지문인식 결제 시스템
개발 보고서

이연준 <yeonfish6040@dimipay.io>

Rev 0.1 (초안)

목차

Version 1 초기설계

본 지문결제시스템은 2026학년도부터 DIMIPAY가 서비스되는 학교인 한국 디지털미디어 고등학교에서 초·중등교육법 제 20조의 5에 근거하여 교내 정규수업시간에 스마트폰의 소지를 제한함에 따라 얼굴인식 같이 핸드폰 없이 결제할 수 있는 방법을 하나 더 추가하자는 의미에서 개발이 시작되었다.

요구조건

- 결제 Kiosk로 iPad를 이용중이었고 하드웨어 개발 후 MFi인증을 받을 수 없어 열려있는 API인 Bluetooth를 이용해야한다
- 지문 모듈이 사용자를 인증하는 것이 아닌 이미지를 제공하여 사용자 인증은 서버에서 진행되어야한다
- 개인정보 보호법 제 21조에 의거하여 개인정보의 처리 목적 달성 및 개인정보 보유기간의 경과등 개인정보가 불필요하게 되었을때는 파기하여야한다
- 개인정보 보호법 제 29조에 의거하여 개인정보 보관을 이행할때 내부 관리계획을 수립하고 접속기록을 보관하는 등 기술적 • 관리적 및 물리적 조치를 하여야 한다
- 개인정보보호위원회의 생체정보 보호 가이드라인에 따라, 원본이 아닌 미뉴티아 추출 후 저장, 지문 이미지 전송구간 암호화가 이루어져야 한다
- 지문을 읽어들이기 시작한 시점으로부터 결제 토큰 발급까지 가급적 2초 이내에 처리되어야한다

지문센서의 선정

원활한 통신 및 인증을 위하여 다음중 하나에 해당하는 센서가 필요했다.

- 시중에서 구매할 수 있는 지문센서중에 USB를 이용하며 프로토콜을 공개하는 센서
- UART, I2C, SPI등 통상적인 MCU가 지원하는 통신 프로토콜을 지원해야한다.
- 높은 신뢰성을 위하여 지문 유사도 Threshold를 높였을때 원활한 인증을 위하여 넓은 유효면적이 있어야한다.

해당 요구조건을 탐색하던 중, 2개의 지문센서를 선정하였다.

HI-LINK ZW0608 – UART 지원, 유효캡쳐면적 12.05*12.05mm^2

ZKTeco LIVE20R – USB 2.0 지원, 유효캡쳐면적 15.24*20.32mm^2

ZKTeco의 제품으로 모든 지문센서를 대신하고싶었으나, USB 프로토콜을 공개하지 않고 Window, Android, Linux SDK용으로 제공하여 RaspberryPi에 Android를 기반으로하는 LineageOS를 올려 등록용으로 사용하기로 하였고,

Live20R에서 얻은 넓은 이미지로 ZW0608에서 얻은 국소이미지의 Score를 높히려고 시도하였다.

알고리즘의 선정

신뢰성있는 결제 시스템 구성을 위하여 요구되는 사항은 다음과 같다

- 신뢰성이 입증된 알고리즘일 것
- 템플릿 추출이 가능할 것
- 빠른 매칭이 될 것

그래서 다음 2개를 한번 두고 비교해보았다.

