ON.O.ON
(ONly Our face OpeN)

Ver.1.0
2020.6.25.

한국외국어대학교
정보통신공학과
Xproject 5Team

팀장 :
강경문(201500012)

팀원 :
강두영(201600017)
이효섭(201602790)
이채영(201802919)

머리말

본 문서는 안드로이드, 장고, 라즈베리파이를 이용하여 서로간의 통신(레트로핏, 소켓통신)과 데이터베이스, open cv를 활용한 머신러닝을 통해, 사용자들에게 보다 안전하고 편리한 도어락 시스템을 제공한다. 사용자는 앱을 통해 사진을 찍으면, 미리 학습시켜놓은 사진 모델과 비교해 도어락을 해제할 수 있다.

목차

1 개요

‘ON.O.ON’ 은 얼굴인식을 통해 자신의 등록된 정보(회원 사진)와 비교 및 인증 할 수 있는 도어락 서비스를 제공하고자 한다. 또한 파이썬에서 제공하는 numpy와 cv2를 활용해 효율적으로 연산할 수 있게 하였다.
본 장에서는 얼굴인식 도어락 시스템인 ‘ON.O.ON’에 대한 목적, 참고자료, 용어 및 약어를 제시한다.

1.1 목적

해당 프로젝트가 실현하고자 하는 바는 기존의 카드키나 열쇠, 비밀번호를 이용한 도어락에서의 문제점을 해결하고, 누구나 쉽게 접할 수 있는 휴대폰 어플리케이션을 통해 접근성 및 확장성을 높이고, 보다 편리하고 효율성 있는 도어락을 만드는 것이다.

기존의 도어락에서 가장 불편함을 느꼈던 것은 바로 인증장치(카드키)의 필요성이었다.
얼굴인식을 활용하면 따로 휴대해야할 인증장치가 필요 없어 도난 및 분실의 우려를 없애고, 비밀번호 유출이나 간혹 가다가 비밀번호를 까먹는 일도 방지할 수 있다.
또한 비접촉식 특성을 통해 코로나가 터진 요즘 같은 시국에도 안전성을 높이고 위생문제에도 도움이 될 수 있다.

1.2 용어 및 약어

머신러닝	기계학습 자료, 데이터를 가지고 기계 스스로 학습하여 데이터를 축적시켜놓고, 유사한 작업 수행시 자동으로 빠르고 효율적으로 수행하도록 하기위한 기술
API	Application Programming Interface 운영체제와 응용프로그램 사이의 통신에 사용되는 언어
라즈베리파이	초소형 컴퓨터/ 사용자가 원하는 대로 기능, 용도 확장 및 변경 가능
서보모터	빈번하게 변화하는 위치나 속도의 명령치에 대해 신속하고 정확하게 추종할 수 있도록 설계된 모터

2. 제품 소개

2.1 제품 개요

'ON.O.ON'은 ONly Our face OpeN 의 약자로서, 이 프로젝트의 핵심 기술인 얼굴인식을 의미한다. 이 제품은 데이터베이스에 저장된 사용자만의 사진을 인식하여 도어락이 열림을 나타내었다. 등록되지 않은 사용자는 인식되지 않아 도어락을 열 수 없다.

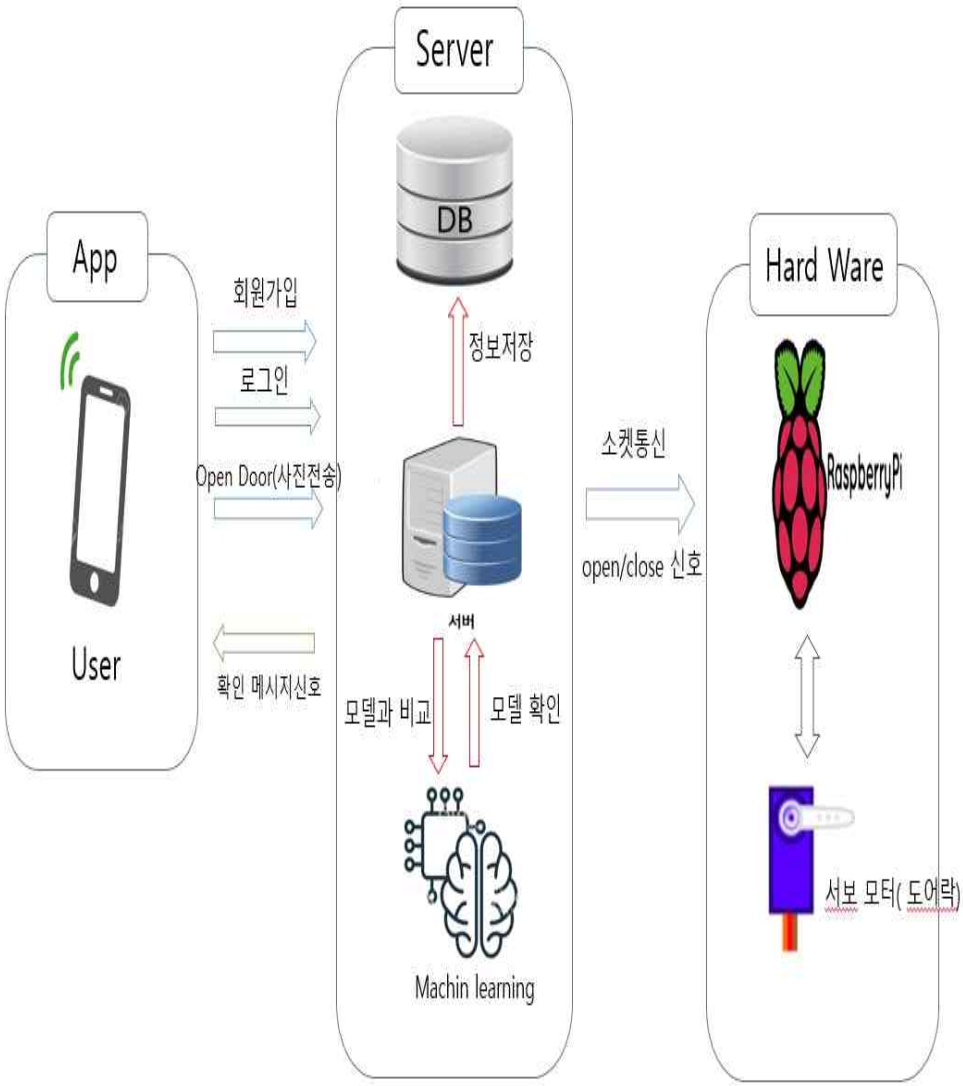
2.2 제품 필요성

위의 사진들은 카드키와 비밀번호를 이용한 도어락의 취약성에 대한 뉴스자료들이다.
카드키는 밖에 외출을 할 때면 항시 소지하고 다녀야만 한다. 그렇게 되면 분실 및 도난 위험이 커질 수 밖에 없다. 실제로도 카드키를 분실하는 것을 주변에서 쉽게 볼 수 있다. 카드키를 분실 또는 도난 당할 시, 언제든지 주거침입을 통한 2차적인 위험에 노출될 수 있다. 또한 카드키를 분실하게 되면, 새로운 카드키를 구입해야 하는 비용적인 문제점도 있다. 비밀번호 도어락 역시 비밀번호 유출 위험에 항상 노출되어 있다.

