Fenot Hailemichael

Fast Wi-Fi Setup Using QR Code & NFC

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Electronics

Thesis

01.10.2014

Helsinki Metropolia University of Applied Sciences

Author	Fenot Hailemichael
Title	Fast Wi-Fi Setup Using QR Code & NFC
Number of Pages	29 pages + 2 appendices
Date	01 October 2014
Degree	Bachelor of Engineering
Degree Programme	Electronics
Instructor	Janne Mäntykoski, Senior Lecturer

The purpose of this project was to configure a smart phone to a given Wi-Fi network using NFC tags and QR Code. In this thesis, the methodology of both technologies, NFC and QR Codes, is discussed.

Near Field Communication (NFC) is a recent technology with many applications. Most of recently released smart phones have a built in NFC reader/writer chip, allowing them to write and read an NFC tag. In this project Adafruit's PN532 NFC/RFID controller breakout board and Mbed LPC1768 microcontroller were used to communicate with an NFC-capable smart phone in Peer-to-Peer mode.

QR Codes are 2-dimentional barcodes developed for holding more information than basic barcodes. In this thesis Wi-Fi network information is encoded to QR Code and displayed on an E-paper display. A library developed by Fukuchi Kentaro was used for encoding the Wi-Fi data. Later the displayed QR Code was scanned with a Samsung Galaxy s4 phone to establish a Wi-Fi connection.

For testing both the NFC PN532 board and QR Code a specific application was needed to be installed on the phone. Barcode Scanner from ZXing Team was used for the QR Code and AppNearMe application was used for the PN532 Breakout Board. Easy and fast Wi-Fi network configuration was achieved through QR Codes.

Keywords	NFC, QRcode, E-paper Display, Mbed Microcontroller, PN532, Automatic Identification System

Contents

Appendices

Appendix 1. Source code, NFCtest main.cpp

Appendix 2. Source code, QREncode main.cpp

1 Introduction

Wi-Fi network is wireless technology which is widely used as a means of connecting to the internet. But it is also a technology which allows fast data exchange between electronic devices. The purpose of this project is finding a fast and easy way of connecting an electronic device to a Wi-Fi network, which will allow the device to control a robot car. Even though the outcome of this project will be implemented on the robot car, this project was carried out independently.

One option for satisfying our need for fast and easy Wi-Fi configuration is using QR Codes. QR Codes are two-dimensional optical barcodes capable of storing various types of data. This paper is going to discuss how QR Codes can be generated and used to establish Wi-Fi network communication in a simple way. C++ programming language and an mbed LPC1768 microcontroller is used for generating the QR Code. Then the QR Code is displayed on an electronic paper for scanning. By using a barcode reader application installed on smart phones, a very fast and easy setup of Wi-Fi network is possible with QR Codes.

Another technology which can be used for establishing a simple connection handover to a Wi-Fi network is near field communication (NFC). It is a new contactless technology for two-way interaction between electronic devices. NFC tags can be written for carrying out different tasks like save a contact, start a specific application on the phone, or connect to a Wi-Fi network etc… Most of recently released smart phones come with a built in NFC reader/writer chip, allowing them to use NFC technology as desired. This paper is going to discuss about the technology in detail and also the possibility of using it to connect a smart phone to a specific wireless network.

2 Near Field Communication (NFC)

Near field communication (NFC) is a standard short range means of communication between two electronic devices. It operates on the frequency range of 13,56MHz and allows data transmission between electronic devices in close proximity up to 10cm. Unlike other wireless communication technologies, NFC connects two electronic devices in close proximity automatically without any setup by users. Its short distance

operation makes it more secure than other wireless communications, thus it is currently incorporated in many applications.

Most of its applications are on NFC enabled mobile phones, as many people are always with their mobile phone it is a best method for providing everyday services. For example an NFC enabled smart phone can be used as a ticket, a contactless credit/debit card or an access key. It can also be used to initiate other kinds of wireless communications, such as Bluetooth or Wi-Fi, between two electronic devises [1].

2.1 History

The history of NFC dates back to the beginning of radio frequency identification (RFID) technology in 1983. In 2002, Sony and NXP Semiconductors (Philips by then) developed a new RFID technology which makes it easier to add an RFID interface to electronic devices. For security purposes this new technology had a shorter communication range, and,thus the name Near Field Communication (NFC) was given.

Later in 2004, three companies Nokia, NXP Semiconductors and Sony jointly established the NFC Forum to add touch-based interactions in consumer electronics, mobile devices, smart objects, and for payment purposes. The Forum's goals include; developing standards-based Near Field Communication specifications that define a modular architecture and interoperability parameters for NFC devices and protocols, encouraging the development of products using NFC Forum specifications, and working to ensure that products claiming NFC capabilities comply with NFC Forum specifications [1].

NFC is specified according to standards ISO/IEC 18092 NFCIP-1 (ECMA 340) and ISO/IEC 21481 NFCIP-2 (ECMA 352). These standards specify the modulation schemes, coding, transfer speeds, and frame format of the RF interface of NFC devices. NFC is compatible with the NXP's MIFARE® and Sony's FeliCa™ contactless smart card technologies [2, 375-379].

The first NFC enabled mobile phone, Nokia 6131, was produced and sold in 2006 the same year when NFC Forum formally outlined the architecture of NFC technology. At this stage the idea of smart posters, NFC tags or smart tags were circulating in many companies for incorporation. These technologies would allow the user to get informa-

tion, download a video, music, picture etc just by passing their phone close to the writing/reading device. As the years passed, more specifications emerged and the technology grows to peer to peer (P2P) applications, which allowed sharing data between smart phones and other NFC devices. In 2010 Samsung released the first android NFC enabled phone. Today NFC technology is used most dominantly in Europe and Asia, and is growing fast in the US [4; 3].

2.2 Technical Principle and Operating Modes

NFC is a wireless data interface between devices using high-frequency alternating magnetic fields. Similar to proximity RFID and contactless smart cards, NFC uses the license-free 13.56 MHz frequency radio spectrum and it offers data rates up to 424Kb/s. In this technology NFC devices can act both like a reader and a transponder, as they integrate both the transmitter and receiver which are alternately connected to the antenna.

Data is transmitted from one NFC device to the other by the principle of mutual induction; an alternating current passes through the transmitter antenna to generate an electromagnetic field which, if in proximity, will induce a current in the receiver antenna as it is shown in Figure 1 below. The RF sine wave, at 13.56 MHz frequency, generated by the antenna is called the carrier signal. For data transmission the carrier signal's frequency, amplitude or phase is altered depending on the type of modulation used. Modulation is the method of adding extra information to the carrier signal. The two frequently used modulation techniques in NFC are Amplitude Shift Keying (ASK) and Phase Shift Keying (PSK).