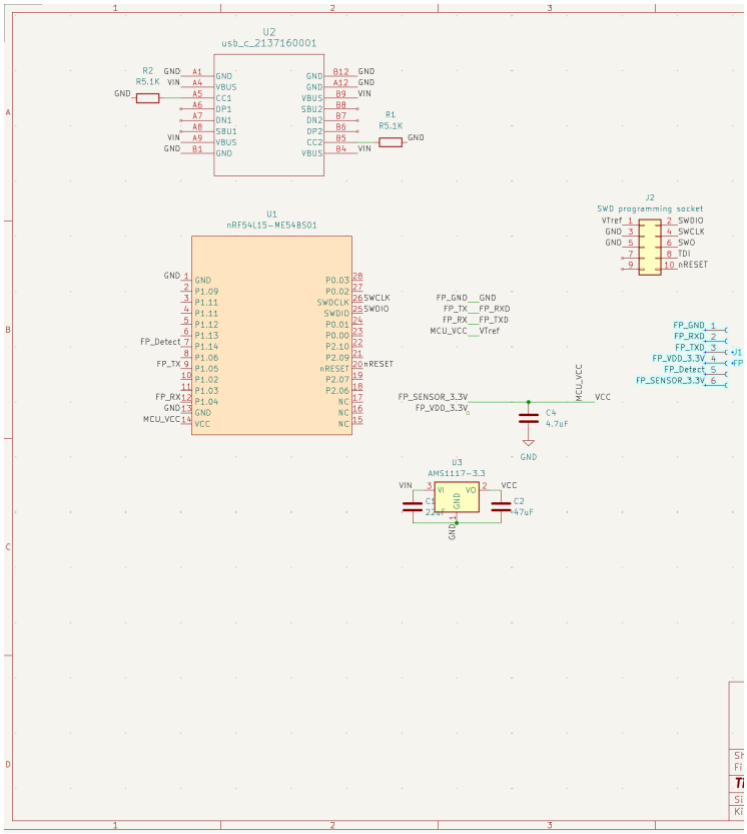
1. NFIS – NIST에서 만들었다. FBI가 쓴다고 한다. 근데 코드를 직접 빌드해서 써야하는데 너무 레거시시스템이라 사용중인 환경에서 빌드시 오류가 많이 나왔다. 코드보고 빌드 직접 하나하나 고치기에는 너무 코드가 많았다. 하지만 C로 작성되어 굉장히 빠르고 템플릿 저장도 가능했기에 도커를 이용하여 빌드한 이후 테스트를 해봤지만 지원하는 최소 이미지 크기가 ZW0608이 제공하는 이미지 크기보다 컸기 때문에 반려되었다.
2. SourceAFIS – Score를 조정하여 FAR를 근사치로 직접 조정할 수 있고 템플릿을 따로 저장 가능하다. 또한 매칭 시간이 굉장히 짧아 구상하고 있는 프로젝트에 이상적이었다. 현재 사용중인 백엔드가 Bun 런타임을 기반으로하기에 JAVA와 .NET 구현체만 존재하는 SourceAFIS 공식 라이브러리를 사용하기는 어려웠다. JS로 재구현하자기에는 깡 연산이 JAVA보다 느리고 포팅에 드는 시간과 노력이 너무 많기에 JAVA 웹서버를 하나 더 띄워서 내부망에서만 접근 가능하게 구성하는 것으로 대체하였다.

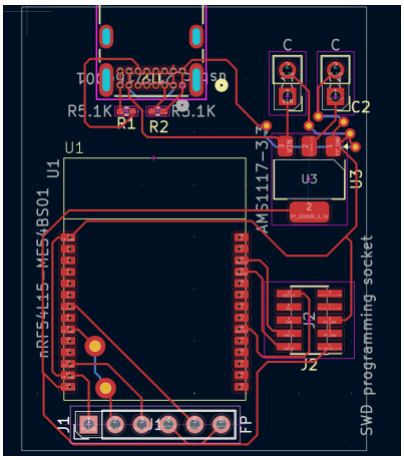
이러한 이유로 결국 SourceAFIS로 알고리즘을 선정하여 지문 비교, 등록, 삭제를 지원하는 웹서버를 만들어서 내부망에 배포하였다.

알고리즘의 최적화

최적화라기에는 다소 빈약하기는 하나, 지문 템플릿은 DB에 저장하여 관리하여야하고, 당연히 매 조회마다 DB에 쿼리를 날려서 비교할 수는 없으니, 캐싱을 진행하였다. 초기 웹서버 실행시 지문 데이터를 전부 조회하여 user id 따로, 미뉴티아 데이터 따로 메모리에 저장하여 해당 캐시에서 비교를 수행하였다. 런타임에서의 지문 DB 업데이트에는 DB 트렌젝션 수행 후 수행 결과에 따라 cache에 락을 건 이후 등록 혹은 삭제 작업을 수행하여 DB와의 안전한 데이터 동기화를 진행하였다.

하드웨어





앞서 요구조건에서 서술하였듯, 지문센서와 아이패드는 Bluetooth를 이용하여 통신하여야한다. 이를 위하여 Bluetooth가 integrated된 mcu를 사용하는 것이 이상적이라고 판단하여 최근에 배우고있고, BLE가 내장되어있는 nrf54l15를 사용하기로 하였다. 하지만 nrf54l15보드를 설계하기에는 당장 3학년 1학기 개학이 2주 채 남지 않은 시점에 교내 인트라넷 개발 마감이 최우선순위이여서 시간이 부족한 관계로 MINEWSEMI사의 nrf54l15-ME54BS01 모듈을 이용하여 설계하였다.

