – run with image rotated 19](#_Toc80368770)

[5.3.6 Graph of Autofocus vs Fixed Focus 19](#_Toc80368771)

[6 Recommendations 20](#_Toc80368772)

[6.1 Current Camera Module 20](#_Toc80368773)

[6.2 Future Improvements 20](#_Toc80368774)

[7 Revision History 21](#_Toc80368775)

# Overview

The Vetscan Hub will use an embedded camera to capture information on consumables being used by analyzers. That information will be encoded into QR codes that will be printed or otherwise affixed onto consumable packaging at the time of manufacturing.

This document contains a test plan and report involving the Vetscan Hub Mockup’s camera and its ability to read QR codes. Multiple combinations of label size, module size, total characters, and types of encoded data will be tested.

## Scope of this Document

This document will be used for engineering staff to plan and document testing.

## Test Equipment

* Vetscan Hub mockup unit with attached LED diffuser over camera flash.
* Galaxy S10 and Lenovo P50 cameras as points of comparison.
* Ruler used to measure size of the QR code label.
* Printed QR Labels. Encoded according to ISO 18004:2006, using Level M error correction, and the max amount of data that can be contained in a QR code for different version.
* qr\_code\_generator.py – A Python application used to generate QR codes.
* qr\_scanner.py – A Python application used to scan QR codes that uses open source pyZbar library.

## Documentation

|  |  |
| --- | --- |
| ISO 18004:2006 | QR code specification |
| <https://www.the-qrcode-generator.com/>scan | Web site – create or read a QR label with given data |
| <https://www.qrcode.com/en/about/version.html> | Web site – QR code density specification |

## QR Specification

A QR code (abbreviated from Quick Response code) is a type of matrix barcode (or two-dimensional barcode) invented in 1994 by the Japanese automotive company Denso Wave. A barcode is a machine-readable optical label that contains information about the item to which it is attached. In practice, QR codes often contain data for a locator, identifier, or tracker that points to a website or application. A QR code uses four standardized encoding modes (numeric, alphanumeric, byte/binary, and kanji) to store data efficiently; extensions may also be used.

Provisional specification for the QR codes for Zoetis Vetscan ecosystem consumables that are to be read by the Hub was provided by Martin Heller and Jeppe Sorrenson.

### Provisional Specification of Zoetis Vetscan Ecosystem Consumable QR Code

1. Format – the content of the QR code must be a valid JSON structure
2. JSON fields are all strings and must include the following
   1. analyzer – the name of the analyzer the consumable is designed to operate in
   2. uuid – the unique identifier of the specific consumable type
   3. lot – the manufacturing lot identification of the consumable
   4. exp – the expiration date in format YYYY-MM-DD
   5. data – a string of indeterminate length and structure. This can be comma separated values, text data, additional JSON data structure, ASCII encoded hex, etc. If no additional data is required, this will be an empty string.
3. Valid QR code according to specification (ISO 18004:2006)
4. Minimum of version 1 (21 x 21 modules)
5. Maximum of version 40 (177 x 177 modules)
6. Minimum size of physical QR code is 10 mm x 10 mm
7. Maximum size of physical QR code is 50 mm x 50 mm
8. Error correction level M
9. Labels printed on a high contrast background

Example showing values that will be filled in as <value> is 53 characters without any data or whitespace:

{

"analyzer": "<analyzer id>",

"uuid": "<uuid>",

"lot": "<string>",

"exp": "<yyyy-mm-dd>",

"data": "<optional data>"

}

An example SPE QR would be as follows with 114 characters when whitespace is removed:

{

"analyzer":"vetscan-spe",

"exp":"2022-05-07",

"uuid":"0b7ec890-3960-11eb-a081-2790e47ff2f4",

"lot":"1234",

"data":""

}

## Test Labels

* The test labels will be printed on standard printer paper by a laser printer
* The test label Sizes that will be used will be 10mm x 10mm, 20mm x 20mm, and 50mm x 50mm.
* The QR codes will contain example JSON data which is a combination of Numeric, Alphanumeric and Binary characters:
  + Numeric:
    - 0 1 2 3 4 5 6 7 8 9
  + Alphanumeric:
    - 0–9
    - A–Z (upper-case only)
    - space
    - $ % \* + - . / :
  + Binary:
    - a-z (lower-case)
    - { } \n “ ,
    - Every other character
* For example, when using the Version 20 QR Code with correction level M, the maximum allowable characters are:
  + Numeric: 1600
  + Alphanumeric: 970
  + Binary: 666
* Test labels will be filled with the maximum amount of data possible by version by adding characters in the data field that force the QR code to use binary encoding which is the least efficient encoding but which most accurately represents that data that would be placed into the fields and also represents a worst case scenario.

