Visual Links

Sri Rama Shanmukh Krishna Sai Nanduri

Vijay Kumar Badugu

Venkata Krishna Sailesh Bommisetti

Sai Sindhur Malleni

SPRING 2015

NORTH CAROLINA STATE UNIVERSITY

RALEIGH

ABSTRACT

Android, the most widely used mobile operating system today, has taken the world by storm with the power, versatility and the reach it provides to millions of users and thousands of developers around the world. For our project this semester, we have chosen to build an android application which takes advantage of the various software and hardware features provided by present day Android phone but with the networking at its core. The main aim of the project is to transfer data between two android devices by merely using the devices' screens and cameras. This kind of a "visual link" is only feasible when both the devices are in each other's line of sight (LOS). The project description on the course website talks about a rudimentary way of implementing this visual link in a binary fashion by dimming and lighting up the screen. It also hinted at exploring QR codes. After going through the current literature in the field and exploring several other possibilities such as barcodes, we zeroed in on QR codes for use in our project. The big picture is simple, the user on one device inputs a message in a text field and then QR codes are generated on a sentence by sentence basis by the sender and the receiver scans these using its front camera, extracts each of the sentences and finally displays the message. The main issue to tackle in this design is the synchronization between the sender and the designer. The receiver needs to know where the message starts and ends correctly in order to successfully decode the codes. Initially a timer based approach was used in which the sender outputs a QR code and waits for 5 seconds before sending the next one. The receiver decodes the QR code within the 5 seconds and appends each of the sentences as strings before displaying the final output as a string. A QR Code for the word "start" was used as indicator for the receiver to know that a sequence was incoming and the next QR Code sent by the sender was the total number of sentences in the message so that the receiver knows how many QR codes it has to scan as each QR code corresponds to a sentence. After the receiver scans the required number of QR codes, the final message is displayed on the screen as a result activity. With synchronization tackled, orientation between the two devices and environment conditions pose threats to the "visual link" and later sections present details about them.

INTRODUCTION

The Android mobile OS which has Linux at its very core, provides a set of libraries and an application framework upon which developers can create new apps aimed at providing productivity or fun or both as well. Before Android came, a phone was something which could only be used to make calls and even a smart phone provided certain additional features such as email and internet access. But Android has taken the definition of a smart phone to a whole new level with its UI, connectivity, storage, media support and multitasking.

QR code which has its beginnings in the automotive industry in Japan has won appreciation all across the world because of its fast readability and good capacity compared to 1-D and 2-D barcodes. "The QR Code is a 2-D matrix code that conveys information not by the size and position of bars and spaces in a single (horizontal) dimension, but by the arrangement of its dark and light elements, called “modules,” in columns and rows, i.e. in both the horizontal and vertical directions. Each dark or light module of a QR Code symbol—a specific instance of a code—represents a 0 or 1, thus making it machine intelligible.”[1]

Since the QR codes are 2 dimensional, they make use of both the length and breadth components to package more data effectively in lesser space. There are several modules in the QR code that improve reading performance and provide features such as symbol alignment, error correction and distortion compensation. There is also a required “quiet zone,” a four-module wide buffer area containing no data, to ensure that any surrounding markings are not mistaken for a QR code. All these features made us choose QR codes over barcodes for implementing our visual link. The greater capacity of QR codes helps us improve throughput of the link which seems to be the major bottleneck.

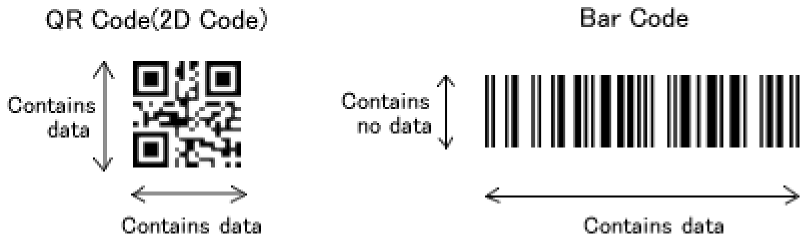


Figure 1: Difference between a QR Code and 1-D Barcode

Hence, the android app creates some sort of a wireless communication system which leverages the features of the visible spectrum. The major advantage of having this sort of a visual communication system would arise from the fact that it is highly directional, secure and interference free [2]. Also, since the app communicates wirelessly but it doesn’t use the radio spectrum, it is immune to attacks such as frequency jamming but also it is heavily dependent on the ambient light.