통신 구조는 SourceAFIS <-> Internal http <-> Core backend <-> HTTPS <-> ipad <-> BLE(encrypted) <-> nrf54l15 <-> UART <-> ZW-0608 이다.

ZW-0608의 손가락 인터럽트 라인을 nrf54l15에서 구독하여 인터럽트가 발생하면 ipad의 키오스크 앱에 해당 사실을 알려서 만약 키오스크가 상품 결제단계에 있다면 이미지 전송 요청을 발송하여 nrf54l15가 zw-0608에 이미지 capture 이후 버퍼에 있는 이미지를 nrf54l15에 업로드한다 해당 이미지를 다시 nrf54l15가 ipad로 전송하여 ipad는 이 이미지를 서버에 전송하여 서버가 신원을 확인하게된다.

프로토타입 생산 이후 문제점

느린 속도

평균적으로 인증이 완료되는데 5.8초가 걸렸다. 시간을 세세히 분해해본 결과 이미지를 센서에 업로드하는데 평균 3.8초, BLE로 전송되는데 평균 1초, 서버에 전송해서 클라우드 플레어 프록시를 통하여 응답이 돌아오는데 600~800ms정도로 오랜시간이 걸렸다. 전송단계에서 압축 또는 최적화를 통하여 최대로 전송속도를 확보한다고 해도, uart baud rate제한으로 3.8초 이하로는 전송속도를 개선할 수 없었다.

국소 이미지로 인한 FRR 증가

Live20R로 얻은 큰 이미지 덕분에 평균적인 매칭 스코어는 50점내외로 올릴 수 있었으나 사람마다 지문 퀄리티의 차이가 있어서 균일한 정확도를 확보하기 어려웠고, 상기한 느린 속도로 인하여 여러번의 재시도를 통한 결제가 빠른 순환이 필요한 점심/저녁의 피크시간대의 매점에서는 현실적으로 어려웠다.

PCB 설계상의 문제

UART의 RX와 TX를 반대로 매핑하여 주문하였다. 간단하게 센서와 보드가 연결되는 선을 바꿔끼움으로써 해결할 수 있었다.

2 layer pcb에서 back layer에 gnd pour을 하지 않은 상태에서 VBUS 5v에서 VDD 3.3v 로 바뀌는 LDO 전원블록에서 gnd를 이어주지 않았었다. Gnd Pour를 하지 않고 gnd via만 추가하여서 생겼던 문제였다.... 손납땜으로 gnd를 연결해주어서 해결하였다.

프로젝트를 마친 후 돌아보며

여러 생각을 했다. 우선 원래는 dk에서 zw-0901로 테스트를 완료하고 급하게 PCB를 시

키고 테스트를 하던 도중에 다른 사람 지문에서는 인식률이 떨어져서 ZW-0901에서 ZW-0608로 급하게 변경한터라 전송속도를 테스트해볼 수 없었고, 데이터양으로 전송속도를 가늠할 수 있었지만 트러블슈팅 과정이었기에 여러가지를 따져보지 못하고 급하게 때우게되어 문제가 발생했던게 참 안타까웠다.

또한 PCB 설계시 IFA 근처에 KEEP OUT영역이 필요한데 해당 KEEP out이 지켜지지 않았고, 안테나 바로 아래에 3.3V 전원라인이 지나갔다. 참으로 제대로 된게 없는 설계였으나 불행중 다행으로? GND pour를 잊어버려서 RF가 동작하지 않는 참사는 일어나지 않았다.

심지어 PCB를 수제작할 생각이었지만 주문생산하게 되어, 수제작에 맞춰 디자인한 VIA 크기가 PCB 주문제작에 그대로 유지되어 비아크기가 비정상적으로 커졌다. 또한 너무 급격하게 꺾인 트레이스가 많았다. 고속통신 트레이스등 임피던스에 민감한 트레이스들이 없었지만 좋은 설계는 아니였다.

조금만 더 여유있게 제작했다면 프로토타입을 좀 더 완성도있게 구성할 수 있었을텐데라는 아쉬움이 있었다

V2. 완제품

요구조건은 V1에서 "12MHz 이상의 SPI통신을 지원하는 지문센서"라는 조건이 추가되었다.