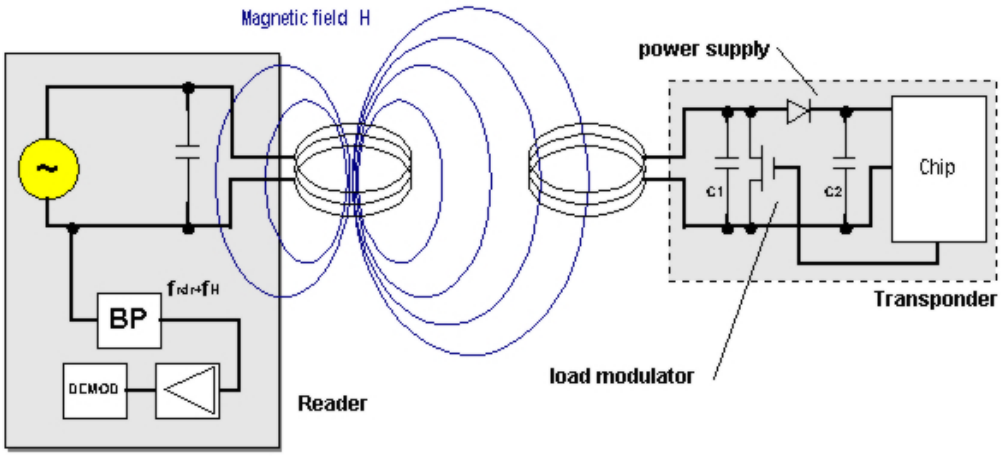

Figure 1. Principle of mutual induction [2, 35]

Depending on how the NFC target device is powered, NFC distinguishes between two operational modes; passive mode and active mode. In passive mode the NFC target device is not powered internally, it will be powered when a NFC initiator device in the proximity starts communication. Here the NFC transmits blocks of data by amplitude modulation to the NFC target device, and continues emitting the unmodulated carrier signal after data transfer is done. The NFC target device then sends back data to the NFC initiator by load modulation. In active mode both NFC devices, which want to communicate, are self-powered and both can act as a reader or a transponder. One of the devices initiates communication by sending an NFC initiator commands and the other NFC device adopts the role of NFC target device. After this data transmission takes place both ways by amplitude modulation, both NFC devices alternately induce magnetic fields when data is sent from transmitter to receiver.

NFC supports three modes of communication depending on how NFC devices interact with each other. Namely: Card Emulation mode, Peer to Peer mode, and Reader/Writer mode.

- Card Emulation mode: In this mode an active NFC device acts as a contact-less smart card allowing other NFC device to read data from it. Here the mode of operation is passive and the NFC device acting as a contactless smart card does not generate its own RF field. This mode enables NFC devices, like NFC

enabled smart phones, to be used for ticketing, payment, and access control applications.

- Peer to Peer mode: In this mode easy data exchange is possible by just holding two NFC devices close together. During communication both NFC devices are in active mode and data is sent over a bi-directional half duplex channel. This means data transfer is not done simultaneously, i.e. one device listens while the other is transmitting, and starts to transmit once the previous data transmission is done. This mode enables two NFC enabled mobile phone to exchange data such as business card, text message, music playlist etc...
- Reader/Writer mode: The NFC device acts as a reader or writer as it gets close enough to a passive tag. A passive mode of operation is in use, thus the tag is powered by the NFC device and it transmits data to the NFC device by load modulation. The NFC device can read the data stored in the tag's memory or write to it depending on the application. In NFC enabled smart phones this mode is used to read smart posters, read electronic product code, start a dedicated application, Bluetooth or Wi-Fi setup, and much more.

In NFC, whether the communication is between active NFC devices or an active NFC device and a tag, an NFC Data Exchange Format (NDEF) message is used for data transfer. As defined by the NFC Forum, NDEF is a light weight binary message format that can be used to encapsulate one or more application-defined payloads of arbitrary type and size in to a single message. An NDEF message is composed of one or more NDEF records, each consisting a header and payload of arbitrary type. The header includes indicators for the payload length, payload type, and if needed a payload identifier. The payload can be of different types: MIME media type or one of the predefined NFC record type definitions (RTD) [1; 3; 4; 5].

2.3 NFC Tags

A tag is a simple device containing an antenna and a memory chip. Most NFC tags are passive and acquire energy from the RF field generated by the NFC device. A simple NFC type 1 tag is shown in Figure 2. NFC tags can be used in applications like smart posters, and other areas where a small amount of data can be stored and transferred to active NFC devices.

Figure 2. NFC tag type 1 and NFC Forum Logo [1]

The NFC Forum has defined four types of tags, so that interoperability between different NFC tag providers and NFC device manufacturers can be achieved. These NFC tag type formats are based on ISO 14443 Types A and B (international standards for contactless smart cards) and ISO 18092 (the passive communication mode) standards.

- Tag 1 Type: These tags are based on ISO 1443A standard. They have 97 bytes of memory and data speed of 106Kbit/s. If needed the memory can be expanded to 2 Kbyte. They are read and re-write capable, in addition users can configure them to be read-only.
- Tag 2 Type: Type 2 tags are similar to type 1 tags and derived from NXP MIFARE® Ultralight tag. They have anti-collision support.
- Tag 3 Type: These tags are derived from the non-secure parts of Sony FeliCa™ tags and are more expensive than type 1 and 2 tags. These tags are pre-configured during manufacturing and, thus, can be either read and re-writable or read-only. They have 2 Kbytes memory and data rate of 212 Kbits/s or 424 Kbits/s. Though they cost more than type 1 and 2 tags, they are more applicable for more complex applications.
- Tag 4 Type: The NFC Tag 4 Type is derived from NXP DesFire tag, and is based on ISO 14443A standard. They support communication speed up to 424 Kbits/s and have a maximum memory capacity of 32KB per service [2; 4].

3 QR Code

QR Code, also called two-dimensional barcode, is a type of two dimensional barcode which can store different types of data, like numeric, alphanumeric, Kanji, Kana, Hiragana, symbols, binary and control codes. Unlike conventional barcodes which are capable of storing 20 digits maximum, QR Codes can store up to 7,089 characters i.e. hundreds times more information. Other features of QR Codes include small print out size, dirt and damage resistance, readability from any direction and structured appending feature. Structured append is used to divide big data blocks in to small and give multiple small QR Codes i.e. instead of one very big QR Code. Figure 3 illustrates a basic QR Code.

Figure 3. QR Code

Applications of QR Codes, though initially were on vehicle part tracking, now target mobile phone users. Users can receive contact information (phone number, e-mail address), calendar event, a URL, an SMS, a geo location, Wi-Fi network setup etc... They are mostly used in the shopping industry, commercial tracking, entertainment and transport ticketing including usage in the airport. The fact that QR Code is license free, the availability of free mobile apps and online QR Code generating software makes its use globally popular [6].

3.1 History

QR codes have been created in 1994 by Denso Wave, a subsidiary of Toyota Car Company. They were used by the company to track car parts in Japan. The main rea-

son for developing QR Codes was the need of more storage capacity and capability of coding Kanji characters.

Mr Mashahiro Hara, who was in charge of the development of QR Codes, came up with idea of using 2D codes to hold more data and at the same time to be read easily. Reading the code as fast as possible was the biggest challenge for the team, so they thought of adding positional information indicating the existence of a code to be read.

The positional information was made up of square marks, and by combining these marks to the code a fast readability feature was achieved. They chose the marks to be square because this pattern was less likely to appear on many business forums and other places. The position detection patterns have to be exclusively unique. If there would be similar looking mark around the area, the reader would be confused and give an erroneous result. After doing an exhausting survey about the ratio of white to black areas in printed matters they came up with the least used ratio, 1:1:3:1:1. Based on this ratio they decided the width of black and white areas in position detection patterns, allowing the code to be read from any angle.

Finally the team was able to develop a QR Code capable of holding 7,000 numerals with additional capability of coding Kanji character. This code could be read 10 times faster than any other codes, thus the name Quick Response was given. After its release in 1994, QR Code was adopted by many companies to trace goods and control their merchandise.

In 2002 following the release of mobile phones capable of reading QR Codes, it became widely used in Japan. Since it is an open code for everyone to use, nowadays it is used all over the world in commercial tracking application and convenience-oriented applications aimed at mobile phone users. Some of these applications are: displaying text to user, add a contact, opening a URL, compose an e-mail or text message, electronic tickets etc...

