

NFC Passworder

Kalen Camacho, Matthew Gaimaro, Richard Ojo, and Jessica Shi

ENEE408A (Section 0101)

Professor Hawkins

Submitted: 5/19/21

Table of Contents:

Signature	2
Executive Summary	4
Main Body	5
Introduction	5
Goals and Design Overview	5
Realistic Constraints	6
Engineering Standards	7
Alternative Designs and Design Choices	7
Technical Analysis for Systems and Subsystems	9
Design Validation for System and Subsystems	17
Test Plan	19
Post Mortem	19
Salvage	20
Conclusions	20
Appendices	21
Bill of Materials	21
Pinout Diagram	22
Gantt Chart	23

Signature

Kalen Camacho

- NFC Expansion Board research and development
- Android Studio Bluetooth Application

Matthew Gaimaro

- Android Studio Bluetooth LE Application
- Connection between HM-10 and Android Phone

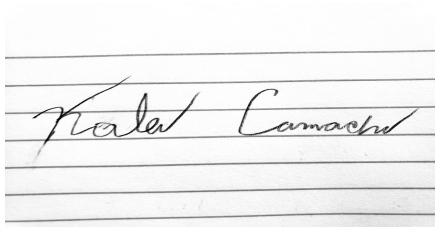
Richard Ojo

- Connection between the STM32 and the HM-10
- Connection between the STM32 and the X-NUCLEO-NFC06A1
- Configure and program the STM32H745ZI-Q

Jessica Shi

- Bluetooth module connection
- NFC tag research
- Encryption and decryption models

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination.



Kalen Camacho



Matthew Gaimaro

Richard Ojo

Richard Ojo

Jessica Shi

Jessica Shi

Executive Summary

This report details the research, design, and implementation of a local password manager device. The system was implemented over 10 weeks by students in ENEE408A at the University of Maryland.

The primary goal of this project was to fabricate a password manager with the functionality to read and write passwords stored on an STM32H745 board using NFC technology and Bluetooth. The system involves a user's phone (specifically Android), which is the primary interface to read and write data to the STM32 board. A user will connect to the board via a Bluetooth app and will either request to write to or read from the board's flash storage. A request will execute if and only if the user is authenticated to exploit the data on the device. This authentication is controlled by a unique ID stored on an NFC tag.

In achieving the goals of this project, the team relied on research and design preparations in Bluetooth Low Energy, NFC card reading, types of NFC, Flash programming, Android app development, and encryption/decryption schemes.

The team implemented the project by dividing responsibilities between members. Team members communicated weekly through activity reports and received feedback and guidance from the professor throughout the course of the semester. The final product of this project is detailed in the rest of this report.

Main Body

Introduction

The goal of our project is to create a password manager, which reads a password contained on a physical device and authenticates to a system if this password is valid. For our project we chose to use NFC tags as the physical devices which we read passwords from, and created an Android phone app to loosely authenticate to. The way our project works at an abstracted level is we read the data stored on a NFC tag. If this data is a valid password we communicate via Bluetooth from our microprocessor board to the phone app, indicating that a valid password has been received. We can also write new passwords to the NFC tags, by writing a new password in the phone app, passing this onto the microprocessor board which contains the NFC tag writer which then writes the new password to the NFC tag.

The system this project implements has several benefits, in contrast to manually inputting a password. A primary one is that it is quicker and more convenient to read a password from an NFC tag than manually typing out a password. Another benefit is that the user does not need to memorize their passwords, which provides convenience and allows for users to create complex passwords without having to remember them. NFC tags are very small and cheap, so it is very easy to have multiple tags for different authentication methods and to carry these tags around.

The functionality that this project brings could be very useful if integrated into other systems which require authentication. A simple example would be unlocking your computer, where you have an NFC tag which contains the password to unlock your computer and instead of using your password you can scan your tag onto the reader and unlock the computer. This simple concept could be applied to literally anything that you authenticate to, and thus has great potential benefits if integrated into these other authentication systems.