지문센서의 선정

시중에서 구매 가능한 지문센서로는 한계가 있다는 것을 확인하고 지문센서 제조업체를 찾기 시작하였다

CAMABIO사의 제품은 지문센서로는 이상적이었으나 UART나 USB밖에 지원하지 않았고 USB프로토콜 또한 공개되어있지 않았다.

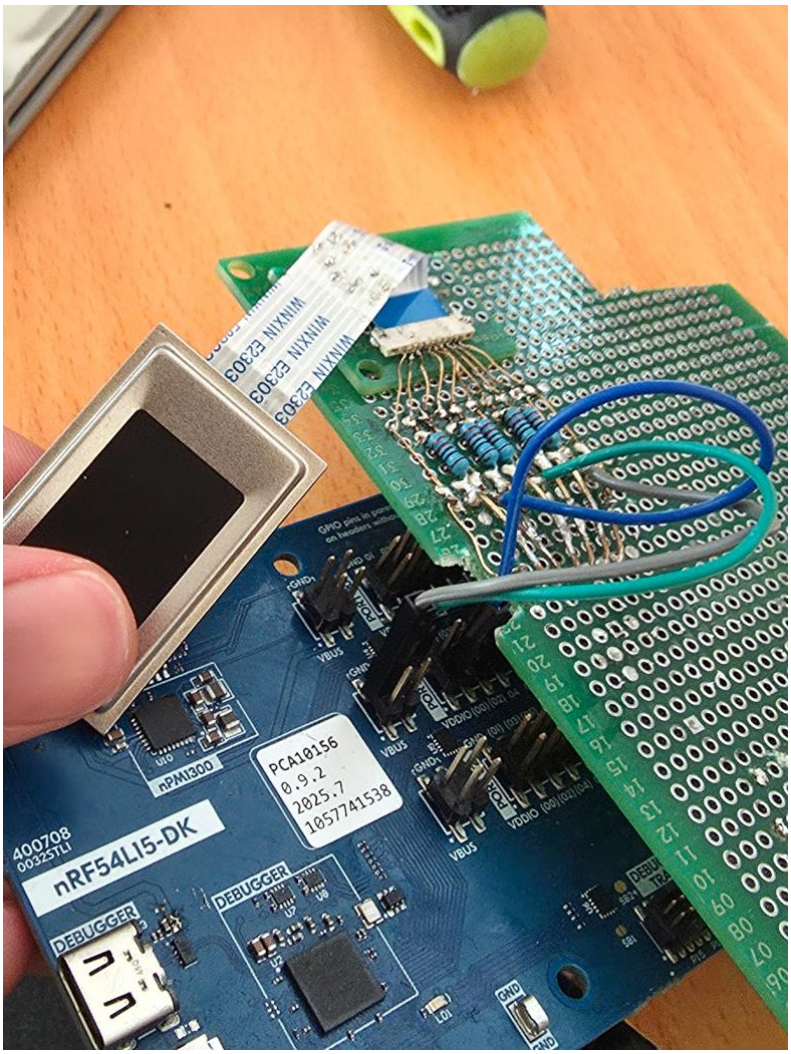
Midas Touch사의 MFC-1208GD 제품은 후술할 ADH-TECH사의 HF302GD보다 픽셀수가 부족하였다

Aratek사의 A400-M 센서는 동작 전압이 MCU와 일치하지 않아 추가적인 설계가 필요하

기에 탈락하였다.

ADH-TECH사의 HF302GD가 최종적으로 선택되었다. SPI 16MHz 통신이 가능하고 지문 이미지를 받을 수 있으며, 256*360에 508 dpi로 넓은 센서면적과 높은 해상도로 적합하였다. 지문 캡쳐도 최대 9fps로 빠른 촬영이 가능했다.

하드웨어 POC



nrrf54l15dk를 이용하여 하드웨어 POC를 제작하였다. ADH-TECH사에 이메일로 컨택하여 하드웨어 샘플과 테스트보드를 구매하여 수령한 이후, 테스트보드에서 FPC 커넥터를 뽑아다가 만능기판에 납땜하여 전원과 SPI 데이터라인을 구성하여 nrf54l15dk에 연결하여 하드웨어 POC를 제작하였다.