하지만 얼굴인식을 활용하면, 이러한 문제점들을 해결할 수 있다.
인증장치가 필요 없어 사용자 편의성을 높이고, 분실, 도난 위험을 최소화 시킬 수 있다.
휴대폰 어플리케이션을 이용하였기 때문에 따로 비용적인 측면을 걱정하지 않아도 된다.
이런 이유로 개인만의 고유한 특징을 가지고 있는 얼굴 특성을 통해 개개인을 확인하는 ON.O.ON의 도입이 필요하다고 할 수 있다.

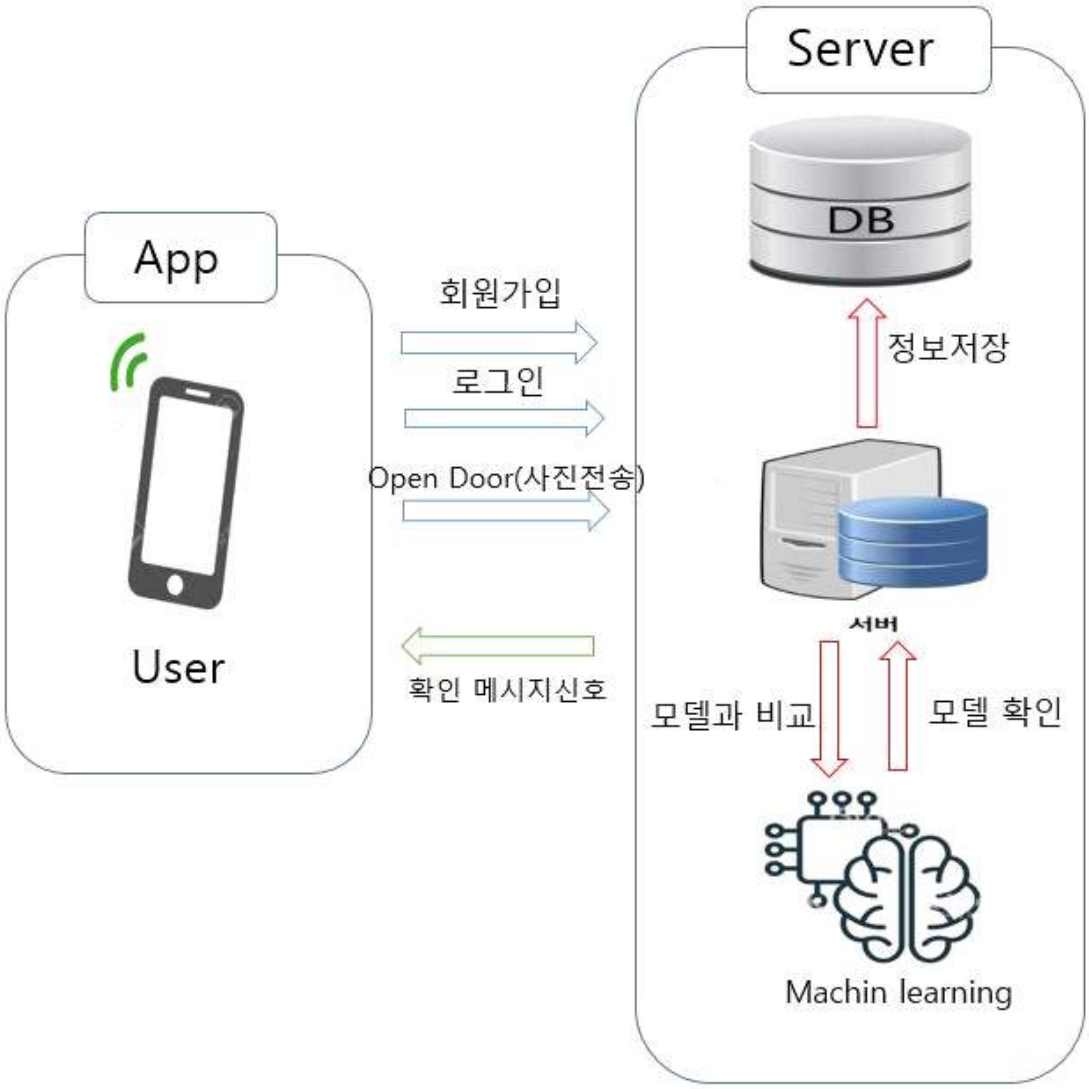
3. 시스템 구성 및 설계

3.1 전체 시스템 구성도

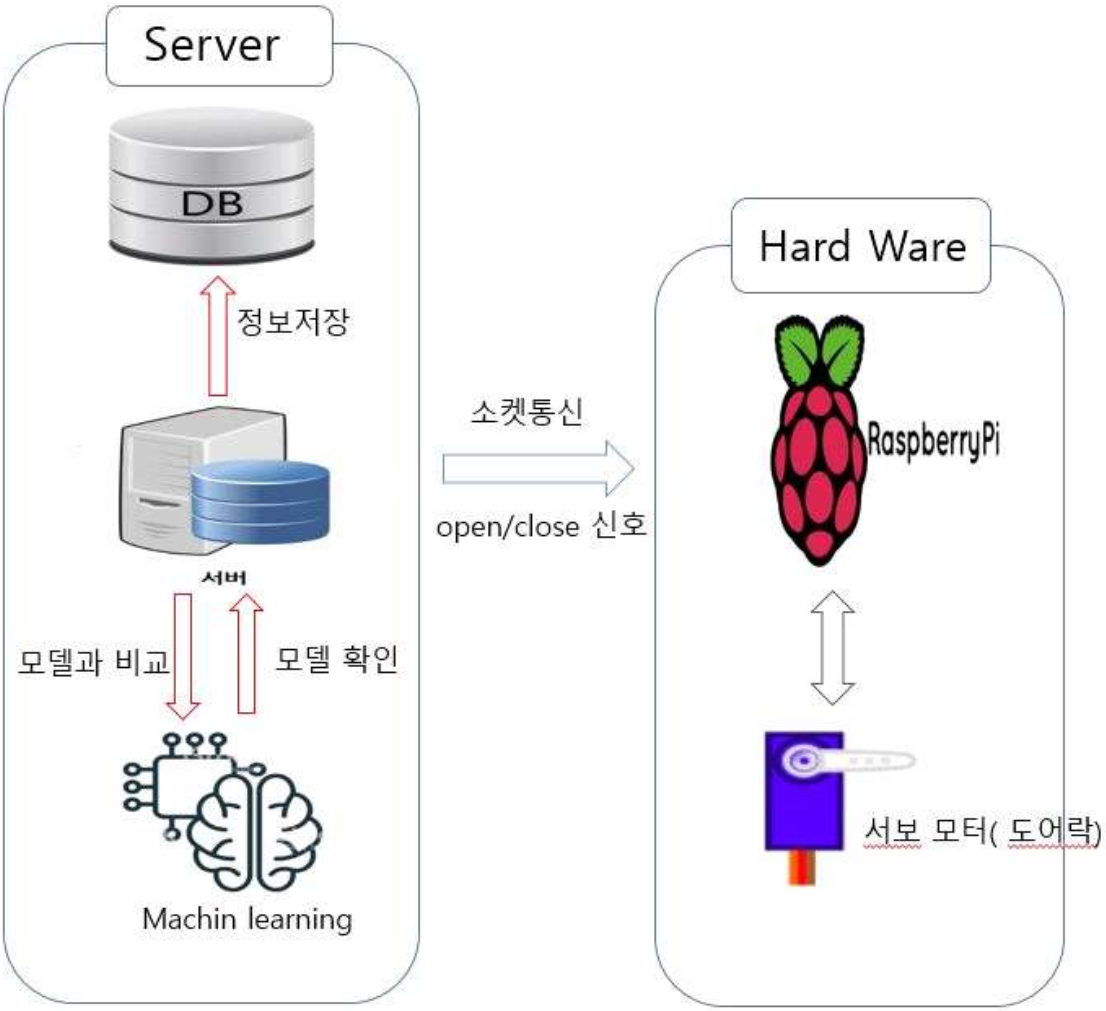


사용자는 앱을 사용해 회원가입을 한다. 회원가입시 정보는 서버의 DB에 저장되게된다. 그 후, 문을 열려고 하면 사용자는 앱을 켜고 회원가입 했던 정보로 로그인하여 Open Door 버튼을 누른다.
Open Door 버튼을 누르면 사진을 찍어 Django의 머신러닝서버로 보내고 학습시킨 모델과 사진을 비교하여 본인확인 절차를 밟는다.
본인확인이 완료되면, 서버는 레트로핏을 통해 앱으로는 문이 열렸다는 메시지를 보내고, 라즈베리파이로는 socket통신을 통해 문(서보모터)을 열라는 신호를 보낸다.

3.2 사용자/서버 구성도


Server
DB
App
회원가입
로그인
정보저장
Open Door(사진전송)
User
확인 메시지신호
모델과 비교
모델 확인
Machin learning

3.3 도어락/서버 구성도


Server
DB
정보저장
서버
모델과 비교
모델 확인
Machin learning
소켓통신
open/close 신호
Hard Ware
RaspberryPi
서보 모터(도어락)

3.4 설계

3.4.1 서버/클라이언트 연결

Django(서버) /android(클라이언트)

```
interface LoginService{
    @FormUrlEncoded
    @POST( value: "/app_login/")
    fun requestLogin(
        @Field( value: "userid") userid:String,
        @Field( value: "userpw") userpw:String
    ) : Call<Login>
}
```

```
interface AddfaceService{
    @Multipart
    @POST( value: "/app_addface/")
    fun requestAddface(
        @Part imageFile : MultipartBody.Part
    ): Call<Addface>
}
```

request 데이터를 하나씩 지정해서 보내기 위하여 @Part를 사용하고 이미지 파일을 전송하기 위해 MutipartBody.Part를 사용하여 HTTP 요청을 수행하는 Call 메소드가 있는 API 인터페이스를 생성한다.

```
var retrofit : Retrofit  = Retrofit.Builder()
        .baseUrl( baseUrl: "http://192.168.0.211:8000")
        .addConverterFactory(GsonConverterFactory.create())
        .build()
```

String이나 int 값들을 전송하는 경우에는 Retrofit Builder 클래스를 생성한 후

```
val intent = Intent( packageContext: this, OpendoorActivity::class.java)
loginService.requestLogin(textId, textPw).enqueue(object : Callback<Login> {
```

Call 메소드 객체를 선언하지만

```
var requestBody : RequestBody = RequestBody.create(MediaType.parse( string: "image/*"),file)
var textId : String  = signup_id.text.toString()