This kind of an android app which communicates through a “visual link” can have real world applications such as metadata exchange and opportunistic data exchange between smart phones close by. Such exchange of data can help in real time transfer of data about environment conditions and other variables. A more advanced application would be its use in named data networking where smart phones use these visual links to get information from each other based on the name of the data. Further extensions to this concept can be made with the introduction of multi-hop visual links. Visual links can also find applications in indoor WLANs and vehicular ad-hoc networks[2]. Apart from these serious applications, android apps based on QR code communication can have some fun applications when used as a “chat messenger” service.

There are a number of applications on the Google Playstore which can be used to scan QR codes and also generate QR codes. But we haven’t come across one which combines both these features in the desired manner to achieve real time communication while keeping in mind ambient conditions, synchronization and feedback. Our major contribution to this field would be the inclusion of a feedback based mechanism to achieve synchronization between the sender and the receiver. Also, having worked on java and android for the first time we discovered the ease with which applications can be developed using the pre-existing development framework and interfaces. We have also come to the conclusion that orientation is the primary hindrance.

RELATED WORK

A lot of research has been going on visible light communication using the cameras of the smartphones by encoding the information into QR codes. An important feature in our app is the generation of QR code of the text that is given. So for that we have read IEEE papers, some of them like QR code Generator[3] and Recognition of QR code with mobile phones[4] where the former gives information of the error correction and the code structure of the QR codes an the latter talks about the flow used by the image processing system present in the smartphone to be able to locate, segment and decode the QRcode. There are numerous papers in the field of visual communication but most of them have been focusing extensively on advanced signal processing techniques.

Scansfer is an android app using which we can upload transfer a file and the app generates QR code for the file that is uploaded , then we can scan the QR code generated using the front camera of our smartphone and download it . We got good clarity about our project after reading the functionality of this app, but the scansfer app does not check if the transferred file is not manipulated in any way when it has been downloaded , which we have added in our app we check if the text that is received is same as the one that is sent and this is done by using a feedback mechanism

PROBLEM FORMULATION AND MODEL

Figure 2: Two Android Phones with screens facing each other

The problems needed to be tackled in several distinct but interrelated steps. Going through the problem description, the initial idea that struck us was to use the front cameras and screens of both the devices, have one go into the sender mode and the other into receiver mode, generate QR codes for each sentence and decode them on the receiver end and then to reconstruct the entire message. We broke up the entire problem into small chunks as follows:

- Making an Android App with the MainAcitivity that lets a user chose whether he /she wants to be a sender or a receiver of the message
- Generating QR code using the existing libraries
- Displaying QR code on the screen
- Using a camera on the receiver side to scan the QR code
- Decode the QR code
- Ensure synchronization between the sender and receiver
- Provision a 2-way feedback mechanism to ensure reliable and timely communication between the sender and receiver sub-modules
- Splitting the screen space in the receiver and the sender into two to facilitate the above mentioned feedback.
- Tackling issues such as ambient light, blur and line of sight.

The sequence of steps we followed to arrive at the final solution is as follows:

1. Since the android app is at the core of the solution and all the team members were using java for the first time, we took an ample amount of time to familiarize ourselves with the nitty-gritty of the language.
2. We had an option to choose between eclipse and Android Studio but finally went with the latter as it comes with a whole lot of integrated features that make app development lucid.
3. As an initial endeavor we developed a simple app, which calls a barcode scanner app(installed as a separate app from Playstore) using intent mechanism.
4. Later we used an open source encoder app we found on github to generate a QR code, but the app sourced the QR code from a website and displayed it on the screen using an intent mechanism as said before.
5. We then came to the stage of importing the ZXing library for using within our project. The ZXing library is a library which supports decoding and generating of barcodes (like QR Code, PDF 417, EAN, UPC, Aztec, Data Matrix, Codabar) within image[5].
6. We went through related part of source code obtained from github[6] for a QR encoder and decoder and reused some of the modules for integration into our project.
7. As a next step, we developed an android application with a MainAcitvity which allows a device to go into either the sender mode or receiver mode.
8. For synchronizing the sender and receiver, we used a timer to ensure that the sender generates the QR codes in intervals of 5 seconds and the receiver is synced with the sender by generating an initial QR code for the string “start” which acts an indicator for the start of the message. The next QR code generated by the sender gives the number of strings being transmitted so that the receiver knows how many times it needs to scan to get the message in its entirety. Additionally, an important feature of the receiver to consider is that it uses the cam to keep clicking pictures of the QR code generated by the sender until it sees a new QR code. This leads to a failure if two strings that are same in every way are sent in succession.

9. The above implementation fails if the receiver misses a code before the next code is generated by the sender. Hence a real time reliable, 2-way feedback system was conceptualized and implemented.
10. For the feedback mechanism mentioned above, the screen on sender and receiver needs to be fragmented in such a way that one part is the sender generates the code and the other part is a masked camera display with a rectangular area for code reception that waits for the input from the receiver to see if the data was indeed correctly received. The receiver activity also contains two fragments, one of which is used to decode the code generated by the sender and the other to reconstruct the received message into a QR code to provide as feedback to the sender.

Even after spending a considerable amount of time, fragments didn't materialize for us and we had to move on by using ViewfinderView class which extends the View class and in which a ImageView is overlaid on top of the camera preview. This view adds the viewfinder rectangle and partial transparency outside it. We moved this rectangle to the top portion of this layout and used the remaining space display generated QR codes. Hence we were able to implement the feedback mechanism by both generating and receiving QR codes irrespective of whether or not the device is in sender or receiver mode.

Based on the modeling and design of our problem we implemented several classes and methods. A few of them have been discussed below and some insights have been given into their functionality.

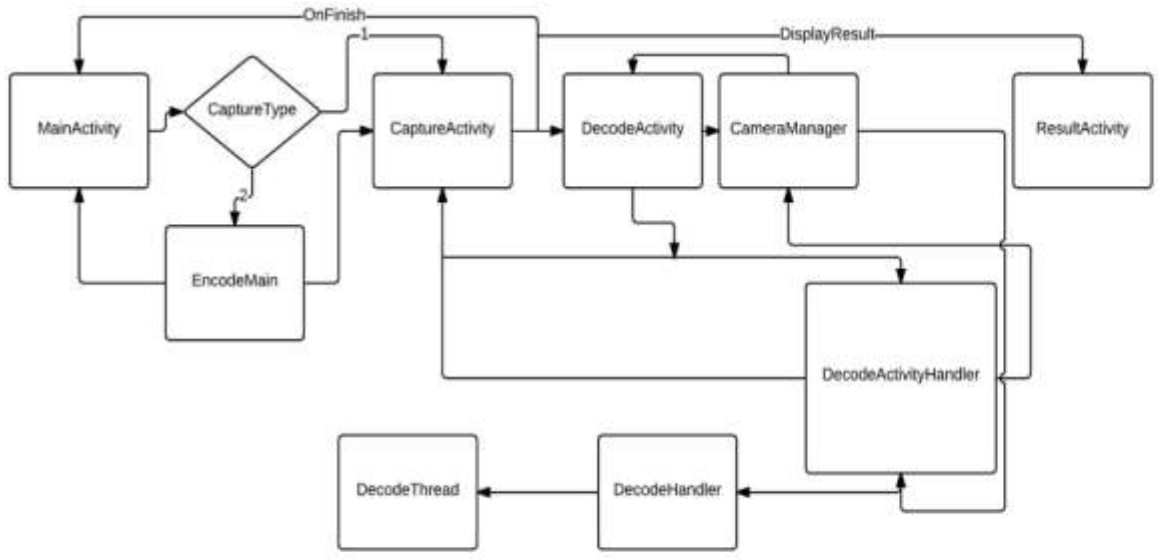


Figure 3: A flow diagram depicting interaction between different classes

Description of some important classes and methods used in our project:

MainActivity:

QRChat application has two modes of operation.One is Encode mode(Send data!!) and other Decode mode(Receive data!!).