Goals and Design Overview

The initial goal of the project was to create a system which read the contents of an NFC tag and verified if the contents were the correct credentials to authenticate for a password. If the password is correct then the system should authenticate, and if the password is incorrect then nothing should occur. Over the course of the project, the goal became more and more specific as we began to flesh out our design. Eventually our goal became to specifically authenticate to a phone app, and to have this same phone app be able to write/rewrite passwords to these NFC tags. Over the course of this project our overarching goals were not really changed.

The design specifications at the start of the project were quite loose, mostly just starting with the base STM32H745 board, and over the course of the project the design got more fleshed out as more components were added to the project. The design process was a mixture of an initial design plan that was updated frequently throughout the development process. An example of this sequential updating was when needed to determine what application we wanted to authenticate

too, and when we determined that we needed to develop our own Android phone app we then began the design for that phone app. This was the biggest change during development, as it also came quite late in the process and became a blocker for the rest of development to move forward. We adapted to this with group meetings to discuss the phone app design, where this app was going to fit in with the rest of our project, and worked out how we were going to build and integrate it into what we already had.

Realistic Constraints

One of the most restrictive constraints on the project is security through obscurity. Since the main purpose of the NFC passworder is to securely store a user's personal passwords, the device must exhibit security during all stages of operation. The main level of security is that the NFC passworder is a physical device that would have to be taken in order to access the passwords.

The second constraint is cost due to the economical impact of the project. In order to be operable, there must be a microprocessor, NFC reader/writer, a bluetooth module, a mobile device, and NFC tags. Since the user must be connected to the application from a mobile phone, there will be an additional economic constraint added to the project. However, the target audience for this project would most likely have a phone that is able to run the software portion of the passworder. Additionally, another constraint is that the application software is only designed for Android devices since it was developed in Android studio. This could create a problem as many users, including most of the team, uses Apple devices.

A fourth constraint is storage capability and buffer size. In order to communicate between the hardware components and the application, an HM-10 device operates using Bluetooth LE (Low Energy). Since this is designed for low power dissipation, the maximum transmission size is roughly 20 bytes per transmission. In terms of storage size, each NFC tag will only be able to store one password at a time, and the flash memory of the STM32H745 is limited.

Finally, the last main constraint on the project is manufacturability since several hardware components must be combined in order to function properly. The STM32H745, NFC-06, and HM-10 must all be configured in order to manufacture the passworder. Additionally, there are major software changes that were made in order to properly transmit and receive data between the hardware and mobile application. Overall, the team believes that all of the constraints are reasonable and are adequately handled.

Engineering Standards

There are several IEEE standards that can be seen as guidelines for the team during the preliminary and design stages of the project. During the software development for the mobile application and the STM32, the IEEE829 and IEEE830 standards were incorporated into the design and protocols. IEEE829 is the engineering standard for Software Test Documentation which entails the process of testing software in several phases and recording the findings. The team would review all code through testing and demonstrations throughout the design stage. Additionally, IEEE830 is the engineering standard for Software Requirements Specifications which incorporates the software requirements necessary for the development of a software system. In order to follow this IEEE standard, the team proposed several drafts for the project which involved different hardware and software components. Then, the team was able to narrow down the specific requirements needed to perform NFC and bluetooth communication between the hardware and software. Overall, IEEE standards were used as suggestions throughout the hardware and software development in the project.

Alternative Designs and Design Choices

The first alternative design was the use of computer storage for the passworder. In this implementation, the NFC reader/writer and STM32H745 would be connected to a computer using a Micro-USB cable. Originally, the team believed that this would be a viable storage solution since an external storage device would allow roughly 16GB of storage for passwords and encryption. However, this idea was down selected as the team believed that a mobile application would provide a better user experience. Additionally, the flash memory of the STM32H745 is sufficient for storing multiple passwords and encryptions since each password is roughly 20 bytes. The computer storage design was crafted in order to combat the constraint for data storage using an external storage device. However, the team decided that a mobile application and flash memory would both improve the passworder's functionality.

A second design that the team considered was to solely communicate between the mobile device and the STM32H745 using NFC. In order to implement this design, the team proposed utilizing an application that uses NFC technology to interact with the NFC reader/writer on the NFC-06. One main reason that this design was down-selected is because of the security constraint on the project. Since the purpose of the passworder is to store a user's sensitive information, the overall design must be secure at all times. If the sole communication between the hardware and the application was NFC, there would not be an added layer of encryption, and anyone with an NFC reader would have access to a user's passwords. Therefore, the team decided that the better alternative was to communication via Bluetooth LE which would allow the user to communicate with only one device at a time. Overall, communication with NFC alone was a security threat to the system which was adjusted for a more secure approach.