Currently, there are different types of QR Codes developed to meet more sophisticated user needs. For need of smaller codes to be printed on very small area, a micro QR Code was developed and made a JIS standard in 2004. In 2008, the iQR Code, having a small footprint regardless of its large coding capacity and allowing the use of rectan-

gular code modules, was released. In LogoQ it is possible to use any colour instead of the basic black & white pattern and to embed a picture in it [2; 3].

3.2 Symbol Structure

The QR Code structure mainly consists of an encoding region, function patterns, and a quiet zone. The encoding region is part of the QR Code where information concerning the encoded data is placed. This region contains the Data codewords, error correction codewords, format information, and version information. Function patterns are shapes placed inside the QR Code to make sure data will be decoded correctly by the QR Code reader or scanner. They are namely finder patterns, separator, timing patterns, and alignment patterns. The quiet zone is the area surrounding all four sides of the QR Code separating it from any other markings. The QR Code symbol structure is shown in figure 4 below. The finder patterns at the three corners of the QR Code allow readability from any direction, both rotation and reflection.

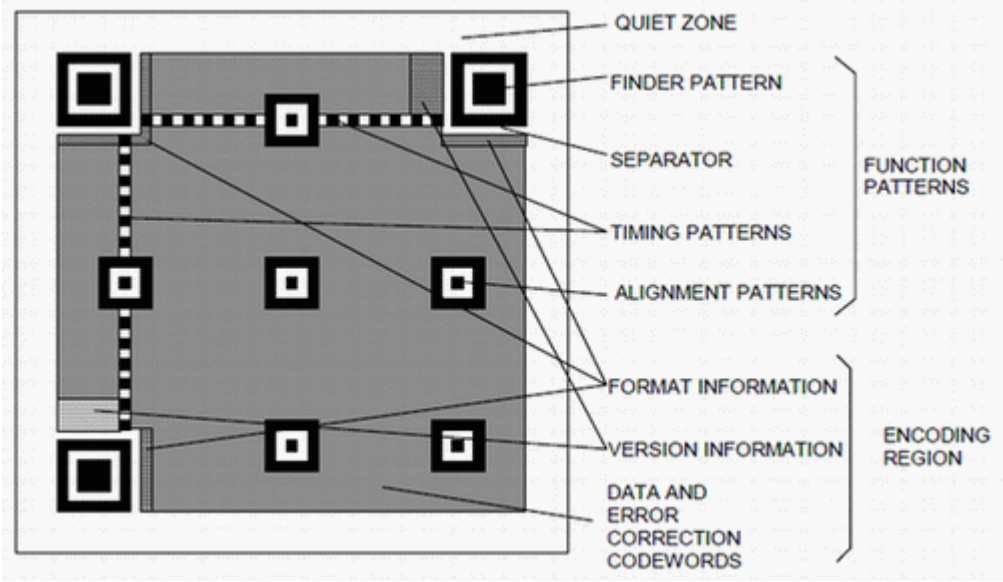

Figure 4. Symbol Structure [9]

The key feature of QR Code, readability from any direction both rotation and reflection, is because of the finder patterns placed at the three corners, as shown in figure 4. There are 40 sizes of QR Code symbol which ranges from version 1 to 40. The smallest version is 21 X 21 modules and the maximum is 177 X 177 modules. The separators are lines of white modules placed besides the finder pattern to separate them from

the rest of the QR Code. Alignment patterns are placed in the QR Code for correcting distortion when the code is bent or curved. All QR Codes that are version 2 and larger are required to have alignment patterns. They help to detect the position of each cell in the QR Code [9].

3.3 Types of QR codes

Today there are many types of QR Codes which are modified versions of the basic QR Code to meet different application needs. In this section detailed information of different QR Code types from the earliest to the most recent developments is given.

3.3.1 QR Code model 1 and 2

Model 1 QR Code is the original QR Code which was capable of encoding 1,167 numerals. Later it was improved to a capacity of encoding 7,089 numerals and called model 2 QR Code. It has 40 versions; the biggest one has capacity of 177 x 177 modules. It also has an error correction level ranging from low to high (L, M, Q, H), thus, it can be read even if some part of it is damaged. Figure 5 shows QR Code model 1 and 2. Here you can see that in model 1 there is no alignment pattern and data modules are more compact in model 2.

Figure 5. QR Code Model 1 and 2 [6]

3.3.2 Micro QR Code

As the name indicates, Micro QR Code is a very small QR Code which is limited to encoding 35 numerals maximum. It has only one position detection pattern and needs only 2-module wide margin around it. It has 4 different versions, M1 (11 X 11 modules) to M4 (17 X17 modules). Since data is encoded more efficiently in Micro QR Code, the

size of M4 is smaller than version 1 of basic QR Code. Figure 6 below compares Micro QR Code with basic QR Code.

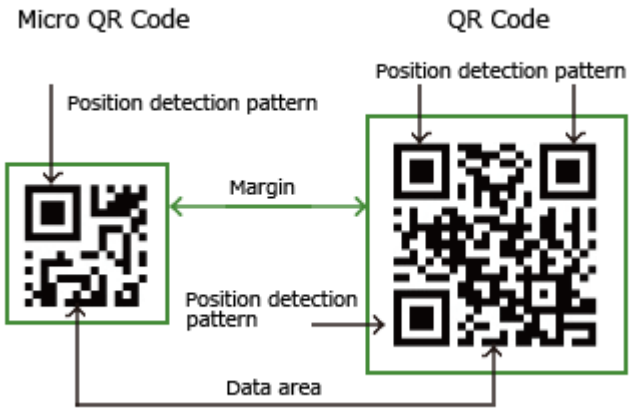

Figure 6. Micro QR Code [6]

3.3.3 iQR Code

iQR Code is a modified QR Code in every feature. It is possible to retrieve data even if 50% of the entire data is damaged or smeared. Printing on cylindrical products and maintaining their readability is made possible by the rectangular modules offered. Three iQR Code versions are shown in figure 7.

Figure 7. iQR Codes [6]

Table 1 below shows the modified features of iQR Code as compared with a basic QR Code.

Table1. iQR Code Vs QR Code [6].

	iQR Code	QR Code
Version	1(9 x 9 modules) to 61(422 x 422 modules) R1(5 x 19 modules) to R15(43 x 131 modules)	1(21 x 21 modules) to 40 (177 x 177 modules)
Error correction	L (7%), M (15%), Q (25%), H (30%) & S (50%)	L (7%), M (15%), Q (25%), H (30%)
Margin	1 module(2 modules for 2-byte character)	4 modules

The maximum version in iQR code is 61 for square QR Code and R15 for rectangular QR Codes. Unlike basic QR Codes, it needs only a maximum of 2 modules wide quiet zone. It also has an additional error correction level S of 50% to the four basic QR Code error correction levels (L, M, Q, &H)

3.3.4 SQRC

SQRC is a type of QR Code equipped with reading restriction function. Data for SQRC has two parts: public and private, meaning it is possible to store 2 control levels of information in one code. An SQRC compatible scanner has to have the same cryptography key as in the SQRC, in order to read the private data. Though SQRC can be read only by specific types of scanners, it does not mean guaranteed securing of data. In figure 8, it is shown how the reading result differs depending on the type of reader/scanner used. So unauthorized reader/scanner can only get the public part of the data [6].