MainActivity.java is the home class with two buttons.

- On clicking Send data!! button, callCaptureQR is called.
- On clicking Receive data!! button, callEncodeQR is called.

callCaptureQR() This will open CaptureActivity which starts the decode mode with CaptureType equal to 1.

callEncodeQR() This will open EncodeMainActivity which takes care of send mode.

CaptureActivity:

This class does the core functionality of the application. It extends DecoderActivity class. Based on the way it is called, it works differently for send and receive modes.

onCreate() method, if called from EncodeMainActivity has 'MyEncoder' set, which will trigger the display of QRCode which encodes "Start" using changeImage() method. This marks the beginning of the communication. If MyEncoder is not set, it means onCreate() is called from MainActivity and the decode mode is used. So, CaptureType is set to 1.

captureType1() is used to access CaptureType from other classes.

Initialize() is called from onCreate() method and it is used to get the data from EncodeMainActivity which passes the message through intent. It splits the data into sentences and sets the number NoterMax to the number of QR codes to be communicated between sender and receiver.

DisplayMethod() is called from DecodeActivityHandler class, when decoding is successful. If Noter is 1, it displays QRCode encoding the number of sentences to be displayed. While Noter is less than NoterMax it displays each sentence in a QRCode and displays it.

onKeyDown() calls the MainActivity when Back button is pressed.

handleDecode() is called from DecodeActivityHandler class when decoding is successful. This returns the string that is obtained from decoding the QR code.

showScanner() sets up the display of the Scan area and the mask area around it.

getResult() is called to display the final result obtained from successfully decoding the information.

changeImage() is used to display QR code which encodes the string passed to it. Usually called to display any data or Start or the count.

onResume(), onPause() and onDestroy() call the corresponding methods in DecoderActivity class which is the super class of CaptureActivity.

DecoderActivity:

This class extends Activity class and implements IDecoderActivity interface methods.

onResume() method initializes the CameraManager and the ViewfinderView which corresponds to the layout of the DecoderActivity class. It gets the surfaceview for the class and also initializes the camera.

onPause() method closes any handler if opened for the activity, closes the camera and removes any callback methods to the surfaceview.

surfaceCreated() method initializes the camera if a surface already exists.

initCamera() method opens a cameraManager driver and creates a handler if none already exists.

changeImage(), Initialize(), DisplayMethod(), CaptureType1() are just initialized and then implemented in CaptureActivity.

DecodeActivityHandler:

This class handles all the messaging which comprises the state machine for captureActivity class.

DecoderActivityHandler() constructor copies the values of the activity used, cameraManager and gets the decoding format from the passing activity. It initializes the camera and also starts a decodeThread.

***handleMessage*()** is called when a message is sent to the DecodeActivityHandler. The switch statement handles the type of message received.

- auto_focus: Requests the cameraManager for AutoFocus.
- restart_preview: Starts the camera display and starts the preview.
- decode_succeeded: Gets the data from message and decodes the string that was encoded. If in receive mode, checks if Initializer is not set, waits for the Start QRCode to mark the beginning of communication and initializer is set when Start is received. Then it displays the Start QRCode and again requests for the next frame. When a new frame is received, if count is not received, it throws an exception screen, after the count is received it decodes the data from QRCodes and stores them in finalResult string. If in the send mode, it calls displaymethod() to get the next qrcode to be displayed.
- decode_failed: If decode fails, it requests another frame to decode.
- return_scan_result: Returns the result, not handled in our case.

quitSynchronously() method stops display, closes the thread and the handler associated with it.

restartPreviewAndDecode() method starts the camera sets the state to preview, requests focus and preview from cameraManager. It also gets viewfinder from the activity.

isNumeric() method checks if the string passed is an integer or not.