펌웨어 설계

지문센서 통신

지문센서 데이터시트[i]에 커뮤티케이션 프로토콜에 대하여 자세한 스팩정의가 되어있지 않아, 제조사에서 제공한 AS608칩이 이용된 테스트 보드의 소스[ii]를 보고 프로토콜을 유추/복구하여 펌웨어를 설계하였다. 초반 테스트보드 소스 없이 구현할때 레지스터 테이블에 각 레지스터별 정확한 설명이나 라이프사이클이 명시되어있지 않아서 개발에 어려움이 있었다.

블루투스 통신 인터페이스

NUS위에서 소규모 커스텀 바이너리 프로토콜을 이용한다. 다음은 패킷 구조이다.

Host -> Sensor 패킷 구조

Offset	**Size**	**Name**	**Description**
0	1	Command	명령코드
1	1	EOF	End of Packet (0x0A)

Commands

ID	**Command**	**Description**
0x00	BT_COMMAND_FINGERPRINT_RQ	지문 캡쳐 요청. 캡쳐 및 전송을 모두 포함한다
0x01	BT_COMMAND_FINGERPRINT_ABORT	지문 캡쳐 요청을 중단하고 관련 상태값을 초기화한다
0x02	BT_COMMAND_RESET	MCU Cold Reboot.
0x03	BT_COMMAND_STATERESET	모든 상태값을 초기화한다. 어찌보면 Soft Restart라고 할 수도 있다
0x04	BT_COMMAND_SECURE_RQ	BLE 보안연결 성립 요청이다. 보안연결이 성립되지 않은 상황에서는 0x00 명령이 후술할 BT_ERROR_LOW_SECURE_LEVEL 에러를 반환한다.

EOF는 BLE serial terminal에서 디버깅의 편의를 위해 넣었다.

Sensor -> Host 패킷 구조

Offset	Size	Name	Description
0	1	Packet Type	패킷의 종류이다. 자세한내용은 아래의 표를 참조.

1-2	2	Packet Length	뒤에 올 본문에 총 길이이다.
3-N	N	Body	패킷 본문

패킷 타입

ID	Command	Description
0x00	BT_PACKET_STATE	지문센서 상태종류에 대한 패킷이다. Finger interrupt와 ble param update등을 서빙한다.
0x01	BT_PACKET_FINGERPRINT_IMG_HEADER	지문 이미지 헤더이다. 차원과 bpp 정보를 서빙한다.
0x02	BT_PACKET_FINGERPRINT_IMG_DATA	지문 이미지 데이터이다.
0x03	BT_PACKET_FINGERPRINT_IMG_END	지문 이미지 엔드 플래그이다. 에러가 발생했을때 호스트가 Listen에 묶여있는 상황을 방지하기 위하여 추가되었다.
0x04	BT_PACKET_ERROR	에러 패킷이다. 에러 종류는 아래의 표를 참고.