var body : MultipartBody.Part = MultipartBody.Part.createFormData( name: "uploaded_file",  filename: textId+"_1.jpeg", requestBody)
```

이미지 파일을 전송하는 경우에는 ResponseBody에 넣고 MutipartBody.Part로 한번 더 감싸서 Call 메소드 객체를 선언하고

```
override fun onFailure(call: Call<Addface>, t: Throwable) {
    Log.d( tag: "통신 실패", t.message) // 통신에 실패 했을 시
}
override fun onResponse(call: Call<Addface>, response: Response<Addface>) {
    if (response?.isSuccessful){ // 통신에 성공하고 응답이 있을 경우
        Log.d( tag: "통신 성공", msg: ""+response?.body().toString())
        var login : Addface?  = response.body()
        Log.d( tag: "LOGIN", msg: "msg : "+login?.msg)
        Log.d( tag: "LOGIN", msg: "code : "+login?.code)
    }else{ //통신에 성공하였지만 응답이 오지 않았을 경우
        Toast.makeText(getApplicationContext(),  text: "Some error occured...", Toast.LENGTH_LONG).show();
    }
```

비동기로 실행한다. 이후 서버에서 응답을 받아와 원하는 작업을 수행한다.

3.4.2 회원가입

```
register_addface.setOnClickListener { it: View!
    var textId : String  = register_id.text.toString()
    var textPw : String  = register_pass.text.toString()
    var textName : String  = register_name.text.toString()
    val intent = Intent( packageContext: this, AddFaceActivity::class.java)
    signupservice.requestSignup(textId, textPw, textName).enqueue(object : Callback<Signup> {
```

어플에서 ID, PASSOWORD, NAME를 입력 한 후 START REGISTER FACE 버튼을 누르면 Django 서버로 이 정보들을 보내주게 된다.

```
@csrf_exempt
def app_signup(request):
    if request.method == 'POST':
        print("리퀘스트 로그" + str(request.body))
        id = request.POST.get('userid', '')
        pw = request.POST.get('userpw', '')
        username = request.POST.get('username', '')
        print("id = " + id + " pw = " + pw + " name = " + username)
        myuser = Addresses.objects.filter(userid=id)
        if myuser:
            print("중복!")
        else:
            # db 저장
            form = Addresses(userid=id, userpw=pw,name=username)
            form.save()
            print("성공")
            return JsonResponse({'code': '1001', 'msg': '회원가입성공입니다.'}, status=200)
```