We used some methods of DecodeThread, DecodeHandler and CameraManager classes as a library.

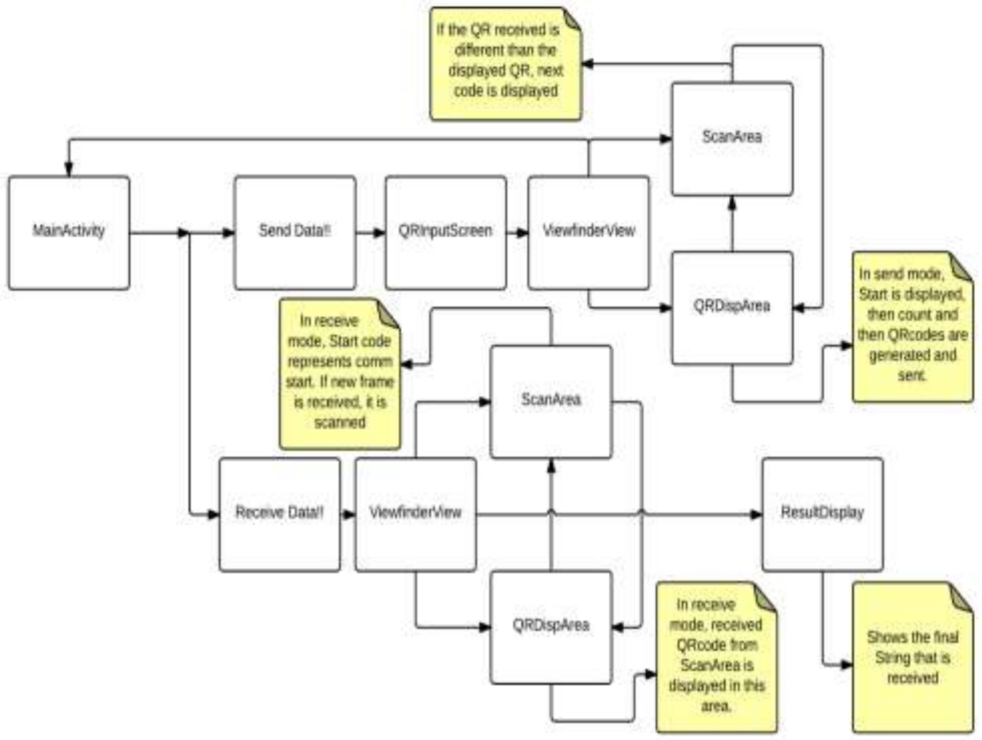

If the QR received is different than the displayed QR, next code is displayed
ScanArea
MainActivity
Send Data!!
QRInputScreen
ViewfinderView
QRDispArea
In send mode, Start is displayed, then count and then QRcodes are generated and sent.
In receive mode, Start code represents comm start. If new frame is received, it is scanned
ScanArea
Receive Data!!
ViewfinderView
ResultDisplay
QRDispArea
In receive mode, received QRcode from ScanArea is displayed in this area.
Shows the final String that is received

Figure 4: Flow Diagram showing the Activity Transitions

EXPERIMENTAL RESULTS

We have tested our application on the test Android devices provided and have obtained satisfactory performance. The feedback mechanism also proved to be extremely useful in achieving synchronization between the sender and the receiver. We are attaching some screenshots to depict the results