The last design that was under consideration was to use the NFC-04 which is a programmable NFC tag. In this alternative design, the team decided to develop a mobile

application that would send a password to the STM32H745 which would update the programmable NFC tag. Then, the user would use their mobile device to read the NFC tag that has been updated. This idea was ultimately disregarded for two main reasons. Primarily, there would be a redundancy in the system of transmitting and receiving the same password from the phone. In the current design, the passwords are stored in the flash memory of the STM32H745 which eliminates the redundancy. The second reason is that the passworder needs to be able to both read and write to NFC tags, and the NFC-04 only has the capability to write to the programmable tag. Security and storage constraints were both considered when the team decided to build the passworder with the NFC reader/writer rather than the NFC-04. Overall, the team believes that the final design is an accumulation of the best ideas from the alternative designs created in the development stages of the project.

Criteria	Importance Weighting	External Storage	Computer Storage	App Interface	Computer Interface	NFC Alone	NFC + Bluetooth
Resources available	5	1	1	1	1	1	1
Cost to implement	2	-1	1	0	0	1	-1
Potential problems during implementation	5	-1	-1	-1/0	0	1	-1
Time to implement	5	0	0	-1	0/1	1	-1
Storage constraints	3	0	0	0	0	0	0
Team interest	3	-1	1	0/1	0	0	1
Equal distribution of work	5	0	0	0	0	0	0
Totals		-5	5	-2	10	17	-4

Technical Analysis for Systems and Subsystems

Mathematical Models

The most commonly used symmetric-key encryption scheme is the Advanced Encryption Standard (AES). Symmetric encryption is a type of encryption where only one secret key is used to both encrypt and decrypt information. By using symmetric encryption algorithms, data is converted to a form that cannot be understood by anyone who does not possess the secret key to decrypt it. Only an intended recipient can use the key to reverse the encryption algorithm and receive the message in its original form.

AES is the standard set by NIST under which the AES cipher has a block size of 128 bits and key lengths ranging from 128 to 256 bits. With block algorithms, a set length of bits are encrypted in blocks of data with the use of a secret key. One such blocking algorithm is the cipher block chaining (CBC) mode, which takes the first block of plaintext and exclusive-OR's it with an initialization vector (IV) prior to the application of the secret key. An IV is a block of bits that is required to allow block ciphers to be executed and to produce a unique stream independent from other streams produced by the same encryption key. The resultant block is the first block of ciphertext. Each subsequent block of plaintext is XOR'd with the previous block of ciphertext. Due to this XOR chaining process, the same block of plaintext will not result in identical ciphertext being produced.