에러 타입

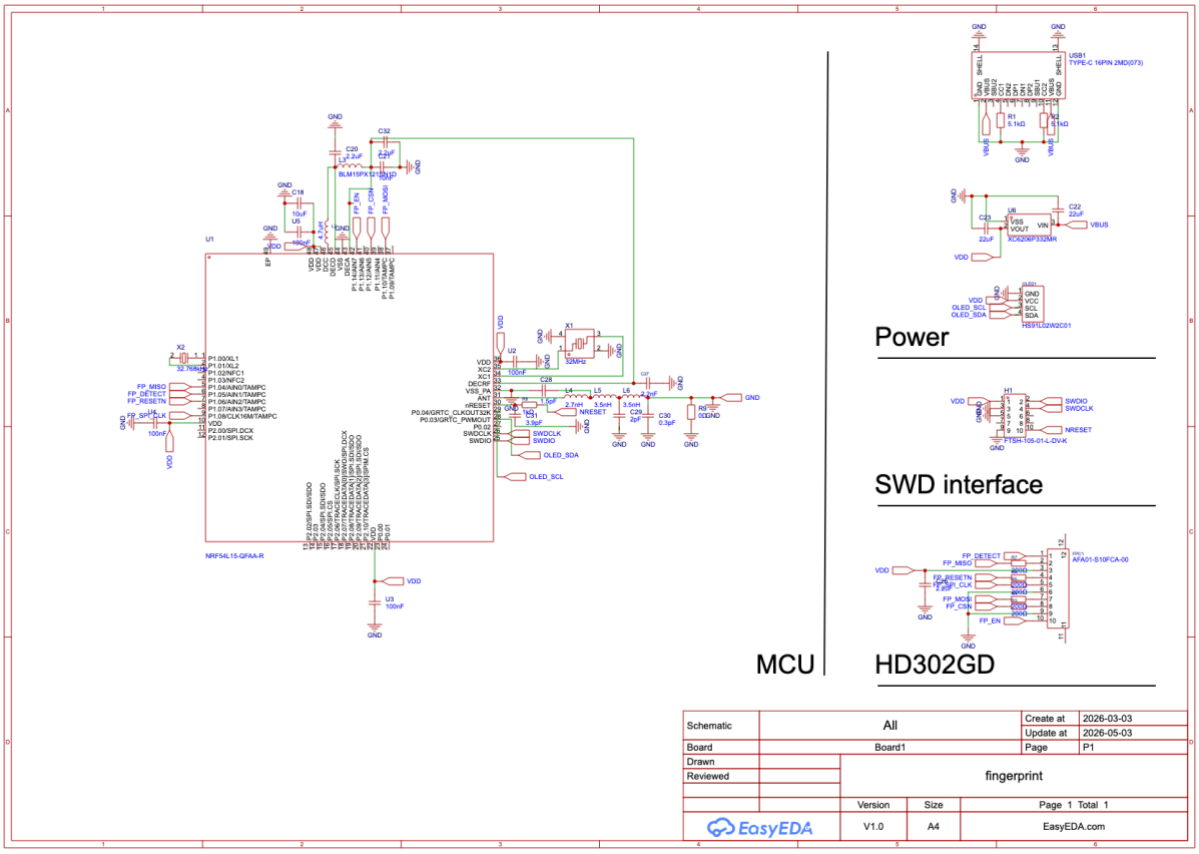
ID	Command	Description
0x00	BT_ERROR_BUSY	캡쳐 요청에 대한 동작이 진행중일때 캡쳐 요청이 한번 더 오면 발생한다.
0x01	BT_ERROR_INVALID_PACKET	정의된 구조와 일치하지 않는 명령 패킷이 들어왔을때 발생한다.
0x02	BT_ERROR_INVALID_COMMAND	정의되지 않은 명령이 들어왔을때 발생한다.
0x03	BT_ERROR_INTERNAL	알 수 없는 에러이다. 패킷 뒤에 딸려오거나 SWD Interface 또는 OLED에 표시된 에러코드를 참조해야 한다.
0x04	BT_ERROR_DEBUG	디버깅용 에러 플래그다.
0x05	BT_ERROR_LOW_SECURE_LEVEL	보안연결이 성립되지 않은 상황에서 캡쳐를 요청하면 발생한다.
0x06	BT_ERROR_CAPTURE_ABORTED	호스트 명령으로 캡쳐가 끊기면 발생한다
0x07	BT_ERROR_SENSOR_TIMEOUT	센서 타임아웃시에 발생한다. 현재 이용되지 않는다.
0x08	BT_ERROR_SENSOR_IO	데이터라인 불안정등의 이유로 이미지 데이터가 깨지면 발생한다.

이미지 처리

이미지 데이터는 260*320 1bpp이미지를 제공한다. 원래 센서 자체는 8bpp 이미지를 제공하나 전송속도의 개선을 위하여 1bpp로 변환하여 전송한다.

또한 1bpp로 변환하며 지문 이미지의 검은 픽셀(융선)이 전체 픽셀수의 25~35%정도 차지하게 threshold를 동적으로 조절한다. 퍼센티지에 따른 지문 이미지를 비교해본 결과 25%에서 좀 끊긴 융선이 보이지만 가장 이미지가 깨끗했고 35%에서는 융선이 다소 굵지만 솔리드하게 나타났다. 지문센서에 얕게 지문이 접촉했다면 25%까지 끌어올려서 지문 이미지를 살리고 과하게 접촉해서 융선이 너무 두꺼워진다면(이때는 보통 40~50%정도) 35%까지 끌어내려 융선 이미지를 살렸다. 해당 시스템을 도입하기 전에 고정 threshold를 이용해서 서비스하였는데, 사람마다 지문의 명확도 편차로 인하여 인식률이 떨어져 해당 시스템을 적용하게 되었다. 적용 후 인식률이 비약적으로 올라갔다.