Figure 8. SQRC [6]

3.3.5 LogoQ

In LogoQ design is integrated in QR Code, where a QR Code can be customized by adding a company Logo. Also it is not limited to using only black and white colours; other printing colours can be chosen. With LogoQ it is possible to combine design ability and readability. Figure 9 below shows a simple LogoQ construction example.

Figure 9. LogoQ [6]

4 QR Encoding

In order to generate a QR Code for some data, one must follow a series of steps. An overview of these steps is given in this section.

4.1 Data Analysis

Data analysis is the first step in generating a QR code. There are four modes of encoding the text as string of bits: numeric, alphanumeric, byte, and Kanji. In this step we

choose which one of these modes can give the shortest possible string of bits for our data to be encoded.

As the name already implies, Numeric mode is for encoding decimal digits (0 - 9). Alphanumeric mode is for encoding decimal digits (0 - 9), 26 alphabetic characters (A - Z), and 9 symbols (SP, $, %, *, +, -, ., /, :). The Byte mode is for encoding characters from the ISO-8859-1 character set. Kanji mode is for encoding Kanji characters in accordance with the Shift JIS system based on JIS X 0208.

4.2 Data Encoding

As mentioned above there are four different methods of encoding depending on the content of our data. For each mode there is a specific binary mode indicator and they are shown on Table 2.

Table 2. Mode indicators

Mode	Indicator
Numeric	0001
Alphanumeric	0010
8-bit Byte	0100
Kanji	1000

QR Code symbol has 40 different sizes, called versions, ranging from Version 1 to Version 40. The capacity of each version depends on the mode and error correction level in use. Refer to the Character Capacity table [7, 28-32] for checking the capacity of version for a specific encoding mode and error correction level.

In the simplest numeric mode encoding the string of digits are divided into groups of three digits, and each group is converted to its 10 bit binary equivalent. If the string is not a multiple of 3, the last group will have 2 or only 1 digit which will be converted to 7 or 4 bits respectively.

4.3 Error Correction

A QR Code reader or scanner needs some kind of extra information to recover data in case of a physical damage to the code or misdecoded symbol character. For this reason an error correction codeword is created and added to the data codeword sequence during the encoding process. The error correction code words are generated from the

data bits by using Reed-Solomon error correction. There are four error correction levels of which the user can freely choose, considering how much data recovery is needed. Below Table 3 describes the error correction levels.

Table 3 – Error Correction Level

Error Correction Level	Data Recovery Capacity
L	7%
M	15%
Q	25%
H	30%

Note that the error correction level has a direct relation to the size of the generated QR Code, choosing higher error correction level results in a larger QR Code than in using low error correction level.

4.4 Structure Final Message

After generating the data codeword and error correction codeword they should now be arranged in a proper order. This is fairly easy in case of small QR Codes as the error correction codeword can just be placed after the data codeword. In large QR Codes data and error correction code words are generated in blocks. Before module placement, these blocks should be interleaved according to QR Code specification.

4.5 Module Placement in Matrix

Once the code words are arranged in order, they are ready for placement in the QR Code matrix. In this step function patterns and code words are placed in a specific manner inside the matrix. First a blank matrix is constructed as the size of the version in use, after this positions for the function patters are filled with dark or light modules as appropriate. The encoding region of the QR Code is then filled with the data bits.

4.6 Masking

Dark and light modules inside QR Code are expected to be balanced for getting a reliable reading output; especially the position detection pattern should be avoided in other areas of the QR Code, as possible. So a mask pattern, which changes light and dark modules according to a particular rule, is used to meet the above condition.

4.7 Format & Version Information

A format information string encodes the error correction level and mask pattern used in the current QR Code. This string is 15 bit long and added to the QR Code matrix, below the topmost and next to the leftmost finder patterns. For QR Code version larger than 7, an 18-bit version information string must be placed in the bottom left and top right corner of the QR Code. After adding this last information a finalizing quiet zone, a 4-module wide area of light modules, must surround the QR Code for reliable reading [7; 8].

5 Implementation and Analysis

The goal of this project was to setup a Wi-Fi wireless connection using the two most widely used automatic identification systems, QR Codes and NFC. This section explains the method and materials used for encoding and retrieving a data in QR Code and NFC systems, by going through the hardware and software requirements.

5.1 Hardware

In this project mainly 3 different electronic devices were used.This part gives detailed information on the hardware components used and their respective electronic connection. Adafruit's PN532 Breakout board, NFC reader/writer device, was used with a host microcontroller Mbed LPC1768 for getting started with NFC. For the QR Code part, again the Mbed LPC 1768 microcontroller was used for generating and displaying the QR Code on a 2.7 inch electronic paper display.

5.1.1 PN532 Breakout Board

The PN532 Breakout Board is a product of Adafruit Industries and it mainly uses a PN532 highly integrated transceiver module by NXP Semiconductors for contactless communication at 13.56 MHz frequency band. It has an antenna, antenna matching circuit, PN532 IC, external connection pins, and SEL0 & SEL1 options for choosing interface with the host microcontroller. The board will be powered from the host microcontroller and thus should always be connected to it for performing a task. Figure 10 shows PN532 Breakout board, note that header pins are not soldered during purchase.

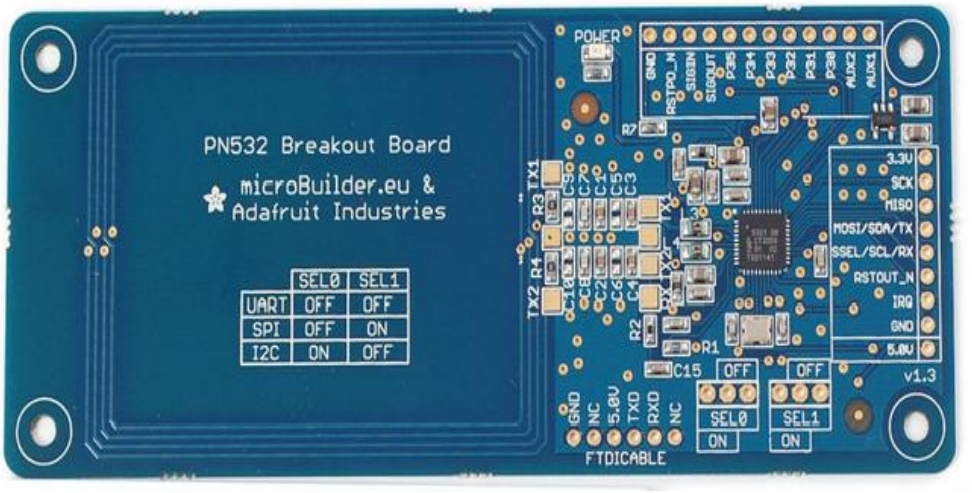

Figure 10. PN532 Breakout Board [10]

PN532 NFC controller supports all three NFC communication modes discussed in section 2.2 and, thus, can be used as a FeliCa & MIFARE Reader/Writer, FeliCa & MIFARE card emulation, or ISO/IEC 18092, ECMA 340 Peer-to-Peer. The breakout board supports SPI, I^2C, and TTL serial interface [10].

5.1.2 Mbed LPC1768 Microcontroller

This is a series of ARM microcontroller development boards designed for prototyping many devices. For simple programming, it has a built-in USB drag 'n' drop FLASH programmer. It includes 512KB FLASH, 32KB RAM, and interfaces like built-in Ethernet, USB Host and Device, CAN, SPI, I^2C, ADC, DAC, PWM, and other I/O interfaces. The most commonly used interfaces and their locations are shown in figure 11. It has a USB port for PC interface.