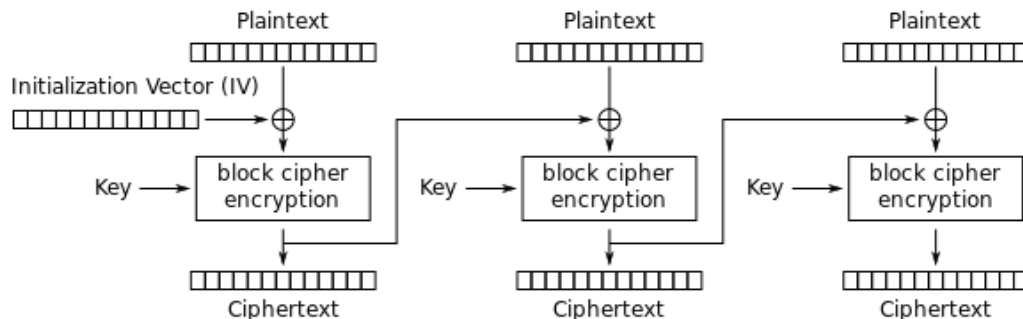


Figure 1: CBC mode encryption

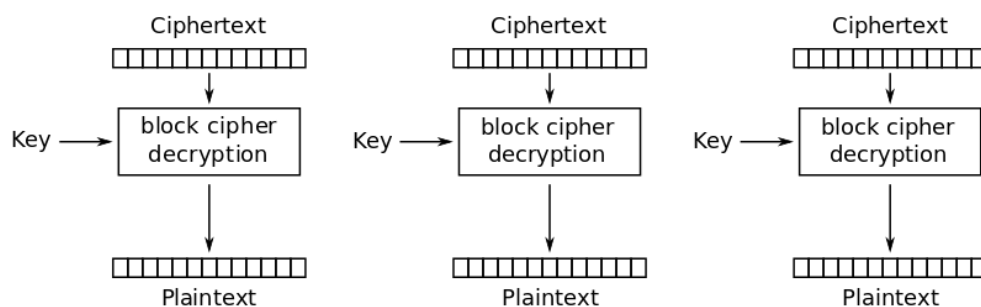


Figure 2: CBC mode decryption

Subsystems

The NFC Passworder can be interpreted as the culmination of four subsystems. These subsystems are the mobile application (software), the Bluetooth module (hardware and software), the encryption system (software), and the NFC system (hardware and software). The system uses STM32's blue push button to toggle between READ MODE and WRITE MODE.

The blue push button should be initialized with these parameters:

1. Pin – PC13
2. GPIO Mode – Input
3. GPIO Pull-down
4. User Label – USER_BUTTON

Subsystem Flow: Saving password

NFC → Bluetooth → Application → Bluetooth → Encryption

Subsystem Flow: Reading password

NFC → Bluetooth → Application

Mobile Application

The mobile application is developed to allow communication between the user's device and other Bluetooth-enabled devices. This includes the HM-10 Bluetooth module. The mobile application was only developed for the Android operating system and thus the application is only usable on Android devices. However, there are other applications available on the Google Play Store and the Apple App Store that allow the user's mobile device to communicate with the HM-10 and subsequently the STM32H745ZI-Q. For example, DSDTECHBluetooth and BLE Scanner. The mobile application scans for all Bluetooth devices in the surrounding area. When the HM-10 module is powered, it will appear on the app. The user can then connect to the HM-10 through the application.

*When scanning for Bluetooth devices, search for "DSD TECH" and ensure to turn off scanning to connect.

Bluetooth

The Bluetooth subsystem consists of the HM-10 Bluetooth module (hardware) and the STM32H745ZI-Q's USART6 data transfer interface (hardware and software). There are two types of Bluetooth: Bluetooth Low Energy (BLE) and Bluetooth Classic. The HM-10, however, communicates with the mobile application over Bluetooth Low Energy. When the HM-10 is in discovery mode, the HM-10's red led will blink repeatedly. Once the user pairs to the HM-10 through a mobile application, the HM-10's red led stays on and will stop blinking.

The HM-10 module communicates with the STM32H745ZI-Q using USART/UART.

USART6 is initialized with these parameters:

1. CM7 and CM4 Asynchronous mode
2. Global interrupts enabled
3. Baud Rate – 9600 Bits/s
4. Word Length – 8 Bits
5. Parity – None
6. Stop Bits – 1
7. Data Direction – Receive and Transmit

To achieve USART/UART communication, the HM-10 has four connections/pins: RXD, TXD, GND, and VCC.

VCC is connected to either the STM's (CN8) 5V or 3.3V pin.

The GND pin is connected to one of the STM's (CN8) GND pins.

RXD is connected to the USART6_TX GPIO pin (PC6).

TXD is connected to the USART6_RX GPIO pin (PC7).

Relevant Functions

HAL_UART_Receive_IT(huart, pData, Size) - Receive data over USART using interrupts

huart: UART handle.

pData: Pointer to data buffer (u8 or u16 data elements).

Size: Amount of data elements (u8 or u16) to be received.

HAL_UART_Transmit(huart, pData, Size, Timeout); - Transmit data over USART

huart: UART handle.

pData: Pointer to data buffer (u8 or u16 data elements).

Size: Amount of data elements (u8 or u16) to be sent.

Timeout: Timeout duration.

Encryption

Encryption should be used to encrypt and/or decrypt the tag id and the password that the user would like to store.

