

QR CODE DETECTION



SUBMITTED BY:

TEAM 16

RESHAM SHARMA (663728755)
PURVANG LAPSIWALA (662689378)

1. PROBLEM STATEMENT

Introduction to QR code:

- QR code abbreviated from Quick Response Code is the trademark for a type of matrix barcode first designed in 1994 for the automotive industry in Japan.
- QR codes are two-dimensional quick response codes whereas barcode is one dimensional data storage. Because of two dimensionality, QR code stores more amount of information. Since QR Codes encode data in the transverse and longitudinal directions, a QR code can represent data using approximately 10 times less space than a traditional barcode.
- A 101x101 QR code, with high level error correction, can hold 3248 bits of data. Most of the barcodes are 1D and hence can store a limited number of alphanumeric characters.
- A QR code uses different standardized encoding modes such as numeric, alphanumeric, byte/binary, and kanji to store data efficiently.
- The Quick Response (QR code) system became popular due to its fast readability and greater storage capacity compared to standard UPC barcodes and RFID.

Reason for fast development of QR code due to following combined disadvantages of UPSC Barcode and RFID readers.

- Materials like metals and liquid can impact the signals.
- Sometimes not accurate and reliable.
- Expensive scanning equipment.
- Implementation can be difficult.

These all limitations can be eliminated with QR code so That's why it is keep increasing in day to day usage.

Why to do QR code Scanner:

Reason for choosing QR code scanner because of following advantages of QR code.

- A QR code can store higher amount of data in a smaller surface area compared to other source and that is why it is gaining more popularity where it can be application.
- A QR code is more secure than other sources such as barcode.
- QR codes are trackable.
- Easy way to send mobile users to online content.
- QR codes are also more difficult to decrypt or hack into as compared to a RFID tag or a barcode making them a much more secure option.
- QR codes can be cost effective compared to RFIDs.
- QR codes need not be physically printed on a card (like RFID) but can be downloaded on a mobile device or an app which would make it safer and lesser chances of losing it.

Applications

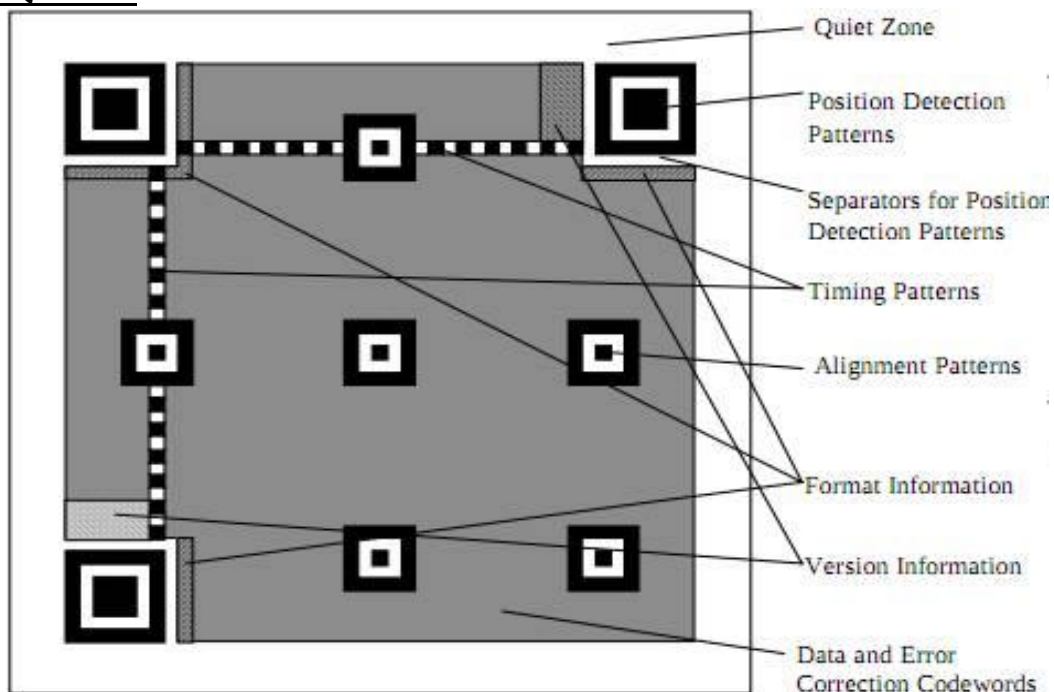
QR code is being used many places for wide variety of reason depending upon the application it is used.