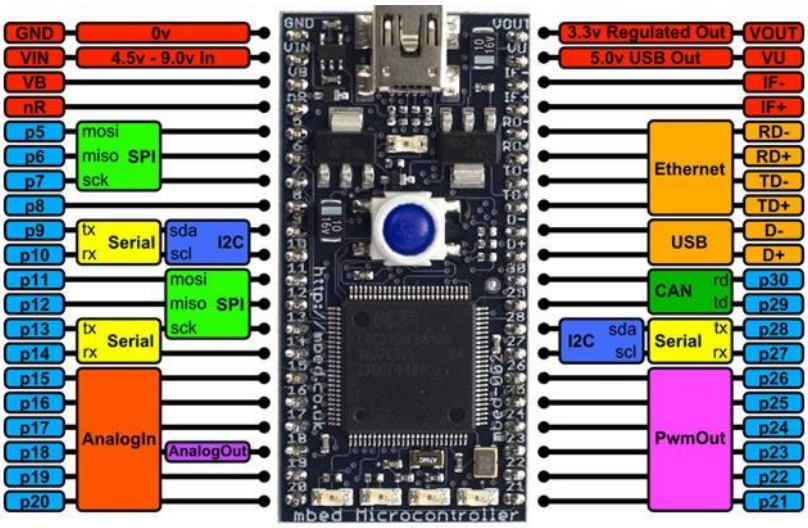

Figure 11. Mbed LPC1768 [10]

5.1.3 E-paper Display

For display purpose a 2.7 inch E-paper Display by Embedded Artist was chosen. It is an electronic paper display which holds the displayed data on screen after power has been removed. It is a black and white display and supply power of 3.3V is required for updating the data. It has an SPI interface and control signals including PWM [11; 12]. The pin numbering on the display is shown in figure 12.

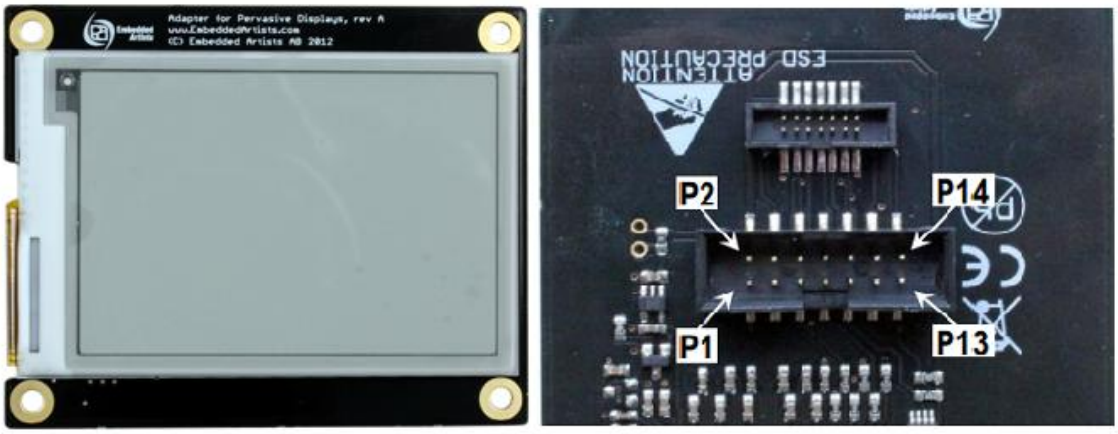

Figure 12. Front and back view of a 2.7' E-paper Display[11].

Hardware Wiring

Hardware components listed below were used for working on the NFC part of the project.

- Mbed LPC1768 microcontroller

- PN532 Breakout Board
- 7 pieces female to male prototyping wires and 2 jumpers and
- A USB connector cable

Using the prototyping wires the Mbed microcontroller was connected to the PN532 Breakout Board according to the pin functions as shown in table 4. Since it was decided to use SPI interface, jumpers were placed on SEL0 OFF and SEL1 ON. A USB connector cable was used to connect the Mbed microcontroller to host PC.

Table 4. Mbed LPC1768 to PN532 board pin connetion [10].

Mbed LPC1768 microcontroller	PN532 Breakout Board
Vout	3.3V
p13	SCK
p12	MISO
p11	MOSI/SDA/TX
p19	SSEL/SCL/RX
p18	IRQ
GND	GND

Table 5. Mbed LPC1768 to E-paper Display Pin connection [11].

Mbed LPC1768		E-paper Display	
Pin	function	Pin	function
GND	GND	1	GND
p5	MOSI	4	MOSI
p6	MISO	5	MISO
p7	SCK	3	SCK
p8	GPIO	6	SSEL
p9	GPIO	13	Power control
p10	GPIO	8	Border
p11	GPIO	14	Discharge
p12	GPIO	12	Reset
p13	GPIO	7	Busy
p26	PWM	11	PWM
p27	SCL	9	SCL
p28	SDA	10	SDA
VOUT	3V3	2	3V3

For the second part of this project, QR Code, an E-paper display, Mbed microcontroller, 14 prototyping wires and a USB cable for connection were used.
The first step was to connect the display with mbed LPC1768 microcontroller and the microcontroller to the PC. The necessary pin connections are shown in the above Table 5.

5.2 Software

Getting started with Mbed is quite easy with mbed.org Developer Website, which offers lightweight Online Compiler and cookbook of published libraries and projects. For using this online compiler a user registration is required. After this a user can start working on new or imported projects here. Figure13 below shows the online compiler.