## Test Label Generation

* The qr\_code\_generator.py Python application was used to generate the images for the different label data densities with varying encoding types. The following GitHub library was used as a resource:
  + https://github.com/lincolnloop/python-qrcode

### QR code maximum data capacity

The following table shows the amount of characters that may be encoded for each version of QR codes 1-40. The columns indicate how much the version can hold by spec, how much the version can hold if we use the provisional JSON specification from Zoetis and only have alphanumeric data in the data field, and how much the version can hold if we use the provisional JSON specification from Zoetis and have complicated data (i.e. stringified JSON object) in the data field.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Modules** | **AlphaNum Only** | **JSON with AlphaNum data** | **JSON with JSON data** |
| 1 | 21 x 21 | 20 | Too small | Too small |
| 2 | 25 x 25 | 38 | Too small | Too small |
| 3 | 29 x 29 | 61 | Too small | Too small |
| 4 | 33 x 33 | 90 | Too small | Too small |
| 5 | 37 x 37 | 122 | 88 | 84 |
| 6 | 41 x 41 | 154 | 120 | 106 |
| 7 | 45 x 45 | 178 | 144 | 122 |
| 8 | 49 x 49 | 221 | 187 | 152 |
| 9 | 53 x 53 | 262 | 228 | 180 |
| 10 | 57 x 57 | 311 | 274 | 213 |
| 11 | 61 x 61 | 366 | 329 | 251 |
| 12 | 65 x 65 | 419 | 382 | 287 |
| 13 | 69 x 69 | 483 | 446 | 331 |
| 14 | 73 x 73 | 528 | 491 | 362 |
| 15 | 77 x 77 | 600 | 564 | 412 |
| 16 | 81 x 81 | 656 | 619 | 450 |
| 17 | 85 x 85 | 734 | 697 | 504 |
| 18 | 89 x 89 | 816 | 779 | 560 |
| 19 | 93 x 93 | 909 | 872 | 624 |
| 20 | 97 x 97 | 970 | 933 | 666 |
| 21 | 101 x 101 | 1035 | 999 | 711 |
| 22 | 105 x 105 | 1134 | 1097 | 779 |
| 23 | 109 x 109 | 1248 | 1211 | 857 |
| 24 | 113 x 113 | 1326 | 1289 | 911 |
| 25 | 117 x 117 | 1451 | 1415 | 997 |
| 26 | 121 x 121 | 1542 | 1505 | 1059 |
| 27 | 125 X 125 | 1637 | 1600 | 1125 |
| 28 | 129 X 129 | 1732 | 1695 | 1190 |
| 29 | 133 X 133 | 1839 | 1803 | 1264 |
| 30 | 137 x 137 | 1994 | 1957 | 1370 |
| 31 | 141 x 141 | 2113 | 2076 | 1452 |
| 32 | 145 x 145 | 2238 | 2201 | 1538 |
| 33 | 149 x 149 | 2369 | 2332 | 1628 |
| 34 | 153 x 153 | 2506 | 2469 | 1722 |
| 35 | 157 x 157 | 2632 | 2595 | 1809 |
| 36 | 161 x 161 | 2780 | 2744 | 1911 |
| 37 | 165 x 165 | 2894 | 2857 | 1989 |
| 38 | 169 x 169 | 3054 | 3017 | 2099 |
| 39 | 173 x 173 | 3220 | 3183 | 2213 |
| 40 | 177 x 177 | 3391 | 3355 | 2331 |

## Test Label Printing

Using the program ifranview (<https://www.irfanview.com/>), it is possible to scale and print all the QR codes at once.

1. Install irfanview
2. Open irfanview
3. Select File->Thumbnails
4. Using file view pane on left hand side of screen, navigate to the directory where the QR code images are located
5. Now click on the files to select the ones you wish to print
6. Select File->Print selected files as single images (batch print)
7. Select “Print Size:” “Custom”. Set the Width and Height.
   1. For the 10mm x 10mm QR codes: 1.1 cm
   2. For the 20mm x 20mm QR codes: 2.1 cm
   3. For the 50mm x 50mm QR codes: 5.3 cm
8. Now check the box for Headnote and put $D $F in the input box so that the file name is printed with the image
9. In the “Position on paper” section, select “Center horz”
10. In the “Position on paper” section, enter a “Top margin” of 5.00
11. You can go into the Printer setup and set the printer options on the printer driver (2-sided, flip, etc.)
12. See below.