- **QR code payment:** QR code can be used to make payment. One of the most popular app WeChat is using QR code method to make payments.
- **URLs:** QR code can be used as an alternative for an URLs. If you don't have browser and don't want to type long URLs, one can scan QR code and can get easily access of website.
- **Website login:** One of its application is login process for WhatsApp web. Data of QR code keep changing if you don't scan within certain period as it has lot of data to generate code.
- **Wi-Fi network login:** By scanning QR code, one can also login in to any wi-fi network without typing password.
- **Product detailing:** QR codes are used for providing information about when, how, and where the product was created. Vineyards have opted for this strategy to allow people to get in touch with them and buy more of the similar wine that they had at the restaurant.
- **Creative Ads for Mobile Applications:** Companies are creating creative ads for apps, allowing customers to act directly from the app. This enables businesses to attract customers using ad creativity through easily understandable and appealing promotion of the apps. These QR codes can be created using an online QR code generator as well.
- **Security entry method:** QR codes are also used in many offices and industries for entry purpose.
- **Medical Prescriptions:** Entry of the contents of prescriptions is made smoother through the reading QR codes printed on prescriptions.
- **Traceability:** By scanning the QR Code of each individual package before it is put into a cardboard box and printing a packaging label when the set number of lots is reached, traceability can be achieved.

With all these pros and wide variety of applications keep in my mind, we have decided to make QR code scanner system which detects the QR code in a live feed or an image, detect it and display the output.

- The system would be capable of detecting when a QR code is in the field of view of the camera, put a box around it and possibly decrypt it as well.
- We are trying to achieve a 100% accuracy in detection under any lighting conditions. We would like the system to able to detect correctly under varying brightness.

Anatomy of a QR code



A QR code is made up of four main parts:

Finder patterns: These are the big black/white/black squares on the three corners on the QR code. These help identify the presence of a QR code in an image and its orientation. These are made such that they can be detected fast.

Alignment patterns: These are smaller than finder patterns and help straighten out a QR code drawn on a curved surface. The larger a code, the more alignment patterns it'll have.

Timing pattern: These are alternating black/white modules on the QR code. The idea is to help figure out the data grid accurately.

The actual data: The blacks/whites form bits. Groups of 8 such modules make one byte. You could combine 16 modules to get Unicode data.

2. ALGORITHM:

The algorithm was based on feature detection and some basic image processing. The feature detection used was detecting the location of the three Finder's pattern and eventually detecting the location of the entire QR code.

One of the key reasons to use Finder's pattern as the key feature because it is something which is highly unique to a QR code. There are several other algorithms to check for a QR code like edge detection but the drawback with these algorithms is that they generally get confused with some other similar black and white boxes in the background or foreground.

The finder's pattern has a very unique ratio of 1:1:3:1:1 (widths of dark-light-dark-light-dark) which isn't common to find in any other object. Another major advantage of using this particular ratio as the key feature was that this ratio is constant irrespective of the angle at which the QR code is located.