When the overall system is in READ MODE, when the correct tag is placed on the NFC module, the stored encrypted tag id would be read from FLASH storage (location 0x08040000). This encrypted tag id would be decrypted and compared to the tag id in question. If both id's match, then the stored encrypted password is read from FLASH storage (location 0x08060000). This encrypted password would be decrypted and transmitted to the HM-10 over USART6 and finally sent to the mobile application over Bluetooth, where it is displayed.

When the system is in WRITE MODE, the user sends the password that they would like to save to the STM32 over Bluetooth using the mobile application. The password must be sent with a trailing colon (:). For example, if the password to save is "DEMOPASS." The user must send "DEMOPASS:" in the application.

Encryption and decryption should be executed by the CM4. Thus, this project requires data transfers between CM4 and CM7. This is done with the shareMemCM7 and shareMemCM4 files contained within the project.

NFC

The Near Field Communication (NFC) subsystem consists of two parts: The X-NUCLEO-NFC06A1 NFC Card Reader Expansion Board (hardware) and the STM32H745I-Q's SPI1 data transfer interface (software).

The SPI1 should be initialized with the following parameters:

1. CM7 Full-Duplex Master
2. Frame Format – Motorola
3. Data Size – 8 Bits
4. First Bit – MSB First
5. Pre-scalar - 8 → Baud Rate – 12.5 MBits/s
6. Clock Polarity – Low
7. Clock Phase – 2 Edge

The X-NUCLEO-NFC06A1 NFC expansion board connects to the Nucleo-144's Arduino Zio connections. Three of these pins are initialized for SPI1 communication. The other eight pins are initialized as GPIOs that communicate with the NFC06.

SPI1

Pin	Signal	GPIO Mode	Pull up/down
PA5	SPI1_SCK	Alternate Func	No
PA6	SPI1_MISO	Alternate Func	no
PB5	SPI1_MOSI	Alternate Func	No

GPIO

Pin	Signal	GPIO Mode	Pull up/Down
PA3	IRQ_3911	Ext. Interrupt Rising edge triggered	No
PB1	MCU_LED3	Output push pull	No
PC0	MCU_LED1	Output push pull	No
PC2	MCU_LED4	Output push pull	No
PC3	MCU_LED2	Output push pull	No
PD14	SPI_CS	Output push pull	No
PF11	MCU_LED5	Output push pull	No
PG12	MCU_LED6	Output push pull	no

For the NFC module to operate correctly, it is recommended to use STM's example projects that are contained within the X-CUBE-NFC6 initiator software. NFC Passworder is based on "STM32L476RG-Nucleo_PollingTagDetectNdef", which is one of the provided example projects. This project was then modified, and additions were made to ensure proper functionality on the STM32H745ZI-Q.

Required libraries/ files

Drivers/Library/ File	Use
FLASH_SECTOR	Storing and reading data from FLASH storage
SharedMemCM7	Sharing data between CM7 and CM4
sharedMemCM4	Sharing data between CM4 and CM7
Ndef (NFC Data Exchange Format)	Initialize NFC6's ST25R3916 device and handles communication/data exchanges between the NFC6 and a tag/NFC device (STANDARD)
Rfal (RF/NFC abstraction layer)	STR25R device common interface. Handles communication/data exchanges between the NFC6 and a tag/NFC device
BSP (Board Support Package)	Based on HAL drivers. The BSP drivers allow quick access to the board services using high-level APIs, without any specific configuration as the link with the HAL.

Redesign Ideas

1. Cleanup the project code that the STM runs to reduce storage use and improve overall system speed.
2. Add code/state(s) that would allow users to store more than one password.
3. Add code/state(s) that would allow users to have more than one tag for authentication.
4. Use a Bluetooth Classic module instead of the HM-10. This would increase the system's overall speed as UART data transfers between the Bluetooth module and the STM would increase.
5. Improve/fix bugs contained in the mobile application because data is sometimes lost or misrepresented on screen.
6. Develop or encase the STM/NFC/HM-10 subsystem within a case for protection.
7. Allow the system to work with tags other than NFC Type A (NTAG215).
8. A simple encryption scheme could be used. For instance, bit shifting of characters to encrypt.