# Test

The purpose of this test is to determine what the maximum amount of data that can be read with the camera at given distances, label sizes, and label densities.

## Test Setup

* The camera for each test configuration will be tested using printed QR labels and QR code reader applications that can read the QR label’s code.
* The applications will display the camera image and display the QR code.
* The QR labels will be placed in camera view by hand.
* If the QR label is decoded, the data will be displayed by the QR code reader application.
* The Vetscan Hub Mockup with the LED diffuser will be used.

## Glare with Varying LED Configurations

The glare from the Vetscan Hub Mockup’s camera light is causing glare in the captured image if the LED diffuser is not present. The LED diffuser greatly diminished this effect but did not eliminate it completely. The final picture shows the image with only ambient light from the office in Durham.

All test data in this report was collected using the Vetscan Hub Mockup with an LED diffuser.

|  |  |
| --- | --- |
| Figure 1 No Diffuser | Figure Diffuser #1 |
| No LED  Figure Camera LED covered | Text, qr code  Description automatically generated  Figure Diffuser #2 (less light) |
|  |  |

## Fixed Focus vs Autofocus

The camera module in the Vetscan Hub is an autofocus camera, meaning it has the ability by use of an actuated mechanism to change the distance from the camera that it brings things into focus. ADLink recommended that we try the camera is fixed focus to give a better experience and this test will evaluate both fixed focus and autofocus modes of operation.

For all fixed focus testing, the focal length was moved to as close to the lens as possible. This should yield the best results for the 10 mm x 10 mm QR codes but will make the 50 mm x 50 mm QR codes a challenge to get in view and still be in focus.

# Test Instructions

* For each test configuration, use the QR codes printed per the “Test Label Generation” section.
* Scan progressively denser QR codes (higher versions) until they are no longer able to be read.

## Web App

Web page: <https://www.the-qrcode-generator.com/scan>

Vetscan:

Monitor titled to max angle upwards.

ThinkPad P50 Laptop:

Distance from Web camera: ~2.25-inches

Perform the following test setup once at the beginning of testing:

1. Turn on the Vetscan Hub Mockup or Laptop that is under test.
2. Open the web site: <https://4qrcode.com/scan-qr-code.php>.
3. On the web page, click on the button labelled “Open camera”.
4. You should see an image from the unit’s camera displayed on the web page.
5. See screen capture of web page below.



For each test case:

1. Using the web page’s camera image, center the QR tag’s image in the center of the image.
2. If the QR label can be decoded, the web page will display the QR label’s data on web page.
3. Once the image is centered, wait at most 10 seconds for the web site to decode the QR label.
4. If the QR label was decoded within 10 seconds, then the test case passed, else the test case failed.

## Galaxy S10

Android app: QR & Barcode Reader - TeaCapps

Distance from camera: ~2-inches

Perform the following test setup once at the beginning of testing:

1. Open the QR & Barcode Reader – TeaCapps on the Galaxy S10.
2. See below. Left: QR code being scanned. Right: QR code decoded.

 

For each test case:

1. Move the Galaxy S10 a few centimeters above the QR code.
2. Using the camera view on the phone, center the QR tag’s image in the center of the image.
3. If the QR label can be decoded, the value will be displayed on the phone screen.
4. If the QR was decoded within 10 seconds, then the test case passed, else the test case failed.

## Python App

Python app: qr\_scanner.py

Perform the following test setup once at the beginning of testing:

1. Turn on the Vetscan Hub Mockup or the Laptop being tested.
2. In a console window, enter the command:

$ python3 qr\_scanner.py

1. A window will display the view of the camera.
2. When the QR code is decoded successfully, a box is drawn around the QR code in the window, and the JSON is decoded.



For each test case:

1. Using the displayed camera image, center the QR tag’s image in the center of the image.
2. If the QR label can be decoded, the JSON text will be displayed and a beep will sound.
3. Once the image is centered, wait at most 10 seconds for the web site to decode the QR label.
4. If the QR label was decoded within 10 seconds, then the test case passed, else the test case failed.