서버에서는 request가 오면 정보를 확인 한 후 ID 정보가 데이터 베이스에 저장된 정보 중에 똑같은 것이 있는지 확인한다. 중복하는 정보가 없다면 서버는 회원가입 정보를 데이터베이스에 저장한 후 클라이언트에게 회원가입 성공이라는 결과 메시지를 보내준다.

```
signupservice.requestSignup(textId, textPw, textName).enqueue(object : Callback<Signup> {
    override fun onFailure(call: Call<Signup>, t: Throwable) {
        Log.e( tag: "DEBUG1", t.message)
        var dialog = AlertDialog.Builder( context: this@SignupActivity)
        Toast.makeText( context: this@SignupActivity,  text: "통신에 실패했습니다.", Toast.LENGTH_SHORT).show()
    }
    override fun onResponse(call: Call<Signup>, response: Response<Signup>) {
        if (response?.isSuccessful) {
            var login : Signup?  = response.body()
            Log.d( tag: "LOGIN",  msg: "msg : " + login?.msg)
            Log.d( tag: "LOGIN",  msg: "code : " + login?.code)
            var dialog = AlertDialog.Builder( context: this@SignupActivity)
            Toast.makeText( context: this@SignupActivity,  text: "사진을 찍으세요.", Toast.LENGTH_SHORT).show()
            intent.putExtra( name: "userid", textId)
            // id가 중복이 아니여서 회원가입이 되면 사진을 찍는 창으로 이동하는 경우
            startActivity(intent)
            finish()

        }else {
            // id가 중복되어 회원가입이 실패되고 창 이동이 없는 경우
            Toast.makeText(getApplicationContext(),  text: "아이디가 중복되었습니다.", Toast.LENGTH_LONG).show();
```

정보가 중복된다면 아무런 메시지를 보내주지 않아 결과 메시지를 받지 못한 어플은 아이디가 중복되었다는 것으로 인식해 사용자에게 아이디가 중복되었다는 것을 알려준다.

3.4.3 로그인

```
button.setOnClickListener { it: View!
    var textId : String  = editTextTextPersonName.text.toString()
    var textPw : String  = editTextTextPassword.text.toString()
    val intent = Intent( packageContext: this, OpendoorActivity::class.java)

    loginService.requestLogin(textId, textPw).enqueue(object : Callback<Login> {
```

어플에서 ID, PASSOWORD 를 입력 한 후 LOGIN 버튼을 눌렸을 때 Django 서버로 이 정보들을 보내주게 된다.

```
@csrf_exempt
def app_login(request):
    if request.method == 'POST':
        print("리퀘스트 로그" + str(request.body))
        id = request.POST.get('userid', '')
        pw = request.POST.get('userpw', '')
        print("id = " + id + " pw = " + pw)
        ##디비에서 id,pw 가져와서 -> 확인=result
        myuser = Addresses.objects.filter(userid=id, userpw=pw)
        print(myuser)
        if myuser:
            print("로그인 성공!")
            return JsonResponse({'code': '0000', 'msg': '로그인성공입니다.'}, status=200)
        else :
            print("로그인 실패")
```

서버에서는 request가 받은 정보에서 ID와 PASSWORD 모두가 데이터베이스에 저장된 것과 일치하는 것이 있는 지 확인한다.

```
loginService.requestLogin(textId, textPw).enqueue(object : Callback<Login> {
        override fun onFailure(call: Call<Login>, t: Throwable) {
            Log.e( tag: "DEBUG", t.message)
            var dialog = AlertDialog.Builder( context: this@MainActivity)
            dialog.setTitle("실패")
            dialog.setMessage("통신에 실패했습니다..")
            dialog.show()
        }
        override fun onResponse(call: Call<Login>, response: Response<Login>) {
            if (response?.isSuccessful) {
                var login : Login?  = response.body()
                Log.d( tag: "LOGIN",  msg: "msg : " + login?.msg)
                Log.d( tag: "LOGIN",  msg: "code : " + login?.code)
                var dialog = AlertDialog.Builder( context: this@MainActivity)
                Toast.makeText( context: this@MainActivity,  text: "로그인이 되었습니다.", Toast.LENGTH_SHORT)
                    .show()
                // 로그인 시 다음 창으로 이동
                intent.putExtra( name: "userid", textId)
                //intent.putExtra("userpassword", textPw)
                startActivity(intent)
                finish()
            }else{
                // id또는 password가 틀렸을 시 로그인이 실패되기 때문에 창 이동이 없는 경우
                Toast.makeText( context: this@MainActivity,  text: "로그인이 실패하셨습니다.", Toast.LENGTH_SHORT).show()
```

일치하는 정보가 있으면 서버는 클라이언트에게 회원가입 성공이라는 결과 메시지를 보내주고 만약 일치하는 정보가 없다면 아무런 메시지를 보내주지 않는다. 결과 메시지를 받지 못한 어플은 로그인이 실패하였다는 신호로

인식하고 로그인 실패 여부를 알려준다.

3.4.4 얼굴등록

```
//테드퍼미션 설정
private fun setPermission() {
    val permission = object : PermissionListener {
        override fun onPermissionGranted() {//허용되었을경우 이거 수행
            Toast.makeText( context: this@AddFaceActivity,  text: "Access success.", Toast.LENGTH_SHORT).show()
        }

        override fun onPermissionDenied(deniedPermissions: MutableList<String>?) {
            Toast.makeText( context: this@AddFaceActivity,  text: "Access denied.", Toast.LENGTH_SHORT).show()
        }
    }
    TedPermission.with( context: this)
        .setPermissionListener(permission)
        .setRationaleMessage("카메라 앱을 사용하시려면 권한을 허용해주세요.")
        .setDeniedMessage("권한을 거부하였습니다.")
        .setPermissions(android.Manifest.permission.WRITE_EXTERNAL_STORAGE, android.Manifest.permission.CAMERA)
        .check()
```

얼굴등록 화면에 들어오자마자 카메라 사용 권한을 설정하고 권한 허용 여부에 대한 메시지가 뜬다. 이때 권한 거부 시 사진을 찍을 수 없다.

```
private fun takeCapture() {//카메라 실행
    //기본카메라 앱 실행
    Intent(MediaStore.ACTION_IMAGE_CAPTURE).also{takePictureIntent : Intent  ->
        takePictureIntent.resolveActivity(packageManager)?.also { it: ComponentName
            val photoFile: File? = try{
                createImageFile() //이미지파일 생성하는 함수
            }catch (ex: IOException){
                null
            }
            photoFile?.also { it: File
                val photoURI: Uri = FileProvider.getUriForFile(
                    context: this,
                    authority: "com.example.xproject.fileprovider",
                    it
                )
                takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI)
                startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE)
```

권한 허용 후 FIRST PHOTO(혹은 다른 화면일 경우 SECOND PHOTO, THIRD PHOTO) 버튼을 누르면 기본 카메라 어플이 실행되고 사진을 찍으면

```
private fun createImageFile(): File? {//이미지파일 생성
    val timestamp: String = SimpleDateFormat( pattern: "yyyyMMdd_HHmmss").format(Date())
    val storageDir: File? = getExternalFilesDir(Environment.DIRECTORY_PICTURES)
    return File.createTempFile( prefix: "JPEG_${timestamp}_", suffix: ".jpg",storageDir)
        .apply{curPhotoPath = absolutePath}
```