Shortcomings

One design shortcoming is that only one password can be saved at a time and only one tag can be used for authentication. This is a big limitation because if the user loses their authentication tag, then the system would need to be reset (WRITE MODE with a new tag, but the current password would be lost). This is because the authenticating tag id is stored in FLASH storage along with the password. An additional shortcoming is that the system is only configured to work with NFCA tags. However, this issue can easily be fixed by moving a few lines of code. Another shortcoming of the design is the lack of working encryption and decryption. This decreases the amount of password protection that the overall system provides. This also means that the program that the STM runs has unnecessary code (that transfers data between CM7 and CM4). Another shortcoming is the HM-10 because of the module's data transfer limitations. The HM-10 utilizes Bluetooth Low Energy. This means that data transfers are slower (1Mbs or 2 Mbs) than Bluetooth Classic speeds (3 Mbs). Furthermore, could not configure the HM-10 and therefore improve UART data transfer speeds. This is because the HM-10 did not respond to the appropriate USART commands. This meant that we could not increase the HM-10's UART baud rate from 9600 to 115200 Bits/s.

Sources of error

Data bits tend to be corrupted or dropped during USART exchanges. This could cause the incorrect password to be saved or displayed. This could also cause incoherent messages to be sent and subsequently displayed to the user. The HM-10's BLE limitations can also cause these issues. Another source of error could be a potential security breach. If someone were to read the tag's id, then that person could read the password stored on the STM. However, the password thief would still need to have the correct tag, and the STM/NFC/HM-10 subsystem to achieve this. Also, if a user were to lose their authenticating tag, they would be required to reinitialize the system with a new tag, which would erase their stored password.

Design Validation for System and Subsystems

The design has two purposes/validation criteria:

1. WRITE - allow a user to save a password using an NFC tag as an authenticator by sending the password over Bluetooth.
2. READ - allow a user to see their saved password on their mobile device after authenticating themselves using a valid authenticating NFC tag.

RESET MODE (No password saved)

When no password is saved, this also implies that no tag id is saved; thus, there is no authenticating tag yet.

RESET MODE READ

When reading from the device and there is no password saved, then the STM's green led will flash three times, but the system will display the message "No password found."

WRITE MODE

The overall system can store a single password. During this mode, all blue LEDs on the NFC6 will blink until a tag is detected or the timer resets the system back into READ MODE.

***WHEN WRITING A TAG, KEEP THE TAG ON THE READER UNTIL THE SAVED PASSWORD IS DISPLAYED BACK TO THE MOBILE APPLICATION.**

READ MODE

The overall system can read the saved password back.

Valid Tag Detected

When a valid authenticating tag is tapped on the NFC reader, the STM's green led will blink three times and the system will display the message "Success! Here is your password:" along with the saved password. This indicates that the tag validation and FLASH reading procedures succeeded.

Invalid Tag Detected

When an incorrect tag is detected, the STM's red led will blink three times and the system will display the message "INVALID TAG!" This indicates that the tag validation and FLASH reading procedure failed.

Table of specifications

Subsystem	Requirement	Requirement met? (Y/N)
Mobile Application	Receive data from and send data to HM-10. Display data received from HM-10	Y
Bluetooth	Send data between HM-10 and Mobile App	Y
USART/UART	Send data between HM-10 and H745	Y
Encryption/Decryption	Encrypt and Decrypt data	N
NFC	Read and Write to NFC tag	Y
SPI	Send data between NFC6 and H745	Y

Test Plan

Testing the project was a staggered process, where when each component was developed we integrated it into the currently existing project tested for functionality and workability once implemented. Once we had completed all of the components to our project, testing was not too complicated. As done in our demo video, we are able to write passwords to these NFC tags and then read back what was written. We read an NFC tag with a valid password and an invalid password, and then checked the output of the verification on the phone app to make sure it is correct. We also needed to test out the writing capability of our project, which was done by writing to a specific NFC tag and then subsequently reading that tag to make sure that what we intended to write was on the tag. The final iteration of the project passed all of these tests that we performed and so we deemed the project to fulfill the functionality layed out in this report.