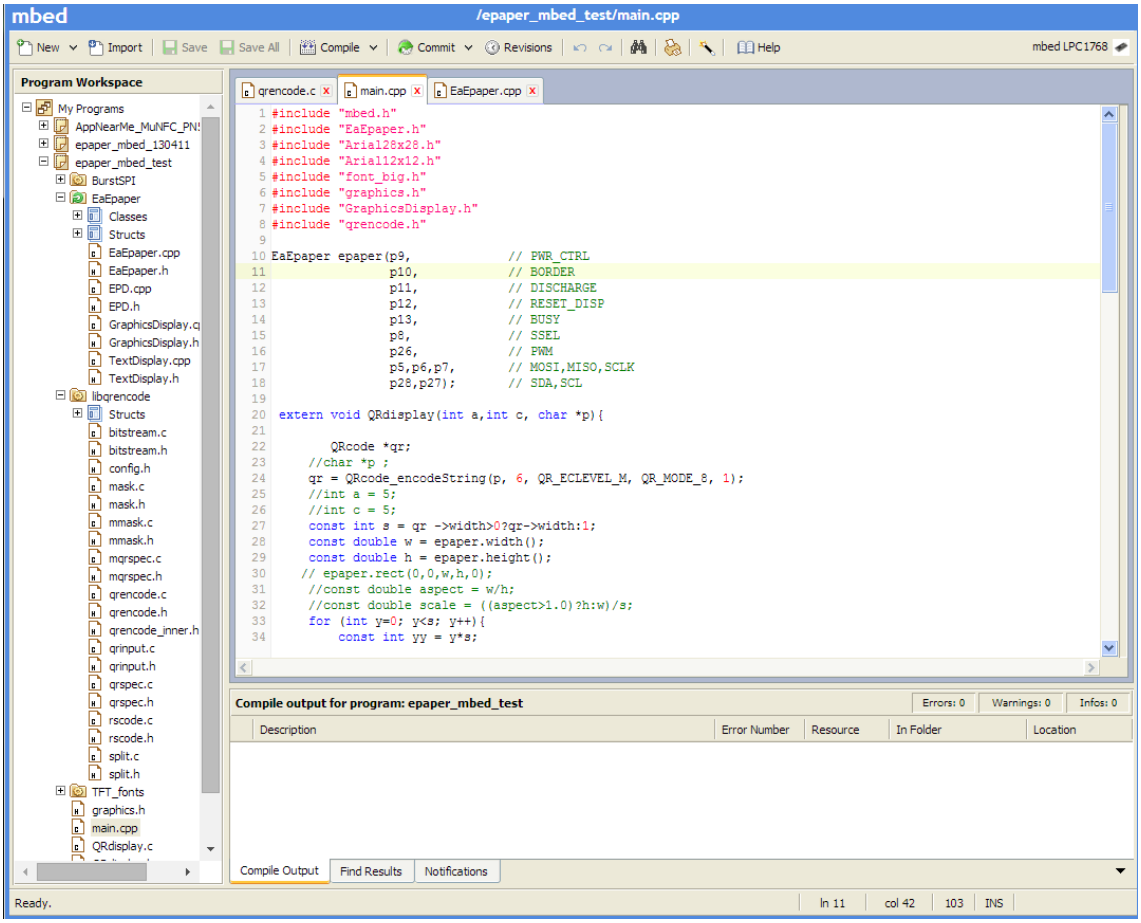

Figure 13. Mbed Online Compiler

NFC Project

At first a simple program for sending a string of characters to another NFC device was run on the PN352 Breakout Board. Here the Mbed microcontroller with the online compiler was used for programming. A Samsung Galaxy S4, NFC-capable, phone will be used to communicate with the PC using PN532 Breakout Board. On the mobile phone there should be a dedicated application, in this case AppNearMe, for communication.

A new program called 'NFCtest' was created on the Mbed Online compiler. Then the necessary libraries for the NFC project were imported. These libraries were: the official mbed C/C++ SDK, the official mbed Real Time Operating System 'mbed-rtos', and AppNearMe_MuNFC_PN532. The AppNearMe_MuNFC_PN532 library [13] is published for public use by AppNearMe Official. After all the necessary libraries were imported, a main.cpp file was created for the program. Here the SPI interface pins are specified, as shown on table 4, and the NFC transaction is started. The main program source code can be found in the appendix 1 for reference. Next, the main program was compiled and the compiled output was downloaded to the Mbed LPC1768 microcontroller.

QR Code Project

For encoding the Wi-Fi data to QR Code in C++, a QR-Encoding C-library 'libqrencode' published to the public use by Fukuchi Kentaro was used [14]. The library accepts a string or list of data and then converts it to a raw bitmap data. Similar to what was done in the NFC project, first a new program named 'QREncode' was created on the online compiler. The port of libqrencode for embed, the mbed-rtos, BurstSPI, TFT_fonts, and mbed C/C++ SDK were then imported to the QREncode project. On the other hand, the E-paper display getting started published project called 'epaper_mbed_test' [15] has been imported and used for testing the E-paper display. This was done to get familiar with programming the E-paper display. The next task was to import the EaEpaper library to our QREncode project for displaying the final QR Code. Then the next task was to write the main program for outputting the encoded Wi-Fi data on the E-paper display. Figure 14 below shows the QREncode program on mbed online compiler.

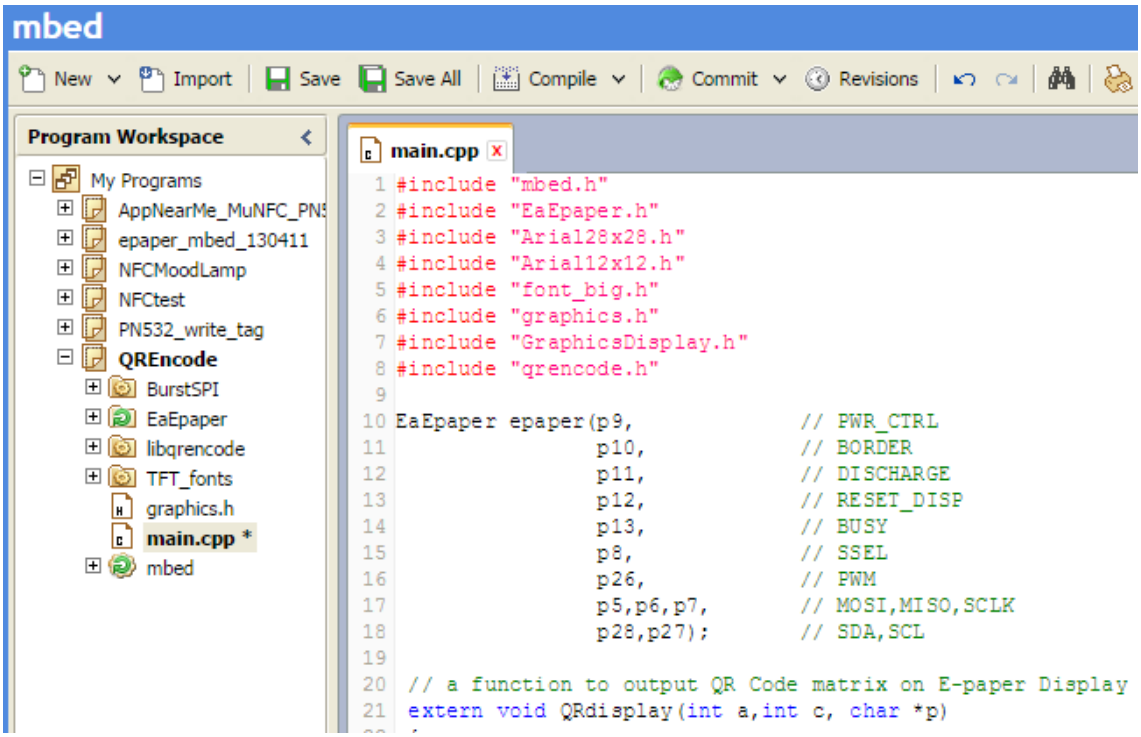

Figure 14. The QREncode program

On figure 14 from the left hand side program workspace list, we can see the QREncode project and the libraries and source code files included inside. The first library, BurstSPI, is used to send SPI data at very high speed. This library and the graphics.h source code were taken from the project 'epaper_mbed_test'. The TFT_fonts library was used for writing text on the E-paper display with different fonts. The main library for display is the EaEpaper library, which handles the interface between mbed microcontroller and the display and graphic display settings.

Before writing the main program a basic understanding of the two main libraries, libqrencode and EaEpaper is needed. As mentioned above, the E-paper was tested with the 'epaper_mbed_test' project and, thus, the EaEpaper library is familiar.
The 'libqrencode' library implements the QR encoding process which was discussed in section 4. It takes a string of data and converts it to a QR Code data matrix. It is possible to generate model2 QR Code or Micro QR Code with this library. For encoding a string, the function *QRcode_encodeString()* can be used.

```
QRcode *QRcode_encodeString(const char *string, int ver-
sion, QRecLevel level, QRencodeMode hint, int casesensi-
tive)
{
    return    QRcode_encodeStringReal(string,    version,
level, 0, hint, casesensitive);
}
```

Listing 1. plot table from qrencode.c

As listing 1 illustrates the function *QRcode_encodeString()* takes four parameters as input and outputs a *QRCode* object. The four input parameters are the data string to be encoded, the version number, the error correction level, the data type, and case sensitivity of data. As mentioned in the QR Encoding section, it is necessary to analyze the data and decide which version, error correction level, and encoding mode to use. The Wi-Fi setup data is composed of the SSID, Password, and type of the Wi-Fi network.