## 

# Test Results

The following tables show the densest QR codes (highest version) that was able to be successfully scanned with each configuration.

## Galaxy S10 Cell Phone

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimension | **Version** | **Modules** | **Max AlphaNum** | **Max JSON** |
| 10mm x 10mm | 16 | 81 x 81 | 656 | 450 |
| 20mm x 20mm | 27 | 125 X 125 | 1637 | 1125 |
| 50mm x 50mm | 26 | 121 x 121 | 1542 | 1059 |

## Laptop Web App

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimension | **Version** | **Modules** | **Max AlphaNum** | **Max JSON** |
| 10mm x 10mm | Failed |  |  |  |
| 20mm x 20mm | 6 | 41 x 41 | 154 | 106 |
| 50mm x 50mm | 17 | 85 x 85 | 734 | 504 |

## Laptop Python App

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimension | **Version** | **Modules** | **Max AlphaNum** | **Max JSON** |
| 10mm x 10mm | Failed |  |  |  |
| 20mm x 20mm | 10 | 57 x 57 | 311 | 213 |
| 50mm x 50mm | 24 | 113 x 113 | 1326 | 911 |

## Vetscan Hub Mockup Web App

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimension | **Version** | **Modules** | **Max AlphaNum** | **Max JSON** |
| 10mm x 10mm | Failed |  |  |  |
| 20mm x 20mm | 10 | 57 x 57 | 311 | 213 |
| 50mm x 50mm | 16 | 81 x 81 | 656 | 450 |

## Vetscan Hub Mockup Python App - Autofocus Off

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimension | **Version** | **Modules** | **Max AlphaNum** | **Max JSON** |
| 10mm x 10mm | 8 | 49 x 49 | 221 | 152 |
| 20mm x 20mm | 17 | 85 x 85 | 734 | 504 |
| 50mm x 50mm | 18 | 89 x 89 | 816 | 560 |

## Vetscan Hub Mockup Python App - Autofocus On

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimension | **Version** | **Modules** | **Max AlphaNum** | **Max JSON** |
| 10mm x 10mm | 7 | 45 x 45 | 178 | 122 |
| 20mm x 20mm | 17 | 85 x 85 | 734 | 504 |
| 50mm x 50mm | 29 | 133 X 133 | 1839 | 1264 |

## UPC Scanning

The same python libraries used to support scanning and decoding of QR codes also scans UPC barcodes. Barcodes on consumer products of sizes 30 mm x 15 mm, 25 mm x 10 mm, and 13 mm x 6 mm were all scanned successfully.

## “Data Matrix” decoding timing using pylibdmtx library

The decoding of “Data Matrix” labels using the Vetscan camera was tested using:

python library: pylibdmtx

python repo: <https://pypi.org/project/pylibdmtx/>

The pylibdmtx library was added to the qr\_scanner.py application. The application takes captured images from a web cam and uses the library to decode and display the data contained in a “Data Matrix” or “QR code” label.

The files are stored in GitHub QR\_code/data\_matrix\_codes

4 files were used to test the timing of the “Data Matrix” decoder:

* data\_matrix\_cdp.png
  + Qr code

    Description automatically generated
  + A generated “Data Matrix” png file.
  + The data that came from a “COMPREHENSIVE DISGNOSTIC PROFILE” consumable package.
* data\_matrix\_cpl.png
  + Qr code

    Description automatically generated
  + A generated “Data Matrix” png file.
  + The data that came from a “cPL Rapid Test” consumable package.
* data\_matrix\_cpl\_camera.png
  + Text

    Description automatically generated
  + A captured camera image file
  + Image of a “cPL Rapid Test” consumable package.
  + Package was held to minimize glare from the camera LED.
* data\_matrix\_cpl\_camera\_glare.png
  + Map

    Description automatically generated
  + A captured camera image file of a “cPL Rapid Test” consumable package.
  + Package was held to maximize glare from the camera LED.

### Laptop - “Data Matrix” decoding times

Timing results below were recorded using a ThinkPad laptop with the built in web cam.

0.08 seconds - data\_matrix\_cdp.png

0.05 seconds - data\_matrix\_cpl.png

1.86 seconds - data\_matrix\_cpl\_camera.png

1.88 seconds - data\_matrix\_cpl\_camera\_glare.png

### Vetscan Hub Mockup - “Data Matrix” decoding times

Timing results below were recorded using a Vetscan Hub Mockup with the built in web cam.