Post Mortem

Post-mortem discussion is an essential in digging deep into key accomplishments, problem areas, ongoing development, and maintenance needs of any project. To start off this reflection, we start with a project overview. The objectives of this project were to convert a general problem description in microprocessor-based systems into a design specification, split up design specifications into a set of design tasks, work in a team to implement design tasks, and construct a working prototype. Our initial proposal of the project discussed using local storage over cloud services to store sensitive data. We presented the use of NFC technology, Bluetooth, and the essential STM32 board to fabricate a password manager device. In later revisions to the project, we additionally tacked on a custom Android Bluetooth app and made an endeavor to incorporate more security measures with hardware encryption and decryption. Overall, the team was met with a lot of success in meeting the initial project goals. Even so, there existed some shortcomings and consequent need for revision.

Project Planning

Planning is the foundation of any project; thus, it is valuable to dig into the initial process. Admittedly, the plan changed multiple times due to confusion about the overall system flow. This confusion was attributed to a misunderstanding or lack of documentation on two parts we ordered for the NFC portion of the project. Having to recuperate and buy new parts consequently caused a wastage of time, money, and resources.

Execution

The team spent a good few weeks waiting for parts to come in, which put a dent on time to incorporate additional features in the later revisions to the system. One particular task that proved to be more difficult to complete was the encryption and decryption of data. The plan for a write operation was to send password data from the app to CM7, pass the plaintext from CM7 to CM4, encrypt the data in CM4, and store the ciphertext in flash. The process for read operations

was to decrypt the ciphertext stored in flash in CM4, pass the plaintext from CM4 to CM7, and transmit it through to the app. A big reason why this endeavor did not work to our favor is because the cryptography library wolfSSL/wolfCrypt does not support the H745 board. For STM32 microcontrollers (e.g. STM32H753) that have hardware crypto acceleration, wolfSSL/wolfCrypt supports it.

Salvage

As discussed in the Post Mortem section of this report, a subsystem of the project that is currently unused is the encryption and decryption schemes. In order to salvage this portion, unfortunately, the only way is to reconfigure the project to operate on the STM32H753 board. While this security feature is not majorly important to the overall functionality of the project, the confidentiality to ensure protection of data is a legitimate design possibility.

Conclusions

By and large over the course of our project we have been able to stick to and implement the original project goal of authentication using an NFC tag. The project that we made reads and validates the passwords on these NFC tags, and determines whether the password is valid or invalid, completing the general goal for the project. The development process for the project was generally quite sequential, with each element that was added being added one after another, with this strategy having its benefits and uses. The benefit is that the problems of integrating all of the components of the project together was relatively minimal, each component added was able to work the existing project without too much issue. However this development process also meant that the pace of the development could be perhaps a bit slower than desired at times, and that the lack of progress on one component could delay the development of the subsequent component.

The project faced several challenges over the course of the development process. Some of the lessons learned by the group in response to these were that frequent and clear communication of expectations and the current plan are very important for staying on track during development. Another lesson learned by the group was in the design stage for the project. The initial design that we had for the project was too vague and was not clearly laid out to carry out the project, and the project design had to be analyzed and changed significantly throughout development. However the project came together nicely in the end, with the initial stated goals of the project being completed.