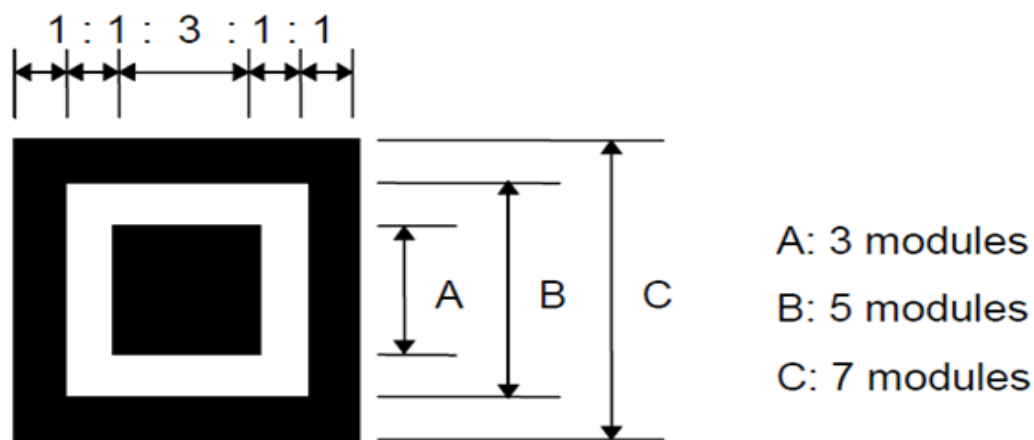


Figure 1: Finder's Pattern Ratio

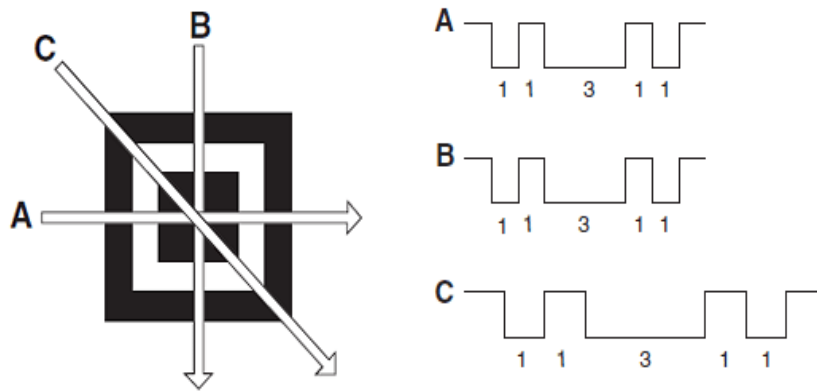
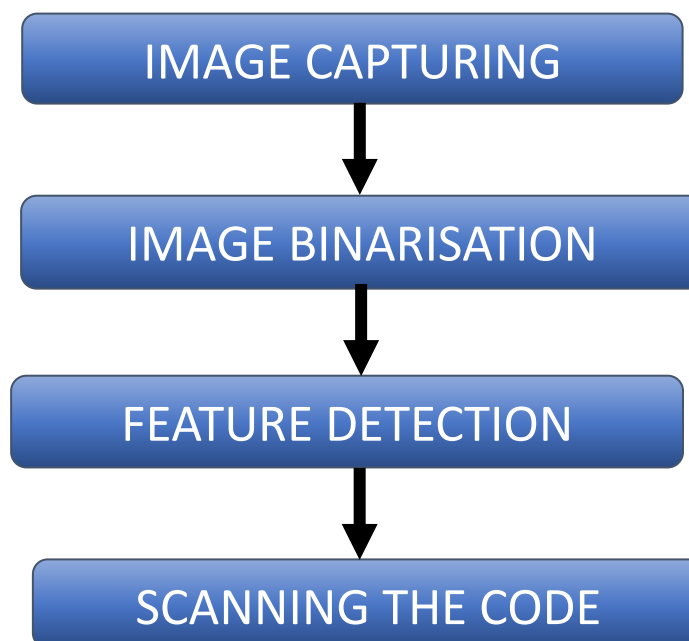


Figure 2: Ratio remains same at any angle

The entire algorithm can be divided in the following four steps:



2.1 IMAGE CAPTURING:

Image capturing was done in 2 ways:

1. STILL IMAGES:

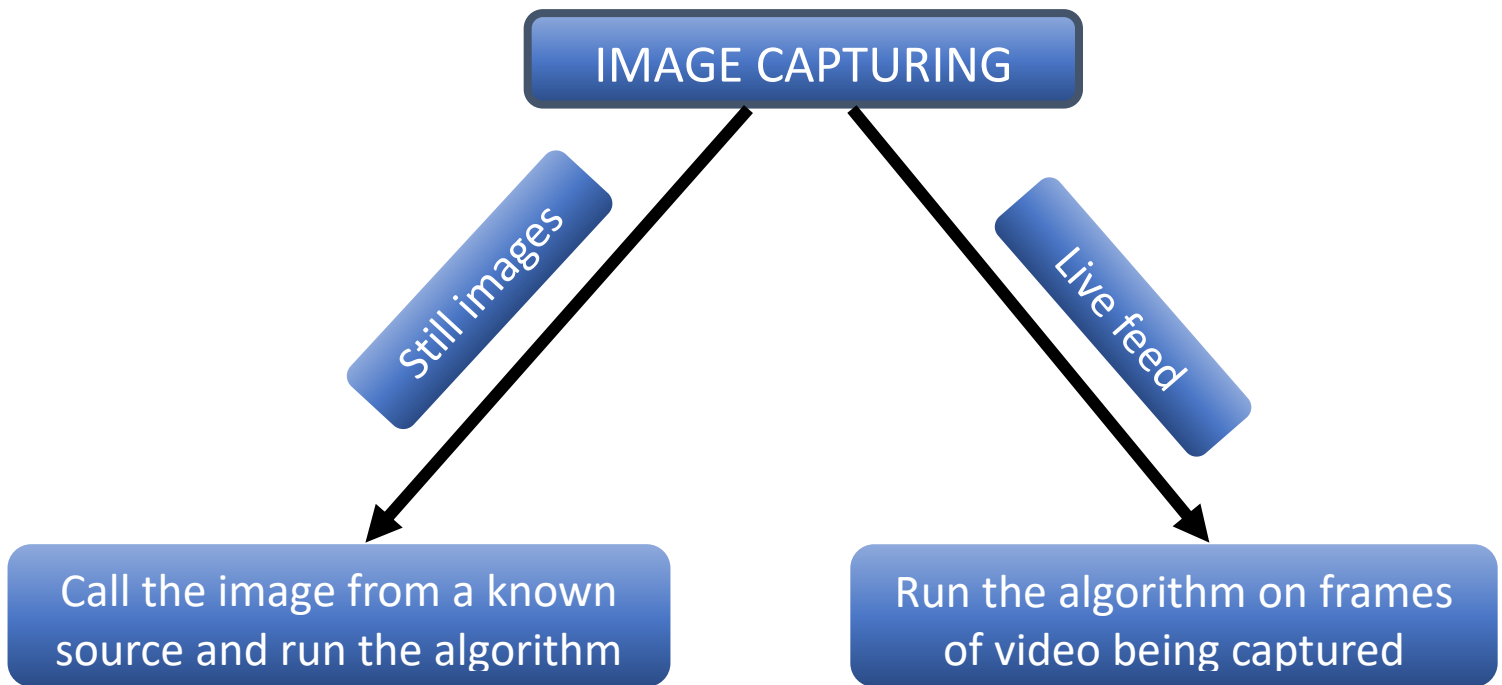
- QR code detection was first tried on still images in the system. For this couple of images under various lightening conditions were preloaded in the system and then called in directly or using a function.

2. LIVE FEED:

- QR code detection was also tested on live video feed. This feed was received from a USB camera (can also use the laptop in built camera).
- For this purpose, the feed taken as input in form of feeds. To understand it in simpler says, we can say the video was converted into a series of still images.
- The frame rate or the speed at which the frames were taken depended on the kind of application. If an application requires a higher efficiency (like security system), the frame rate was supposed to

be higher whereas in a low security application or where we have the pre-requisite knowledge of when a QR code can appear, the frame rate can be much slower.

- The algorithm was then applied on these frames to detect a QR Code.