현재 시각을 이름으로 하는 JPEH 파일이 생성된다. 동시에 사진을 찍었다는 신호가 오면 onActicityResult가 불려

```
private fun savePhoto(file: File, bitmap: Bitmap) {//갤러리에 저장
    val folderPath : String  = Environment.getExternalStorageDirectory().absolutePath + "/Pictures/"
    val timestamp: String = SimpleDateFormat( pattern: "yyyyMMdd_HHmmss").format(Date())
    val fileName = "${timestamp}.jpeg"
    val folder = File(folderPath)
    take_picture =1
    if(!folder.isDirectory) {//현재 해당경로에 폴더가 존재하는지
        folder.mkdir()
    }
    val out = FileOutputStream( name: folderPath + fileName)
    bitmap.compress(Bitmap.CompressFormat.JPEG,  quality: 100, out)
    Toast.makeText( context: this,  text: "사진이 앨범에 저장되었습니다", Toast.LENGTH_SHORT).show()
    sendPhoto(fileName, file)
```

기본 위치 밑 Picture 디렉토리에 사진을 저장을 한 후 file과 fileName을 인자로 받는 sendPhoto 함수를 불러준다.

```
private fun sendPhoto(fileName: String, file: File) {
    var requestBody : RequestBody = RequestBody.create(MediaType.parse( string: "image/*"),file)
    var textId : String  = signup_id.text.toString()
    var body : MultipartBody.Part = MultipartBody.Part.createFormData( name: "uploaded_file",  filename: textId+"_1.jpeg", requestBody)
    //var body : MultipartBody.Part = MultipartBody.Part.createFormData("uploaded_file", fileName+"_1.jpeg", requestBody)


    var gson : Gson = GsonBuilder()
        .setLenient()
        .create()

    var retrofit : Retrofit  = Retrofit.Builder()
        .baseUrl( baseUrl: "http://192.168.0.211:8000")
        .addConverterFactory(GsonConverterFactory.create(gson))
        .build()
```

server로 이미지 파일을 회원가입 때 입력한 id를 파일 이름으로 보내기 위해 MutipartBody.part클래스를 이용하여 FormData를 만들어 ResponseBody를 URL로 보내주게 된다.

```
@csrf_exempt
def app_app_addface(request):
    if request.method == 'POST':
        photo = request.FILES.get('uploaded_file')
        photo_name = photo.name[:-7]
        if('_1' in photo.name):
            # 첫번째 사진 등록시
            form = Pictures(userid=photo_name, first_picture = photo)
            form.save()
        elif('_2' in photo.name):
            # 두번째 사진 등록시
            user = Pictures.objects.get(userid=photo_name)
            user.second_picture = photo
            user.save()
        elif('_3' in photo.name):
            # 세번째 사진 등록시
            user = Pictures.objects.get(userid=photo_name)
            user.third_picture = photo
            user.save()
        return JsonResponse({'code': '0000', 'msg': '사진받았습니다.'}, status=200)
```

얼굴 등록 완료 후 로그인을 하고 문을 열 때 2차 인증 위해 사진을 사용하기 위해 데이터베이스에 회원가입을 진행하고 있는 사용자의 id와 사진들을 저장한다. 성공적으로 저장이 되면 어플로 저장이 되었다는 신호를 보내준다. 만약 사진을 찍지 않고 다음 단계로 넘어가는 것을 시도한다면 사진을 찍어야 한다는 알림을 띄워 사용자의 얼굴이 무조건적으로 등록할 수 있도록 진행한다.

3.4.5 서비스 실행

사용자가 회원가입과 로그인을 성공적으로 마친 후 OPEN DOOR라는 버튼을 누르면 얼굴인식을 진행하기 위해 얼굴 등록 과정과 동일하게 현재 어플 사용자의 얼굴을 찍어 서버로 전송한다.

```
@csrf_exempt
def app_app_opendoor(request):
    if request.method == 'POST':
        photo = request.FILES.get('uploaded_file')
        photo_name = photo.name[:-5]
        form = LoginPicture(userid=photo_name, login_picture = photo)
        form.save() # 현재 로그인을 한 사용자의 id와 문을 열기를 시도한 사람의 사진 저장
         # 문을 여는 것을 시도 한 사람의 사진을 확인
        lock_state, unlock_name = face_recog_3.run(face_recog_3.trains(), photo.file)
        print("1차 인증결과 : "+(lock_state, unlock_name))
        user = Pictures.objects.get(userid=photo_name)
        # 현재 로그인한 사용자의 등록된 사진을 확인
        user_lock_state, user_unlock_name = face_recog_3.run(face_recog_3.trains(), user.first_picture)
        print("2차 인증결과 : "+(user_lock_state, user_unlock_name))
        # 로그인을 한 사람과 문을 여는 것을 시도한 사람이 같을 경우
        if(lock_state == "unlock" and user_lock_state == "unlock" and unlock_name == user_unlock_name):
            sys.path.insert(1, 'C:/Users/w1004/Desktop/xproject/xproject/')
            import server_close
            server_close.opendoor()
            return JsonResponse({'code': '0000', 'msg': '사진받았습니다.'}, status=200)
```

1차로 현재 문을 여는 것을 시도한 사용자의 얼굴을 인식을 하고 2차로 현재 로그인 한 사람의 얼굴을 인식하여 1차가 2차와 동일 인물로 나왔을 경우 라즈베리 파이로 'open'이라는 데이터를 보내어 문을 열고 어플에게는 결과 메시지를 보내어 문이 열렸다는 것을 알려준다.
만약 얼굴인식에 실패하거나 1차와 2차가 다른 사용자가 나온다면 문이 열리지 않고 어플에게도 얼굴인식에 실패하였다는 신호를 보내준다.

3.4.5 (1)
사용자의 사진과 머신러닝

```python
import cv2
import numpy as np
from os import makedirs
from os.path import isdir

# 얼굴 저장
face_dirs = 'faces/'
face_classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')


def face_extractor(img):            # 얼굴 검출
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray,1.3,5)
    if faces is():
        return None
    for(x,y,w,h) in faces:
        cropped_face = img[y:y+h, x:x+w]
    return cropped_face

def take_pictures(name):       # 얼굴 저장하는 장소
    if not isdir(face_dirs+name):
        makedirs(face_dirs+name)

    cap = cv2.VideoCapture(0)
```

```python
def take_pictures(name):       # 얼굴 저장하는 장소
    if not isdir(face_dirs+name):
        makedirs(face_dirs+name)

    cap = cv2.VideoCapture(0)
    count = 0

 # 사진찍기
    while True:
        ret, frame = cap.read()
        if face_extractor(frame) is not None:
            count+=1
            face = cv2.resize(face_extractor(frame),(200,200))
            face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

            file_name_path = face_dirs + name + '/user'+str(count)+'.jpg'
            cv2.imwrite(file_name_path,face)

            cv2.putText(face,str(count),(50,50),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
            cv2.imshow('Face Cropper',face)
        else:
            print("얼굴을 찾을 수 없습니다.")
            pass
        if cv2.waitKey(1)==13 or count==300:
            break

    cap.release()
    cv2.destroyAllWindows()
    print('Colleting 완료!!!')
```

얼굴 검증을 위한 데이터 베이스(사진) 수집용 머신러닝 코드이다.
위의 코드는 사진을 머신러닝 데이터베이스에 찍어서 저장해 놓는 역할을 수행한다.
본 팀에서는 사전에 머신러닝 데이터 베이스에 사진을 저장해 놓고 실행하였다.