0.32 seconds - data\_matrix\_cdp.png

0.23 seconds - data\_matrix\_cpl.png

8.43 seconds - data\_matrix\_cpl\_camera.png

8.38 seconds - data\_matrix\_cpl\_camera\_glare.png

# Conclusions

## Python app versus web app

On the Vetscan Hub Mockup, the Python app was much better at decoding the labels then the web app.

## Comparison of results

Below is a comparison of the test results for each test configuration.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Configuration** | **Max Data with JSON text** | | |
| **10mm x 10mm** | **20mm x 20mm** | **50mm x 50mm** |
| Vetscan Hub Mockup Web App | Failed | 213 | 450 |
| Galaxy S10 Cell Phone | 450 | 1125 | 1059 |
| Laptop Web App | Failed | 106 | 504 |
| Laptop Python App | Failed | 262 | 1326 |
| Vetscan Hub Mockup Python App  Autofocus Off | 152 | 504 | 560 |
| Vetscan Hub Mockup Python App  Autofocus On | 122 | 504 | 1264 |

## Autofocus

The autofocus feature is slow, requiring that the QR code be held at the same distance for a few seconds before the camera is in focus.

The Autofocus did help read the higher density QR codes, but the QR code must be held very still for 2 or 3 seconds so that the slow autofocus can move the lens.

With 2 different operators, fill out the following tables to gauge the relative times it takes to scan barcodes with autofocus on and off using the python application on the Vetscan Hub Mockup with the attached LED diffuser. Perform 5 scans for each field.

### Autofocus Time to Scan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| QR Version | Size  (mm x mm) | Operator | Scan Times (sec) | Avg Scan Time (sec) |
| 5 | 10 x 10 | Bruce | 8.22 1.51 4.40 7.54 2.70 | 4.87 |
| 6 | 10 x 10 | Bruce | 6.98 2.41 2.30 2.32 24.15 | 7.63 |
| 7 | 10 x 10 | Bruce | 4.54 10.59 12.72 5.90 14.43 | 9.64 |
|  |  |  |  |  |
| 5 | 10 x 10 | Brian | 3.74 5.33 7.75 5.96 5.79 | 5.71 |
| 6 | 10 x 10 | Brian | 6.40 4.96 4.82 4.56 9.53 | 6.05 |
| 7 | 10 x 10 | Brian | 8.22 5.62 5.08 4.90 5.35 | 5.83 |
|  |  |  |  |  |
| 14 | 20 x 20 | Bruce | 3.84 5.50 2.86 2.20 6.24 | 4.13 |
| 15 | 20 x 20 | Bruce | 4.64 1.62 1.91 5.27 2.40 | 3.17 |
| 16 | 20 x 20 | Bruce | 4.37 2.95 2.66 5.75 3.65 | 3.88 |
| 17 | 20 x 20 | Bruce | 3.21 2.53 7.12 2.40 2.65 | 3.58 |
|  |  |  |  |  |
| 14 | 20 x 20 | Brian | 3.39 5.65 3.86 4.07 4.34 | 4.26 |
| 15 | 20 x 20 | Brian | 9.85 8.44 1.41 6.45 5.69 | 6.37 |
| 16 | 20 x 20 | Brian | 5.83 3.62 10.90 18.72 8.23 | 9.46 |
| 17 | 20 x 20 | Brian | 5.01 3.94 4.57 3.76 7.63 | 4.98 |
|  |  |  |  |  |
| 26 | 50 x 50 | Bruce | 19.13 9.09 7.87 2.61 2.99 | 8.34 |
| 27 | 50 x 50 | Bruce | 14.97 24.39 14.41 19.70 6.43 | 15.98 |
| 28 | 50 x 50 | Bruce | 14.81 10.92 18.14 27.58 18.67 | 18.03 |
| 29 | 50 x 50 | Bruce | 34.13 12.21 15.54 7.45 9.33 | 15.73 |
|  |  |  |  |  |
| 26 | 50 x 50 | Brian | 2.76 3.64 5.35 5.42 3.23 | 4.08 |
| 27 | 50 x 50 | Brian | 3.71 7.02 2.95 4.58 3.59 | 4.37 |
| 28 | 50 x 50 | Brian | 3.51 5.64 3.94 3.06 3.10 | 3.85 |
| 29 | 50 x 50 | Brian | 3.43 8.14 3.99 6.33 3.19 | 5.02 |