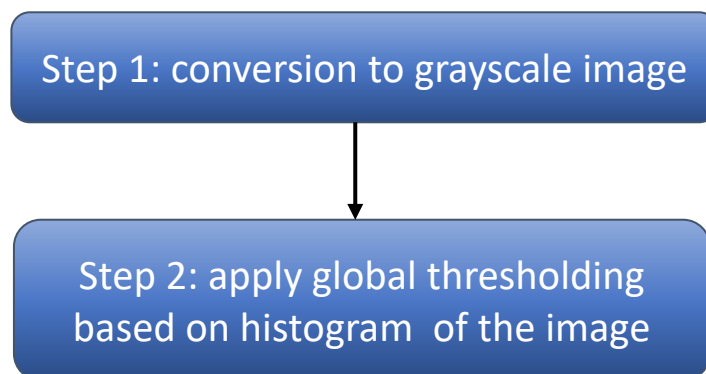


2.2 IMAGE BINARISATION:

Second step in the algorithm included doing few basic pre-processing steps of image processing.

STEP 1: The image was converted to a gray scale image. This is done for various reasons but mainly for the signal to noise conversion and also for the simplicity of understanding. Complexity of code also reduced by a great content and also increases the processing speed.

STEP 2: The image was then converted to a black and white image by thresholding. The threshold level was determined by applying global thresholding on the histogram of the image. This procedure of choosing the threshold value was done internally by the system.



2.3 FEATURE DETECTION:

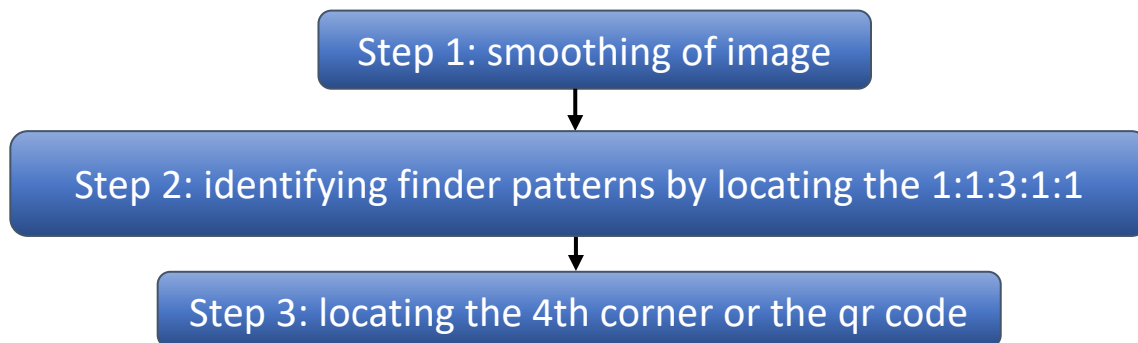
The third step of algorithm was the major part of algorithm. This involved finding the location of Finder's Pattern in the following steps:

STEP 1: The image was scanned row wise from top left and the data was continuously stored in an array.

STEP 2: The aim was to find 7 modules of the Finder's ratio. Hence, the algorithm was written in a way to automatically erase any data in the array once it reads more than 7 modules.

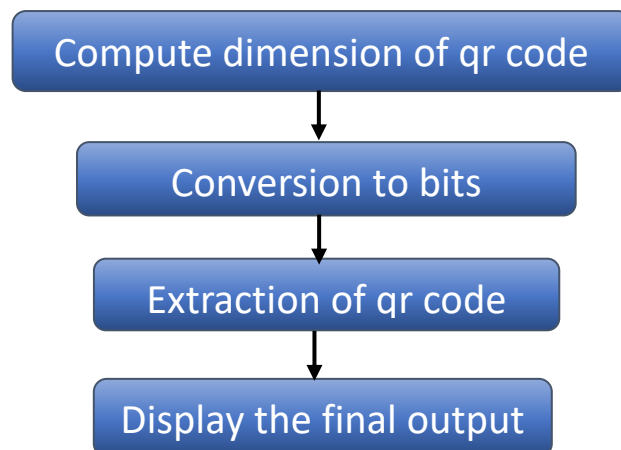
STEP 3: If a particular set of data in the array was of the particular ratio, that particular area of the image was detected as a location of the Finder pattern and the algorithm continued. Same was done till 3 such finder patterns were detected.

STEP 4: Using the location of the three finder's pattern previously found, the fourth corner of the QR code was approximated and the total area in the image was then detected as a QR code. A green box was used to show the same.