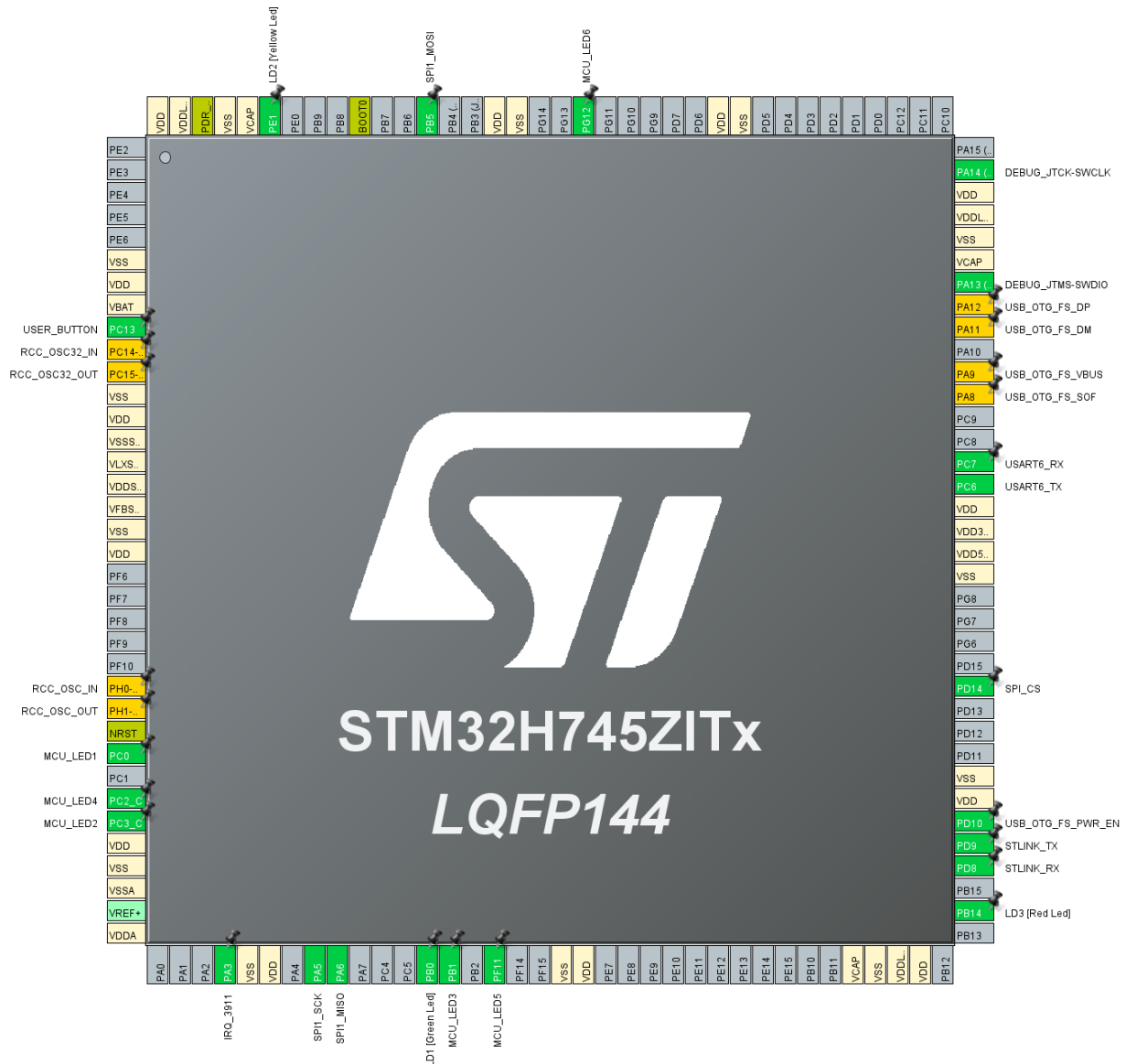
The project is an effective proof of concept which could be extremely prevalent and useful if implemented into other authentication systems. The general idea of the project can be integrated into any application which uses authentication, making it quicker and more convenient to authenticate. The project shows how to easily authenticate using a NFC tag, the next step going forward with the project is to make the project portable enough to implement into other applications and integrate the project into these new applications.

Appendices

Bill of Materials

Part + Link	Price	Quantity
DSD TECH HM-10 Bluetooth LE Module	\$11.49	1
NTAG215 PVC Tags	\$9.99	1
X-NUCLEO-NFC06A1	\$20.00	1
STM32H745	\$29.00	1
Total	\$70.48	

Pinout Diagram



Gantt Chart

	1/28	2/3	2/10	2/17	2/23	3/9	3/16	3/23	3/30	4/6	4/13	4/20	4/27	5/4	5/19
Phase 1 - Brainstorming Project Ideas															
Phase 2 - Project Proposal															
Phase 3 - Costs and Benefits Analysis															
Phase 4 - Research and Organization															
Phase 5 - Bluetooth Module + STM32H745															
Phase 6 - NFC Card Reader + STM32H745															
Phase 7 - Android App Development															
Phase 8 - Documentation and Demo															