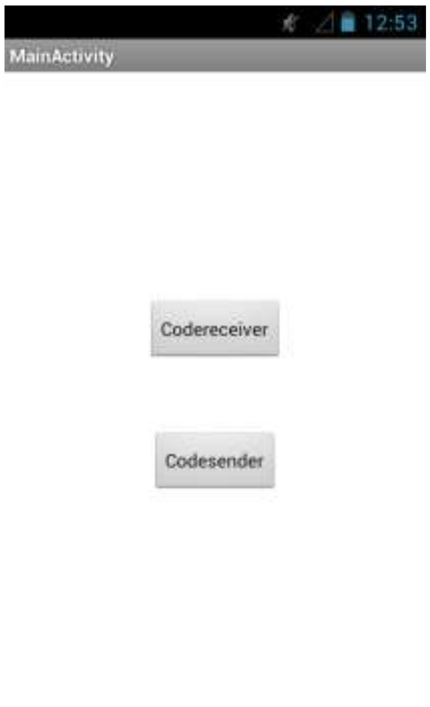
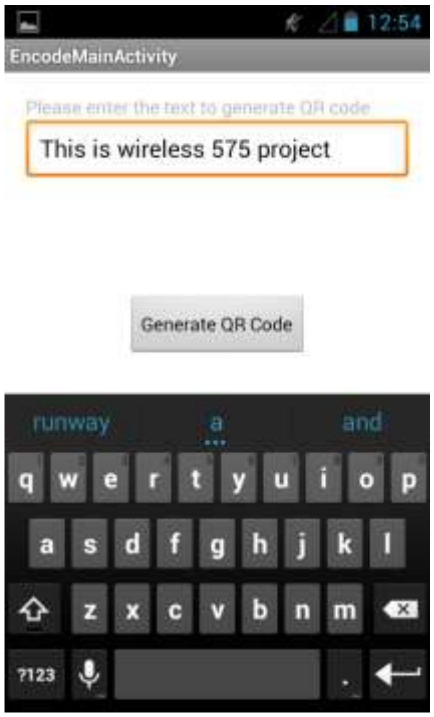


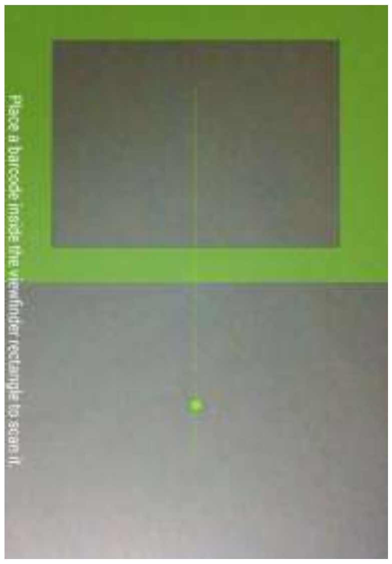
Figure 5: Home Screen



Please enter the text to generate QR code

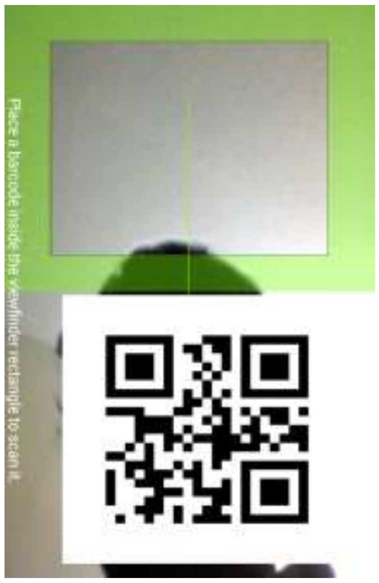
This is wireless 575 project

Figure 6:Textview with message



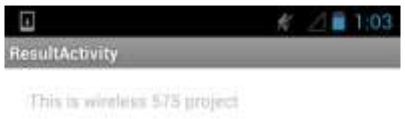
Place a barcode inside the viewfinder rectangle to scan it