머신러닝 데이터 베이스에 저장된 사진을 바탕으로 사용자의 모델을 학습시키는 코드이다. 여러 장의 사진을 통해 사용자의 얼굴 정보를 추출하여 모델을 학습하게 된다. 결과로는 사용자의 이름+모델 학습완료가 결과로 나오게 된다.

```python
import cv2
import numpy as np
from os import listdir
from os.path import isdir, isfile, join

path = 'C:/Users/w1004/Desktop/xproject/xproject/ml/'
face_classifier = cv2.CascadeClassifier(path+'haarcascade_frontalface_default.xml')
# 사용자 얼굴 학습
def train(name):
    data_path = path +'faces/'+ name + '/'
    face_pics = [f for f in listdir(data_path) if isfile(join(data_path,f))]
    Training_Data, Labels = [], []

    for i, files in enumerate(face_pics):
        image_path = data_path + face_pics[i]
        images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
        if images is None:
            continue
        Training_Data.append(np.asarray(images, dtype=np.uint8))
        Labels.append(i)
    if len(Labels) == 0:
        #print("인식 불가.")
        return None
    Labels = np.asarray(Labels, dtype=np.int32)
     # 모델 생성
    model = cv2.face.LBPHFaceRecognizer_create()
    model.train(np.asarray(Training_Data), np.asarray(Labels))
    print(name + " : 모델 학습 완료 !!")

#여러 사용자 학습
def trains():
    data_path =  path +'faces/'
    model_dirs = [f for f in listdir(data_path) if isdir(join(data_path,f))]
    models = {}
    for model in model_dirs:
        print('model :' + model)
        result = train(model)
        if result is None:
            continue

        print('model2 :' + model)
        models[model] = result
    return models           # 학습된 모델들 리턴

#얼굴 검출
def face_detector(img, size = 0.5):
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_classifier.detectMultiScale(gray,1.3,5)
        if faces is():
            return img,[]
        for(x,y,w,h) in faces:
            cv2.rectangle(img, (x,y),(x+w,y+h),(0,255,255),2)
            roi = img[y:y+h, x:x+w]
            roi = cv2.resize(roi, (200,200))
        return img,roi
```

다수의 사용자들이 있기에 OPEN DOOR버튼을 누르게 되면
path = 'C:/Users/w1004/Desktop/xproject/xproject/ml/' + 'faces/'에
저장되어 있는 여러 사용자들의 사진을 학습하여 학습된 모델들을 리턴시킨다.

```python
#얼굴 검출
def face_detector(img, size = 0.5):
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_classifier.detectMultiScale(gray,1.3,5)
        if faces is():
            return img,[]
        for(x,y,w,h) in faces:
            cv2.rectangle(img, (x,y),(x+w,y+h),(0,255,255),2)
            roi = img[y:y+h, x:x+w]
            roi = cv2.resize(roi, (200,200))
        return img,roi
# 인식 테스트 [img = 받아오는 부분]
def run(models, img):

    from PIL import Image
    im = Image.open(img)
    im2 = im.rotate(90)
    im2.save(path+"sample_bmp.jpg")

    frame = cv2.imread(path+"sample_bmp.jpg")
    #frame = cv2.imread(path+'faces/j.jpg')

    image, face = face_detector(frame)

    try:
        min_score = 999
        min_score_name = ""
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

        for key, model in models.items():
            result = model.predict(face)
            if min_score > result[1]:
                min_score = result[1]
                min_score_name = key

        if min_score < 500:
            confidence = int(100*(1-(min_score)/300))


        if confidence > 68 :
            print("Unlocked : " + min_score_name )
            #print(confidence)
            return ("unlock", min_score_name)

        else:
            print("Locked"+"  : not admin user"+ min_score_name )

            print(confidence)
            return ("lock", min_score_name)

    except:
        #return "lock"
        #print("face not found")
        print("Locked")
        #pass

    cv2.destroyAllWindows()

if __name__ == "__main__":
    #models = trains()
    run(trains(),path+'faces/j.jpg')
```

face detector 함수는 칼라 사진을 흑백으로 변환하고, face_classifier = cv2.CascadeClassifier(path+'haarcascade_frontalface_default.xml') (얼굴 인식 데이터를 축적한 xml 파일)을 통해 사진에 얼굴이 없으면 아무것도 리턴하지 않고 사진에 얼굴이 있다면 img와 roi값을 리턴한다.

run 함수는 실제로 사용자의 얼굴을 받은 뒤에 머신러닝처리를 거친 후 Open door 버튼을 눌렀을 때 사용자의 사진이 모델에 있는지를 판단 한 뒤에 사용자가 맞을 경우

return ("unlock", min_score_name)
return ("lock", min_score_name)을 리턴 받습니다.

```
131  @csrf_exempt
132  def app_app_opendoor(request):
133
134      if request.method == 'POST':
135          photo = request.FILES.get('uploaded_file')
136          photo_name = photo.name[:-5]
137          #print("photo = "+photo_name)
138          form = LoginPicture(userid=photo_name, login_picture = photo)
139          form.save()
140          #print(photo_name)
141          lock_state, unlock_name = face_recog_3.run(face_recog_3.trains(), photo.file)
142          print((lock_state, unlock_name))
143
144          #추가
145          user = Pictures.objects.get(userid=photo_name)
146          user_lock_state, user_unlock_name = face_recog_3.run(face_recog_3.trains(), user.first_picture)
147          print((user_lock_state, user_unlock_name))
148
149          if(lock_state == "unlock" and user_lock_state == "unlock" and unlock_name == user_unlock_name):
150              sys.path.insert(1, 'C:/Users/w1004/Desktop/xproject/xproject/')
151              import server_close
152              server_close.opendoor()
153              return JsonResponse({'code': '0000', 'msg': '사진받았습니다.'}, status=200)
154
```

실제로 서비스를 실행하는 장고 서버의 view - app_app_opendoor이다.
신뢰성을 올리기 위하여

1차로 OPEN DOOR버튼을 눌렀을 때의 사용자 사진과 머신러닝 데이터 베이스에 저장된 사진과의 비교,

2차로는 사용자가 SIGN UP 할 때의 사용자 사진과 머신러닝 데이터 베이스에 저장된 사진과의 머신러닝 비교과정을 통해 문이 열릴 수 있도록 디자인 하였다.