### Fixed Focus Time to Scan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| QR Version | Size  (mm x mm) | Operator | Scan Times (sec) | Avg Scan Time (sec) |
| 6 | 10 x 10 | Bruce | 2.74 2.62 2.00 1.75 3.79 | 2.58 |
| 7 | 10 x 10 | Bruce | 1.99 2.82 2.61 1.78 2.54 | 2.35 |
| 8 | 10 x 10 | Bruce | 0.46 1.80 7.93 3.16 2.34 | 3.14 |
|  |  |  |  |  |
| 6 | 10 x 10 | Brian | 5.86 1.18 1.37 1.56 3.17 | 2.63 |
| 7 | 10 x 10 | Brian | 3.80 4.04 2.33 3.06 2.39 | 3.12 |
| 8 | 10 x 10 | Brian | 6.20 7.71 4.78 4.67 3.99 | 5.47 |
|  |  |  |  |  |
| 14 | 20 x 20 | Bruce | 1.91 1.63 2.18 1.71 2.05 | 1.90 |
| 15 | 20 x 20 | Bruce | 2.64 2.39 1.90 2.45 2.59 | 2.39 |
| 16 | 20 x 20 | Bruce | 5.19 2.40 2.02 2.38 15.92 | 5.58 |
| 17 | 20 x 20 | Bruce | 3.30 0.06 2.62 2.93 2.36 | 2.25 |
|  |  |  |  |  |
| 14 | 20 x 20 | Brian | 3.23 3.66 3.08 3.01 2.07 | 3.01 |
| 15 | 20 x 20 | Brian | 3.90 2.42 2.52 2.53 2.38 | 2.75 |
| 16 | 20 x 20 | Brian | 3.48 2.06 2.57 2.50 2.11 | 2.54 |
| 17 | 20 x 20 | Brian | 1.70 1.74 2.23 3.52 2.08 | 2.25 |
|  |  |  |  |  |
| 15 | 50 x 50 | Bruce | 3.06 4.58 11.00 7.45 4.37 | 6.09 |
| 16 | 50 x 50 | Bruce | 6.11 4.35 4.69 8.06 10.48 | 6.74 |
| 17 | 50 x 50 | Bruce | 6.92 5.06 7.21 9.27 6.98 | 7.09 |
| 18 | 50 x 50 | Bruce | 8.99 4.59 10.14 11.71 9.57 | 9.00 |
|  |  |  |  |  |
| 15 | 50 x 50 | Brian | 7.88 3.91 4.16 3.50 1.75 | 4.24 |
| 16 | 50 x 50 | Brian | 6.63 2.25 4.39 2.08 3.58 | 3.79 |
| 17 | 50 x 50 | Brian | 3.88 7.51 2.57 5.90 21.57 | 8.29 |
| 18 | 50 x 50 | Brian | 5.17 7.71 25.39 3.35 6.97 | 9.72 |

### Graph of Autofocus vs Fixed Focus

The time to scan QR codes when in fixed focus mode was faster than autofocus mode for almost all test configurations.

### Autofocus Time to Scan – run with image rotated

The data below was recorded with the images rotated in the qr\_scanner.py app so that the QR code appears in the same orientation as the displayed image.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| QR Version | Size  (mm x mm) | Operator | Scan Times (sec) | Avg Scan Time (sec) |
| 5 | 10 x 10 | Bruce | 18.41 0.94 1.68 5.38 6.25 | 6.53 |
| 6 | 10 x 10 | Bruce | 12.65 1.84 2.07 6.28 2.42 | 5.05 |
| 7 | 10 x 10 | Bruce | 2.48 19.54 11.42 2.65 3.38 | 7.89 |
| 14 | 20 x 20 | Bruce | 10.73 2.22 2.16 2.01 1.53 | 3.73 |
| 15 | 20 x 20 | Bruce | 15.13 16.22 9.63 15.73 2.12 | 11.77 |
| 16 | 20 x 20 | Bruce | 17.53 7.69 10.28 2.36 6.79 | 8.93 |
| 17 | 20 x 20 | Bruce | 16.24 5.69 2.09 11.15 8.31 | 8.69 |
| 26 | 50 x 50 | Bruce | 0.95 3.46 14.81 3.67 2.92 | 5.16 |
| 27 | 50 x 50 | Bruce | 7.92 16.42 11.70 16.77 9.26 | 12.41 |
| 29 | 50 x 50 | Bruce | 15.64 2.26 3.66 5.28 2.03 | 5.77 |