Figure 7: ViewFinder for scanning



Place a barcode inside the viewfinder rectangle to scan it

Figure 8: Feedback



This is wireless 575 project

Figure 9: Result Display

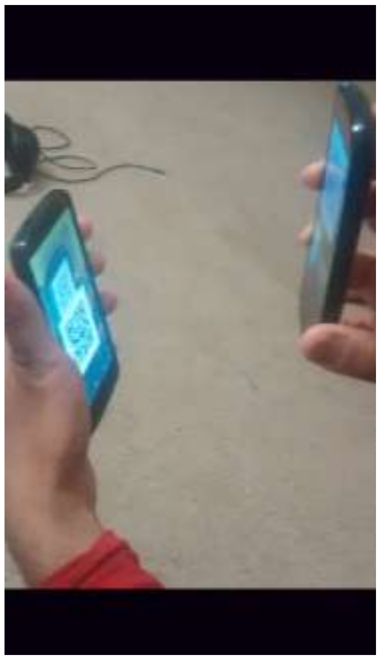


Figure 10: Two devices communicating

ANALYSIS

The major consideration in this project would be the throughput obtained by using this visual link. Since, this is an unconventional mode of communication; the throughput is very low compared to regular wireless communication. On experimenting and taking measurements we found that on observing good orientation practices we were able to obtain a throughput of 690 bps considering a 15% redundancy in data added by QR code generator for error-detection and correction. However, if we neglect the first two QR codes generated by the sender which are used for synchronization, the useful throughput of 493 bps was obtained. These readings were obtained by sending messages from the sender to the receiver and measuring the total time taken for the transfer of 5 messages (2 additional start and count messages) of capacity 174 bytes each. The time taken was found to be 12 seconds for this transfer. On having a bad orientation the throughput dropped drastically owing to the huge loss in time for re-orienting properly.

PROBLEMS FACED

The major problem we faced was ensuring synchronization between the two devices and provisioning a two way feedback. If a time based approach without feedback is used in which the sender waits for a predetermined amount of time, even if the receiver fails to decode one of the QR codes correctly, the sender moves on to the next one which results in the loss of a word or a sentence. A feedback was considered necessary for reliable communication even though its negative effects on the overall throughput were foreseen.

The feedback mechanism we implemented works as follows:

On reading and successfully decoding the QR code, the receiver reconstructs the received message into a QR code and displays it on the second half of the screen. The sender uses the camera on the other half of its screen to read the QR code generated by the receiver and decodes it to match and see if it is the same as the code it transmitted. If both the strings are the same the transmitter moves on to send the next string.

Also, we hit a deadlock when we tried to use fragments for splitting the activity screen and employing feedback. As discussed previously we overcame this by using a ViewfinderView class which extends the View class and in which an Imageview is overlaid on top of the camera preview. It adds the viewfinder rectangle and partial transparency outside it. We moved this rectangle to the top portion of this layout and used the remaining space display generated QR codes.

Now, after using the above mentioned mechanism and taking some care regarding orientation, the app seems to work fine. But there is still a loophole which we plan to eliminate by the time we give the demo. The sender sees the code it generated in the view generated by the front cam of the receiver and presumes it for feedback and keeps moving to the next string while the receiver keeps waiting for the first string. We plan to eliminate this by adding a small string to the received string before the receiver generates the QR code as feedback and hence the sender expects the feedback as “sent message+string” and on receiving the code corresponding to this and proceeds to the next string.

CONCLUSION AND FUTURE WORK

Hence, we have been able to take our first leap into the world of Android programming by implementing this project on visual links. Visual Links presents an interesting area for further research and can have varied applications in diverse fields. The most challenging part of the project was synchronization and the feedback mechanism and we have been able to tackle that successfully. However, there is ample scope for development and the throughput can be improved by using a more complicated feedback mechanism such as computing MD-5 implemented along with Go-Back-N(the hash is displayed as a QR code after every 5 or so messages). A warning mechanism can also be put in place to warn the user if there is poor orientation or bad light conditions.

REFERENCES

[1] QR Code Essentials by Denso
[2] COBRA: Color Barcode Streaming for Smartphone Systems, Tian Hao, Ruogu Zhou, Guoliang Xing

[3] QR-Code Generator by Phaisarn Sutheebanjard and Wichian Premchaiswadi

[4] Recognition of QR Code with Mobile Phones by Yue Liu, Ju Yang and Mingjun LIu

[5] https://zxingnet.codeplex.com/

[6] https://github.com/phishman3579/android-quick-response-code

www.stackoverflow.com for all the invaluable help