3.4.5 (2)

Raspberrypi(서버)/Django(클라이언트) 소켓통신

```
145         user = Pictures.objects.get(userid=photo_name)
146         user_lock_state, user_unlock_name = face_recog_3.run(face_recog_3.trains(), user.first_picture)
147         print((user_lock_state, user_unlock_name))
148 
149         if(lock_state == "unlock" and user_lock_state == "unlock" and unlock_name == user_unlock_name):
150             sys.path.insert(1, 'C:/Users/w1004/Desktop/xproject/xproject/')
151             import server_close
152             server_close.opendoor()
153             return JsonResponse({'code': '0000', 'msg': '사진받았습니다.'}, status=200)
154 
155     def login_page(request):
156         return render(request, "addresses/login.html")
157 
```

views.py에서 사용자의 사진이 일치하게 되면 라즈베리파이로 소켓통신을 통해 문을 열어주라는 신호를 보내게 된다.

이때 장고는 client , 라즈베리파이는 server의 역할을 수행하게 된다.

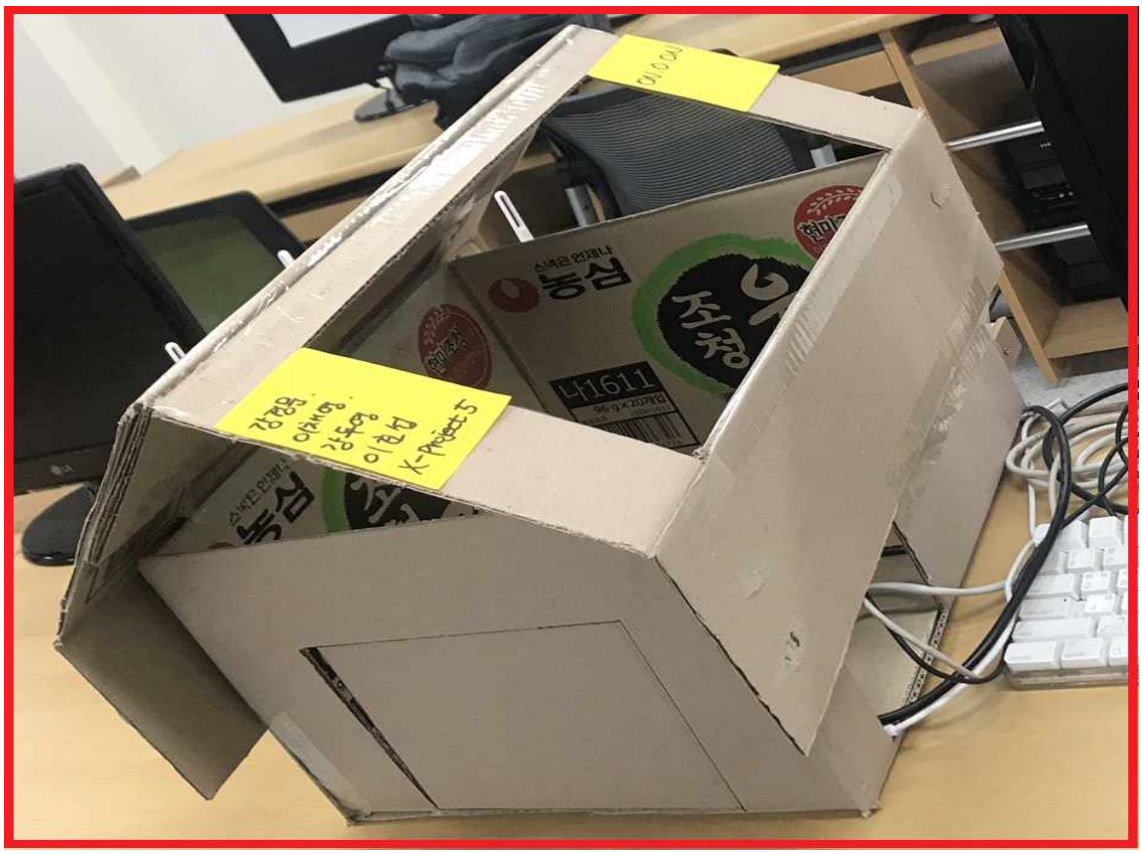
```
1   import socket
2   import argparse
3   port = 65000
4   
5   def opendoor():
6       client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7       # 지정한 host와 prot를 통해 서버에 접속합니다.
8       client_socket.connect(("192.168.0.215", port))
9       sendData = "open"
10      client_socket.sendall(sendData.encode('utf-8'))
11  
12      # 소켓을 닫는다.
13      client_socket.close()
14  
```

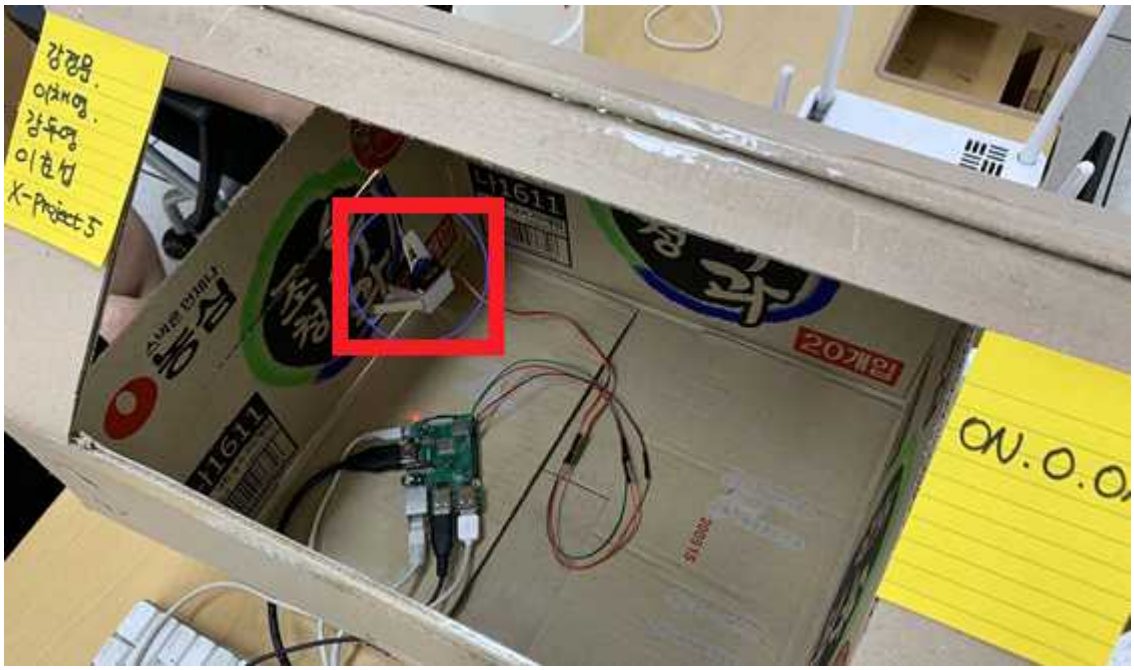
```
port = 65000


GPIO.setmode(GPIO.BCM)
sw = 27
s = 18
GPIO.setup(sw, GPIO.IN, GPIO.PUD_UP)
GPIO.setup(s, GPIO.OUT)
global servo
servo = GPIO.PWM(s,100)
servo.start(0.5)
servo.ChangeDutyCycle(11)

ctrCmd = ['open', 'close']

def closeservo():
        servo.ChangeDutyCycle(11)
        time.sleep(1)
        print("0")

def openservo():
        servo.ChangeDutyCycle(21)
        time.sleep(1)
        print("90")

def handle_client(client_socket, addr):
    recvData = client_socket.recv(1024)
    print(recvData.decode('utf-8'))
    if recvData == ctrCmd[0]:
        servo.ChangeDutyCycle(20)
        openservo()
        start_time = time.time()
        while(True):
                now_time = time.time()-start_time
                if(now_time >5):
                        break
        #time.sleep(5)
        closeservo()

    time.sleep(2)
    client_socket.close()
```

```
56
57  def handle_client(client_socket, addr):
58      recvData = client_socket.recv(1024)
59      print(recvData.decode('utf-8'))
60      if recvData == ctrCmd[0]:
61          servo.ChangeDutyCycle(20)
62          openservo()
63          start_time = time.time()
64          while(True):
65                  now_time = time.time()-start_time
66                  if(now_time >5):
67                          break
68          #time.sleep(5)
69          closeservo()
70
71      time.sleep(2)
72      client_socket.close()
73
74  def accept_func():
75      global server_socket
76      server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
77      server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
78      server_socket.bind(('', port))
79      server_socket.listen(10)
80      while 1:
81          try:
82              client_socket, addr = server_socket.accept()
83          except KeyboardInterrupt:
84              server_socket.close()
85              print("Keyboard interrupt")
86          t = threading.Thread(target=handle_client, args=(client_socket, addr))
87          t.daemon = True
88          t.start()
89
90  if __name__ == '__main__':
91      accept_func()
```