### Fixed Focus Time to Scan – run with image rotated

The data below was recorded with the images rotated in the qr\_scanner.py app so that the QR code appears in the same orientation as the displayed image. This does not appear to have sped up scanning times but Bruce had become accustomed to scanning with an inverse view and it is possible this is why.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| QR Version | Size  (mm x mm) | Operator | Scan Times (sec) | Avg Scan Time (sec) |
| 6 | 10 x 10 | Bruce | 3.45 2.95 2.11 1.47 1.65 | 2.33 |
| 7 | 10 x 10 | Bruce | 4.94 2.38 2.86 2.18 5.25 | 3.52 |
| 8 | 10 x 10 | Bruce | 5.32 3.89 4.19 2.46 5.16 | 4.2 |
| 14 | 20 x 20 | Bruce | 0.49 3.25 4.00 1.75 1.74 | 2.24 |
| 15 | 20 x 20 | Bruce | 5.59 6.45 1.73 2.18 2.64 | 3.72 |
| 16 | 20 x 20 | Bruce | 2.53 2.21 3.59 2.08 3.06 | 2.69 |
| 17 | 20 x 20 | Bruce | 7.20 0.27 2.74 6.03 5.47 | 4.34 |
| 15 | 50 x 50 | Bruce | 9.85 5.35 8.17 3.80 3.67 | 6.17 |
| 16 | 50 x 50 | Bruce | 9.41 4.79 6.19 12.97 2.76 | 7.22 |
| 17 | 50 x 50 | Bruce | 5.56 31.72 41.37 2.98 3.33 | 16.99 |
| 18 | 50 x 50 | Bruce | 45.04 3.35 8.37 6.71 6.34 | 13.96 |

### Graph of Autofocus vs Fixed Focus

### Disfusser #2

Disfusser #2 was received on Aug 30 and the tests were re-run with it installed.

#### Disfusser #2 - Time to Scan

Diffuser #2 was used.

The data below was recorded using the qr\_scanner.py app.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| QR Version | Size  (mm x mm) | Operator | Scan Times (sec) | Avg Scan Time (sec) |
| 6 | 10 x 10 | Bruce | 3.82 3.93 3.24 6.80 4.87 | 4.53 |
| 7 | 10 x 10 | Bruce | 7.68 4.30 22.70 8.12 11.16 | 10.79 |
| 8 | 10 x 10 | Bruce | 23.55 7.21 12.31 10.85 5.52 | 11.89 |
| 14 | 20 x 20 | Bruce | 11.05 12.32 10.52 4.46 4.07 | 8.48 |
| 15 | 20 x 20 | Bruce | 6.04 6.33 5.35 4.91 5.73 | 5.67 |
| 16 | 20 x 20 | Bruce | 4.53 3.54 10.35 5.11 5.11 | 5.73 |
| 17 | 20 x 20 | Bruce | 6.23 4.59 9.58 12.00 10.16 | 8.51 |
| 15 | 50 x 50 | Bruce | 6.15 4.11 2.87 4.76 3.65 | 4.31 |
| 16 | 50 x 50 | Bruce | 2.40 4.20 3.37 2.92 5.32 | 3.64 |
| 17 | 50 x 50 | Bruce | 5.10 6.26 2.27 3.33 5.60 | 4.51 |
| 18 | 50 x 50 | Bruce | 3.74 2.58 3.22 4.57 3.80 | 3.59 |