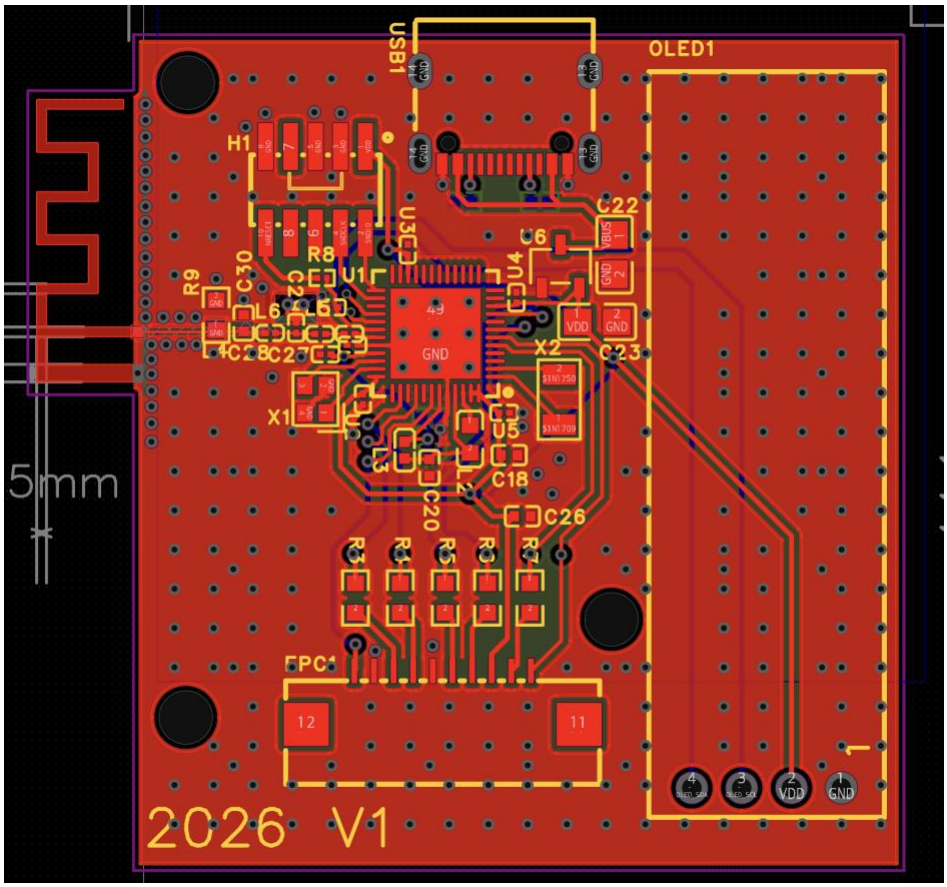
하드웨어 설계



해당 시점에는 시간과 예산이 충분한 관계로 nrf54l15 보드를 직접 설계하여 pcba를 맡기기로 하였다. 또한 EasyEDA를 이용하면 주문시 할인이 되기에 원래 사용하던 Kicad가 아니라 EasyEDA로 설계하게 되었다. Nordic semiconductor사의 nrf54l15 데이터시트에 나와있는 Reference Circuitry[iii]를 참고하여 디자인하였다. 전원부와 필터, RF 서킷은 신뢰성있는 동작을 보장하기 위하여 해당 데이터시트에 있는 BOM을 그대로 차용하여 사용