2.4 QR CODE DECRYPTION:

In order to decrypt the found QR code, the dimension of QR code was detected. The knowledge of dimension helped in estimating the number of bits that the QR code contains and how big this data is. The black and white content of the QR code was then converted to bits. These bits were then translated to a user-friendly image.



3. IMPLEMENTATION:

Materials Used in Project:

- **SOFTWARE:**
 - Python 3.6.
- **HARDWARE:**
 - Raspberry Pi, which runs on Raspbian. Reason for using raspberry pie is that ZBar has some compatibility issues with windows due to some inbuilt bugs and it was faced while working.
 - OTG Cable
- **LIBRARIES:**
 - OpenCV, (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision.
 - ZBar, which is an open source software suite for reading bar codes from various sources.

4. INPUTS:

- The dataset used to test the algorithm was created on our own and was taken by a Samsung Galaxy S8.
- The images were taken under a length of varying conditions. We had tried to incorporate as many variations as we could. The lightening conditions were changed by changing the amount of light in the room.
- We had also pre-processed the images in MATLAB in order to get extreme bright and extreme dark conditions.
- The images were taken at different times of the day also and checked. This helped in not only incorporating different level of brightness but also the shadow problem.

5. TESTING AND PERFORMANCE MEASUREMENT:

- The testing was done on self-taken images. No online dataset or library datasets were used. This was done to incorporate images under lighting conditions of our choices. This gave us the freedom to test the algorithm under multiple conditions like extremely low intensity images, varying brightness, shadows in the images etc.
- For the live feed, not only was the testing done under various lightening conditions but also the QR code was placed at varying distances from the camera. This enabled us to check the accuracy of the algorithm in more depth.
- Performance measurement was done on the bases of true positive rate obtained. This incorporated the correct number of QR code identifications as well as correct decryption of the QR code.
- We found the performance rate to be pretty high. The algorithm worked perfectly under almost every lightening conditions. The condition where the algorithm failed was in extremely bright images where the intensity of the image was high to the point of being whitewashed.
- In case of the live feed also the Total Positive Rate was (TPR) was almost 100%. It worked perfectly under all conditions. In the case where the QR code was extremely far from the camera (thus, taking up a very small area of the image), the detection was done correctly but the decryption was not always correct.

6. PRELIMINARY RESULTS:

The results obtained were as follows:

I. RESULTS OF QR DETECTION IN STILL IMAGES:

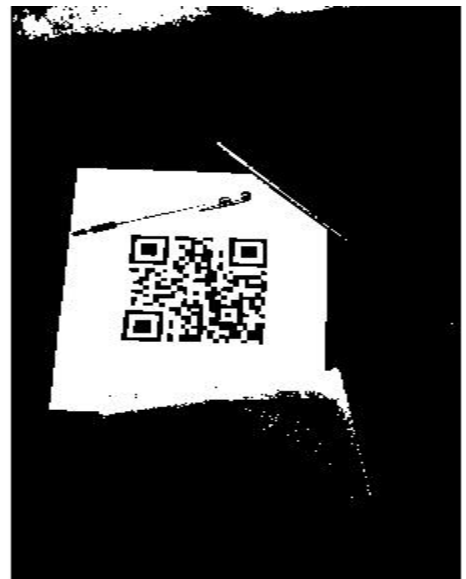
STEP 1. IMAGE CAPTURING:



STEP 2: IMAGE BINARISATION:



Gray scaled Image



Thresholded image

STEP 3: FEATURE DETECTION:



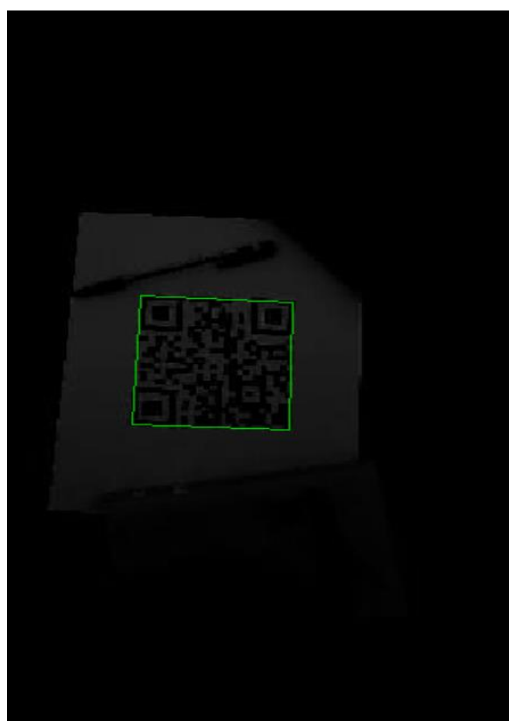
Regular conditions



Dark environment



Bright environment




Extreme dark conditions



Extreme bright conditions

STEP 4: QR CODE DECRYPTION:



QR-Code:http://www.qrstuff.com

II. QR DETECTION IN LIVE FEED:

<https://youtu.be/RyAwdSQV3aQ>

ACCURACY RATE:

1. Under normal conditions: 95% - failed when the QR code was too far from the camera.
2. Under dark conditions: 80% - Worked well till a great extent except when it was too dark and too far.
3. Under Extreme bright conditions: 75%- Worked when too bright and too far.

7. CHALLENGES:

I. HARDWARE ISSUES:

- One of the hardware issues was creating extremely dark and extremely bright environments. There was only so much that we could alter in the lightening conditions. We used MATLAB to increase and decrease the brightness to huge extents of the images to test our algorithm.
- Another hardware that took a bit of time and practice to create varying brightness in our images and live feed. We could overcome it in still images but not up-to the extent that we wanted in the live feed.

II. SOFTWARE ISSUES:

- A major blocker which we faced was using Z bar in windows. Z bar library doesn't have a completely bug free version out for Windows yet and they have only released a test version as of now. Because of this, there were a lot of unexpected errors which we were getting and couldn't solve. Also, the same algorithm would work perfectly fine sometimes and completely crash other times because the Z bar wasn't getting called in properly. Due to this, we had to shift to working on a raspberry pi half way through the project. Our other options were using Linux or re writing the code in C++. But no one in team had a Linux based system nor were we very comfortable with C++.
- Also, our algorithm took a bit of time to get perfect. The reading of modules in array and identification of it as a Finder's pattern wasn't always happening correctly and gave us false results. Hence, we spent a bit of time in tweaking and perfecting the code.
- Another issue was deciding the frame rate in case of the live feed. It took a bit of trial and error procedure to narrow it down to the required frame rate.
- We also spent a bit of time in the beginning researching about various libraries, their functionalities and how we can use them. We also looked at various examples online available of QR code detection able.

8. FUTURE SCOPE FOR PROJECT:

- QR code decryption algorithm doesn't give 100% correct output when the amount of data in QR code is high.
- Must be very particular while designing it. Cannot have any text near it, QR module cannot be distorted or any overlaid texts on the QR code.
- This might give a different result if the QR code is inverted or mirrored.

9. REFERENCES:

- https://www.researchgate.net/publication/290325686_QR_code_detection_and_recognition_algorithm_for_industrial_control (QR code detection and recognition algorithm for industrial control - N.-Z. Liu, J. Su, H. Sun)
- https://en.wikipedia.org/wiki/QR_code#Risks (Basics of QR code)
- <https://pypi.org/project/zbar-py/> (Basics of Z-bar library)
- <https://opencv.org/> (Basics of OpenCV library)
- <https://ieeexplore.ieee.org/abstract/document/4597299> (Recognition of QR Code with mobile phones- Yue Liu ; Ju Yang ; Mingjun Liu)
- <https://dl.acm.org/citation.cfm?id=2448548> (Fast detection and recognition of QR codes in high-resolution images - Istvan Szentandrasei, Adam Herout, Marketa Dubska)

NOTE: FOLLOWING LIBRARIES NEED TO BE INSTALLED FOR THE CODE:

1. OpenCV

2. ZBar