Data string = "`WIFI: T: WPA; S: bunnu; P: pass123; H: HIDDEN`"

In the data string above `WIFI` stands for Wi-Fi configuration, T- is for Wi-Fi type, `S-` is for SSID, P- is for Password, and H- is for keeping the password from being displayed on the reading device. A medium error correction level is chosen considering that the QR Code will be displayed on an E-paper, which reduces the physical damage of tearing or wearing out. Having the data to encode ready and the error correction level fixed, the required version number was looked up from the Character Capacity table [7, 28-32]. The data string has 38 alphanumeric characters and with an error correction level M, the minimum version will be version 2. In 'libqrencode' two data encoding modes are available, QR_MODE_8 and QR_MODE_KANJI, for encoding alphanumeric and Kanji characters respectively. The QR_MODE_8 is chosen here because there are no kanji characters in the string of data.

The *QRCode* object returned from the *QRcode_encodeString()* function has three data fields: the version number, the width, and the data matrix. In order to draw these data to the E-paper display, the *QRdisplay()* function illustrated in listing 2 was used.

```
extern void QRdisplay(int a,int c, char *p)
 {
```

```
        QRcode *qr;
        qr    =    QRcode_encodeString(p,    2,    QR_ECLEVEL_M,
QR_MODE_8, 1);
          const int s = qr ->width>0?qr->width:1;
          for (int y=0; y<s; y++){
         const int yy = y*s;
         for(int x=0; x<s; x++){
             const int xx = yy+x;
             const unsigned char b = qr->data[xx];
             if(b &0x01){
                epaper.fill(x+a,y+c,1,1,1);
                 }
             }
          }
         }
```

Listing 2. Plot table from main.cpp of QREncode project

QRdisplay is a function which takes the starting x,y coordinate for drawing the QR Code and the data string to be encoded. Here iteration is used for drawing the pixel from bitmap. The main function calls the *QRdisplay()* for data encoding and display. Finally, the main program is compiled and downloaded to the mbed LPC1768 micro-controller for outputting a QR Code on the E-paper display.

The size of the QR Code symbol is calculated with a formula

$$W = [8*2 + (4V + 1)]X + 2Q \quad (1)[9]$$

Where W: width of QR Code, X: width of a module, V: version, and Q: silent zone width
By substituting the values of the variables from listing 2 and from description in section 3.3, the approximate width of the QR Code is calculated as follows.

X = 1, V = 2, Q = 4X = 4

W = [8*2 + (4*2 +1)]*1 + 2*4

W = 32 pixel

6 Testing

In this section the testing devices and final results of the NFC and QR Code projects are explained. To test if the generated QR Code was correct, a Samsung Galaxy S4 phone with Zing barcode scanner application was used. The QR Code displayed on the E-paper display as shown in figure 15 was scanned with the Zing barcode scanner application.

Figure 15. QR Code on E-paper display

The free barcode scanner application from Zing can be downloaded and installed from the android app store. In Figure 16 the result of the scanned QR Code is shown, and with just touching the ‘connect to Network’ option at the bottom connects the phone to the Wi-Fi network automatically.

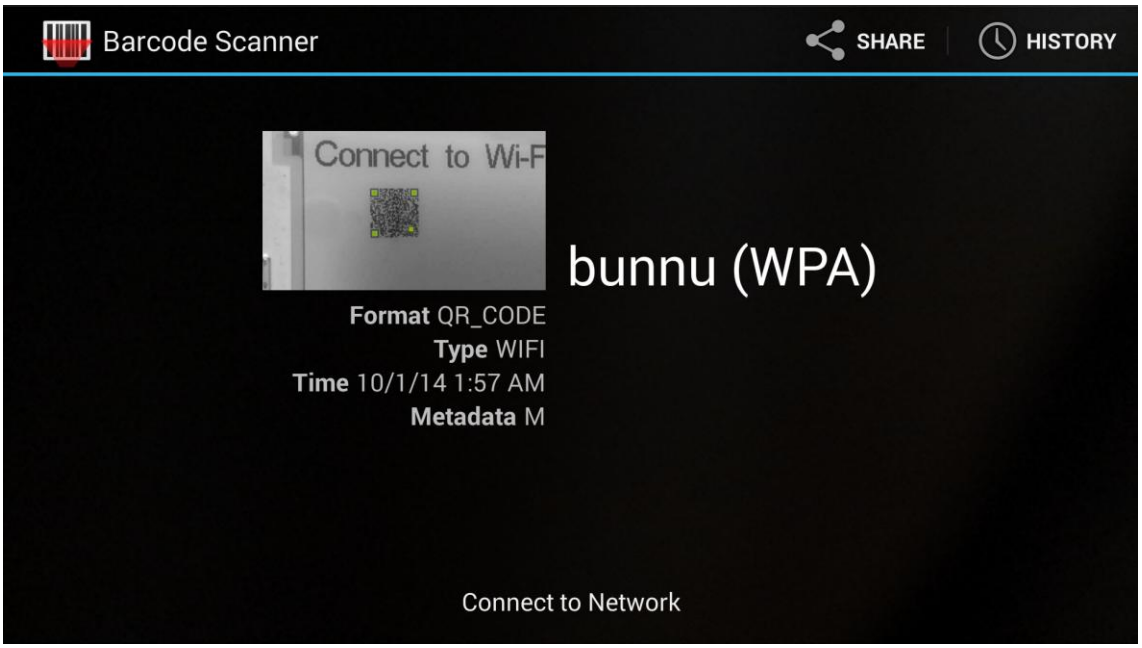

Figure 16. Reading the QR Code

Generating the QR Code for Wi-Fi setup was a success for a mobile phone application. The size of the QR Code is affected by the data size, but it can be bigger or smaller by changing the module width (X). Readability of the QR Code is better when the size is larger.

The NFC part of the project was carried out just to get started with the PN532 Breakout Board showing how it can be used in the Peer-to-Peer mode. After the program 'NFCtest' was compiled and downloaded to the mbed microcontroller, the PN532 board initiates communication with the other NFC device (Samsung Galaxy S4). As the phone is brought to proximity of the PN532 board, it is directed to an app store for installing AppNearMe, the dedicated application for the communication. After installation is finished a text from the host PC is shown on the screen. Now, a string of data can be sent back to the host PC by entering the text and taking the phone close to the PN532 board again. This way, a two way communication between the smart phone and host PC through the PN532 Breakout Board was achieved.

The PN532 Board was not tested for writing/reading a tag, or card emulation mode because of resource shortage. Thus, the Wi-Fi setup of an NFC device using the PN532 board was not a success.

7 Discussion and Conclusion

NFC tags and QR Codes provide similar capabilities and are used in many physical token applications ranging from a simple parking ticket to transportation ticket. In this paper both the NFC and QR Code technologies were discussed and they were tested to configure a smart phone's Wi-Fi setup. By using the licence-free QR encoding library by Fucuchi Kentaro, it was possible to generate a QR Code for Wi-Fi setup. This method can be used for easy and fast Wi-Fi network communication setup eliminating fiddling with settings. The mobile application for scanning barcodes works for all types of barcodes and QR Codes and thus, it is most likely that people have it installed on their phone. Since all smart phones have a camera, there is no physical limitation in using QR Codes for setting up a Wi-Fi network. As compared to the NFC communication which is accomplished with just a touch, reading a QR Code is a little bit slower since a user is expected to open the scanning application and read the code. On the other hand QR Codes are much cheaper to produce than NFC tags.