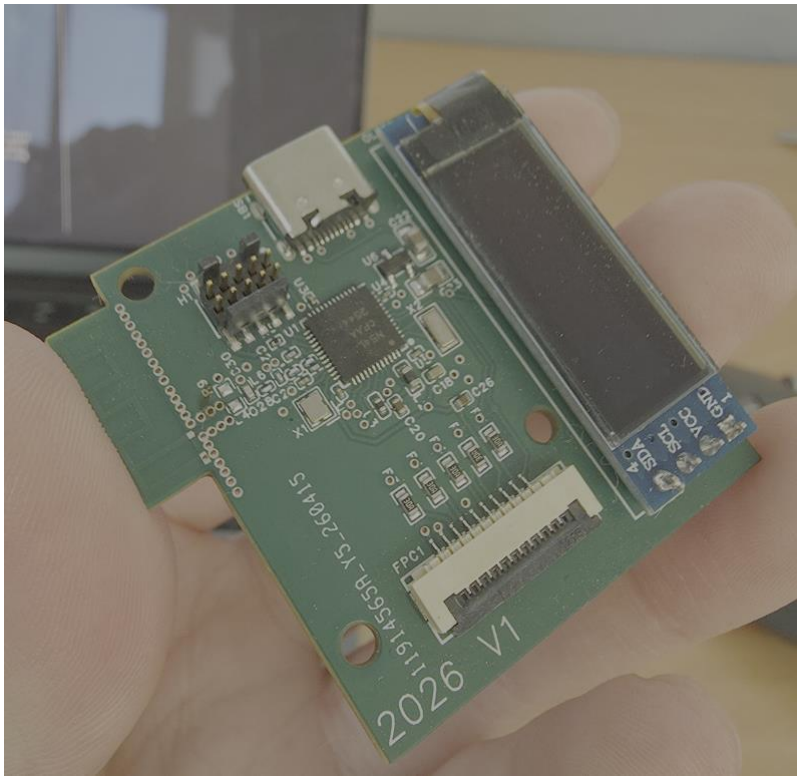
하였다. 2.4GHz 안테나는 간단하게 해결하기 위해 TI AN043 Application note[iv]를 보고 IFA를 서킷에 그대로 옮긴 후, Nordic semiconductor의 Q&A를 통하여 설계를 검증받았다. 지문센서와 MCU간의 SPI 통신라인에는 지문센서 제조사측 데이터시트에 200Ohm의 저항을 넣었다. 해당 설계를 공부하면서 나온 Quarter wavelength monopole 안테나와 거기서 확장해서 IFA안테나, 그리고 DCDC서킷에 이용되는 외부 인덕터 감지, SPI라인의 200Ohm에 대한 추가 연구는 후술된다.

또한 본 제품 개발인원이 항상 SWD interface를 통하여 BLE 인증번호를 확인해서 페어링할 수도 없고 매점에 상주하여 해당기기를 관리할 수 없음으로 매점 상주 인원이 쉽게 페어링, 상태확인, 트러블슈팅 및 개발팀을 위한 오류보고를 할 수 있게 보드에 OLED 모듈을 하나 추가하여 상기한 내용이 표시될 수 있도록 구성하였다.



전원은 보드 상단의 USB C 포트를 통하여 인가받으며, CC핀에 5.1k 저항을 달아 USB 2.0 스팩으로 5V 전압을 VBUS 라인에 받는다. 해당 전압은 LDO를 통하여 3.3V로 전환되어 VDD라인에 공급되며 LDO 전후단에 벌크 커페시터를 위치시켜 공통 전원라인에 버퍼를 만들어주었다.

다음은 PCBA 주문을 통하여 받은 보드이다.



인프라 설계

지문 DB는 개인정보보호위원회의 생채정보 보호 가이드라인에 따라, Dimipay 서비스의 다른 DB와 다르게 분리된 다른 독립 DB로 구성되며, AWS RDB를 이용한다. DB에 저장되는 정보는 암호화되며, DB의 인바운드 트래픽은 디미페이 서버 IP로 제한이 걸리고 아웃바운드는 허용되지 않는다. DB 패스워드는 랜덤 생성된 19자의 영대소문자, 숫자, 특수문자 혼합의 문자열을 사용한다.

지문 DB와 직접 커넥션을 이루는 SourceAFIS JAVA 웹서버는 Version 1에서의 구성과 동일하다. 내부망에만 올라와있으며, 혹시 모를 SSRF 공격에 대비하여 내부망에서도 토큰 인증을 이용한다.

또한 한 유저는 같은 name을 가진 여러 미뉴티아 레코드 묶음을 다수 보유할 수 있으며, 이를 통하여 한 지문을 같은 이름으로 여러번 등록하여 인식률을 높일 수 있고 다른 손가락은 분리하여 등록할 수 있다. 또한 앱에서 해당 지문을 관리 가능하다. 또한 유저가 더이상 지문인식결제를 이용하고 싶지 않아 앱에서 삭제버튼을 클릭함으로써 요청하거나 이용자가 졸업하는 경우 해당 개인정보가 불필요해졌음으로 개인정보 보호법 제21조에 의거하여 즉시 삭제한다.

참조자료

[i] https://www.adh-tech.com.tw/uploads/1764979488_HF302GD%20Module%20datasheet.pdf

[ii] https://www.adh-tech.com.tw/uploads/1764979770_HF302GD_AS608_V2.3.4.x.zip

[iii] https://docs.nordicsemi.com/bundle/ps_nrf54L15/page/chapters/ref_circuitry.html

[iv] https://www.ti.com/lit/an/swra117d/swra117d.pdf