### Graph of diffuser #1 vs diffuser #2

# Recommendations

## Current Camera Module

1. 10 mm x 10 mm is only viable if there is no additional data in the JSON “data” field.
   1. Recommend we do not support 10 mm x 10 mm and support 20 mm x 20 mm as smallest supported consumable QR code size.
2. If additional information in the JSON “data” field is required, 20 mm x 20 mm gives an additional 390 characters. Derating this to 250 would provide design margin.
3. 50 mm x 50 mm is possible but not necessary. Not much additional data was able to be captured, this is due to the difference in focal length needed for scanning dense 10 mm x 10 mm codes and 50 mm x 50 mm codes. Setting the focal length further away from the camera lens would allow for a better experience with 50 mm x 50 mm codes but that would reduce the amount of functionality available for 10 mm x 10 mm.
   1. We can counter this by having each analyzer send the size of the consumable that it is requesting a scan for to the Hub at scan time. This would allow us to maximize the number of characters available in each size QR code we chose to use and to also minimize the operator effort to successfully scan.
4. If additional data is needed past 250 characters, a URL could be embedded in the data field from which data could be retrieved of any size over the internet.
   1. This would impose a requirement for analyzers that needed lots of data from QR scanning would have to be connected to the internet to function.
   2. Intermediate sizes could also be used to increase data density.
5. Fixed focus is faster and takes less user effort than autofocus and should be the setting we use for the camera in production.
6. The camera is installed backwards. The captured camera images are rotated when displayed. This makes it confusing which way to move the QR code to get it into the captured image. The image will need to be flipped for final product.
7. Scanning of “Data Matrix” with the Vetscan camera and the pylibdmtx library is not practical. The decoder library is so slow that it will prevent the camera images from being updated in near real time.

## Future Improvements

1. The work and testing in this document was all accomplished using open source QR code scanning library zbar and the associated python library pylibzbar. Honeywell has a library that is commercially available that could possibly improve performance. This may especially give an advantage when reading QR codes that are slightly damaged, bent, wrinkled, etc.
2. The current camera is not the ideal camera for QR code scanning. There are dedicated QR scanning cameras that have design advantages. Camera characteristics we could attempt to improve upon include but are not limited to:
   1. Fixed focus vs autofocus
   2. Lens aperture or variable aperture control
   3. Camera resolution
   4. Laser guide
   5. LED flash configuration
3. The current LED and diffuser are not the ideal solution for illuminating the QR codes. The LED is near the camera lens, so the intense LED light always appears as glare in the image of the QR code. It would help if a wider light diffuser was used that lights the QR code a few inches from the camera lens or some other physical solution to spread the light.
4. Investigation of other technologies that could remove the need for QR codes to be scanned. NFC and RFID are used for similar applications and have some significant advantages but some challenges as well.

## Diffusers

1. See Diffuser #2 – Glare in image. The darker diffuser reduses the glare in the middle of the captured images. The reduced glare still interfers with the image decoder software and reduces the ablity of the software to decode the QR codes.
2. The glare reduction improves decoding large QR codes, but makes decoding smaller QR codes even worse. I suspect that this is due to light in the office illuminating the QR code instead of the camera’s LED.

# Revision History

Record the results of testing labels at 3 inches (76.2 mm).

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| A beta | 15 JUL 2021 | Bruce Graham | Initial work. The use of the 10 mm x 10 mm QR labels is to be resolved with initial testing. |
| 1 | 22 JUL 2021 | Brian Newberry | 1.4.1 Updated Content of label code. |
| 2 | 26 JUL 2021 | Bruce Graham | 4.5 Added test results Vetscan: 20mm  4.6 Added test results Laptop: 20mm -web app  4.7 Added test results Laptop: 20mm – Python app  4.8 Added test results Galaxy S10 cell phone: 20mm  4.9 Added test results – Vetscan: 20mm with Python app |
| 3 | 28 JUL 2021 | Bruce Graham | Condensed tables for each device.  5 Conclusions – Put the results into a table to make it easier to compare the text configurations. |
| 4 | 5 AUG 2021 | Bruce Graham | 1.7.1 Added table – QR code maximum data capacity  Simplified tables to show only the max QR code density that each test configuration could measure.  5.4 Added chart – Comparison of results |
| 5 | 6 AUG 2021 | Bruce Graham | Updated all tables with test results with QR code that used JSON with more binary data |
| 6 | 8 AUG 2021 | Brain Newberry | 5.4.2 Added tables for autofocus and fixed focus test times.  6 Added – Overall Recommendations |
| 7 | 9 AUG 2021 | Bruce Graham | 5.4.3 Added graph – Autofocus vs Fixed Focus Conclusions  6.6 Added photo to demonstrate backwards camera. |
| 8 | 13 AUG 2021 | Bruce Graham | Added tables after the displayed image was re-orientated.  Added more conclusions |
| 9 | 18 AUG 2021 | Brian Newberry | Added some structure to the recommendations, getting ready to release to wider audience. |
| 10 | 20 AUG 2021 | Bruce Graham | 4.7 Added “Data Matrix” decoding timing using pylibdmtx library |
| 11 | 2 SEP 2021 | Bruce Graham | 6.3 Added Disfussers |