NFC technology is used for various and comprehensive applications in the everyday activities of people. NFC Forum is working on making the standards and specifications for NFC devices. However, for the moment there are not many mobile applications for the public use, which enable interoperability with tag types or type of application used. For example, if someone programs an NFC tag for Wi-Fi setup and lets many people with an NFC-capable mobile phone tap it for connection handover, this may even take longer time than entering an SSID and password, as people will be directed to install the dedicated application first. In the near future, when NFC is widely implemented and more generalized software applications are built, this technology will be ideal for many applications.

This project can be further continued by finding more resources on how to use Mbed microcontroller with PN532 board or maybe using Arduino instead. There are more simplified examples and tutorials provided with Arduino.

References

1 NFC Forum [Online].
URL: http://nfc-forum.org/.
Last accessed 29 September 2014

2 Klaus Finkenzeller. RFID Handbook. Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication. 3rd edition.
UK:John Wiley & Sons, Ltd. 2010

3 Resources and Analysis for Electronics Engineers. Near Field Communication Tutorial [Online].
URL:http://www.radio-electronics.com/info/wireless/nfc/near-fieldcommunications-tutorial.php.
Last accessed 29 September 2014

4 Forum.Nokia. Introduction to NFC. [Online, PDF document]
URL: http://www.adafruit.com/datasheets/Introduction_to_NFC_v1_0_en.pdf
Last accessed 17 September 2014

5 Simon Burkard. Near Field Communication in Smartphones [Online, PDF document]
URL:https://www.snet.tu-berlin.de/fileadmin/fg220/courses/WS1112/snet-project/nfc-in-smartphones_burkard.pdf.
Last accessed 01 September 2014

6 DENSO WAVE. QR code.com [Online]
URL: http://www.qrcode.com/en/index.html.
Last accessed 24 September 2014

7 ISO. International Standard. Information Technology-Automatic Identification and Data Capture Techniques-Barcode Symbology-QR Code [Online, PDF document]. URL: http://raidenii.net/files/datasheets/misc/qr_code.pdf.
Last accessed 01 September 2014

8 Thonky.com. QR Code Tutorial. [Online]
http://www.thonky.com/qr-code-tutorial/. Page last updated 25 July 2014.
Last accessed 17 September 2014

9 OnBarcode.com. QR Code. [Online]
URL: http://www.onbarcode.com/qr_code/.
Last accessed 01 September 2014

10 Lady Ada. Adafruit PN532 RFID/NFC Breakout and Shield. [Online, PDF document]
URL:https://learn.adafruit.com/downloads/pdf/adafruit-pn532-rfid-nfc.pdf.
Page last updated 17 April 2014. Last accessed 27 September 2014

11 Embedded Artists AB. Embedded Artists 2.7 inch E-paper Display Module & mbed LPC1768. [Online, PDF document].
URL:http://www.embeddedartists.com/sites/default/files/support/displays/epaper/Epaper_mbed.pdf.
Last accessed 17 September 2014

12 Wikipedia, The free Encyclopedia. Electronic paper [Online]
URL:http://en.wikipedia.org/wiki/Electronic_paper. Page last updated on 27 September 2014. Last accessed 28 September 2014

13 AppNearMe. AppNearMe_MuNFC_PN532_Test [Online]
URL:http://developer.mbed.org/users/AppNearMe/code/AppNearMe_MuNFC_PN532_Test/
Last accessed 17 September 2014

14 Fukuchi Kentaro. Libqrencode [Online].
URL: http://fukuchi.org/works/qrencode/
Last accessed 01 September 2014

15 Peter Drescher. Epaper_mbed_test [Online].
URL: http://developer.mbed.org/users/dreschpe/code/epaper_mbed_test/
Last accessed 01 September 2014

Source code, NFCtest main.cpp

```
#include "mbed.h"
#include "rtos/rtos.h"

#include "MuNFC.h"

DigitalOut led_alive(LED1);
DigitalOut led_progress(LED2);
DigitalOut led_ok(LED3);
DigitalOut led_failed(LED4);
Semaphore s(1);

MuNFC nfc("00000003RdfgT390", 1, p11, p12, p13, p19, p18);

#define DEFAULT_PACKET 1

char inputText[24];


//Encode callback
void encode_cb(TLVList* tlv, void *)
{
  static uint32_t counter = 0;
  counter++;
  tlv->putUInt8( DEFAULT_PACKET ); //First uint8_t is packet type
  tlv->putString("Hello from mbed!");
  tlv->putUInt32( counter );
}

//Decode callback
void decode_cb(TLVList* tlv, void *)
{
  if(tlv->getNext() != UINT8)
```

```
  {
    return;
  }
  if(tlv->getUInt8() == DEFAULT_PACKET) //First uint8_t is packet type
  {
    if(tlv->getNext() != STRING)
    {
      return;
    }
    tlv->getString(inputText, 23);
  }
  s.release();
}

//NFC event
void event_cb(NFCEvent event, void*)
{
    switch(event)
    {
    case NFC_TRANSACTION_STARTED:
        led_progress=1;
        led_ok=0;
        led_failed=0;
    break;
    case NFC_TRANSACTION_SUCCESSFUL:
        led_progress=0;
        led_ok=1;
        led_failed=0;
    break;
    case NFC_TRANSACTION_FAILED:
        led_progress=0;
        led_ok=0;
        led_failed=1;
    break;
```

```
    }
}

int main() {
    nfc.encode(encode_cb, NULL);
    nfc.decode(decode_cb, NULL);
    nfc.event(event_cb, NULL);
    s.wait();

    bool ret = nfc.init();
    if(ret)
    {
      printf("\nAppNearMe/MuNFC stack initialized\n");

    }
    else
    {
      printf("Could not initialize stack\n");
    }

    nfc.run(); //Start thread

    while(1) {
        led_alive = !led_alive;
        if(s.wait(200) > 0)
        {
          printf("Got: %s\n",inputText);
        }
    }
}
```

Source Code, QREncode main.cpp

```
#include "mbed.h"
#include "EaEpaper.h"
#include "Arial28x28.h"
#include "Arial12x12.h"
#include "font_big.h"
#include "graphics.h"
#include "GraphicsDisplay.h"
#include "qrencode.h"

EaEpaper epaper(p9,           // PWR_CTRL
                p10,          // BORDER
                p11,          // DISCHARGE
                p12,          // RESET_DISP
                p13,          // BUSY
                p8,           // SSEL
                p26,          // PWM
                p5,p6,p7,     // MOSI,MISO,SCLK
                p28,p27);     // SDA,SCL

 // a function to output QR Code matrix on E-paper Display
 extern void QRdisplay(int a,int c, char *p)
{
    QRcode *qr; // pointer to QRcode
    //specify the data to be encoded, version number, error correction level,
    // data type, and case sensitivity
    qr = QRcode_encodeString(p, 6, QR_ECLEVEL_M, QR_MODE_8, 1);
    //make sure the width of QR Code is not zero
    const int s = qr ->width>0?qr->width:1;
    //an iteration for drawing the pixel from bitmap
    for (int y=0; y<s; y++){
        const int yy = y*s;
        for(int x=0; x<s; x++){
```

```
            const int xx = yy+x;
            const unsigned char b = qr->data[xx];
            if(b &0x01){
              epaper.fill(x+a,y+c,1,1,1);
               }
            }
         }
        }

int main() {

    epaper.cls();
    epaper.set_font((unsigned char*) Arial28x28);  // select the font
    epaper.locate(5,10);
    epaper.printf("Connect to Wi-Fi");

    QRdisplay(50,50,"WIFI:T:WPA;S:bunnu;P:pass123;H:HIDDEN;");

    epaper.write_disp();
    }
```