장고 클라이언트에서 'open' 신호가 소켓 통신을 통해 오게 되면
if recvData == ctrCmd[0]
구현해 놓은 서보모터 함수에 의해서 openservo()
문이 열리고 5초 뒤에는 자동으로 잠길 수 있도록 설계했다.

accept_func을 통해서는
t.daemon = True
클라이언트가 닫혀도 서버가 계속 열리게 하는 기능을 지원하고,
t = threading.Thread(target=handle_client, args=(client_socket, addr))
쓰레딩을 이용해 여러 사용자가 접속 할 수 있도록 하는 기능도 지원한다.

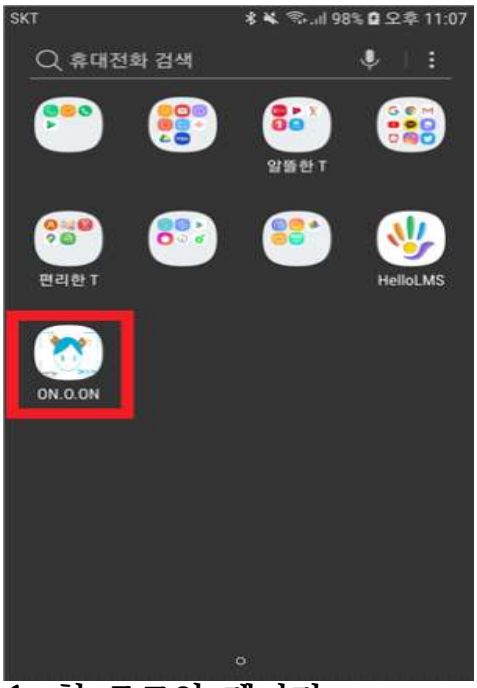


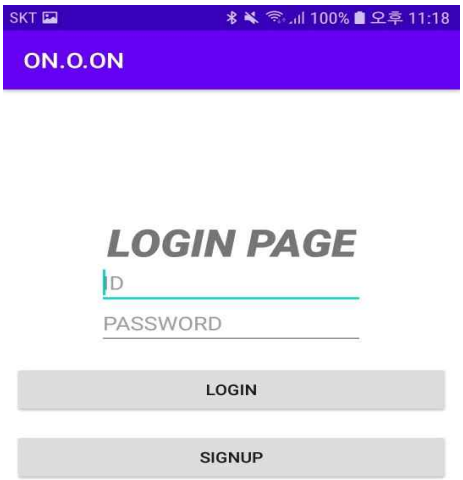


실제 구현이 되는 서보모터 도어락 형태

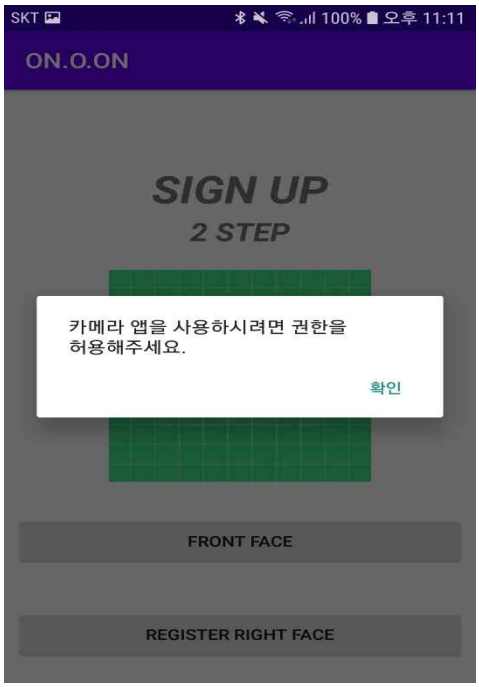
4. 시스템 시나리오

4.1 어플 실행 순서

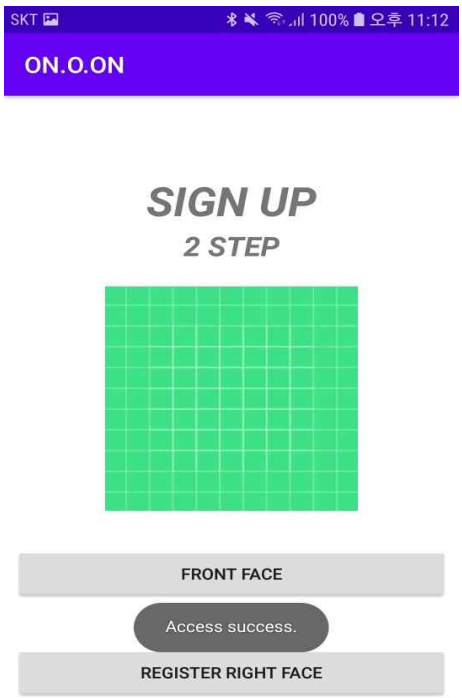


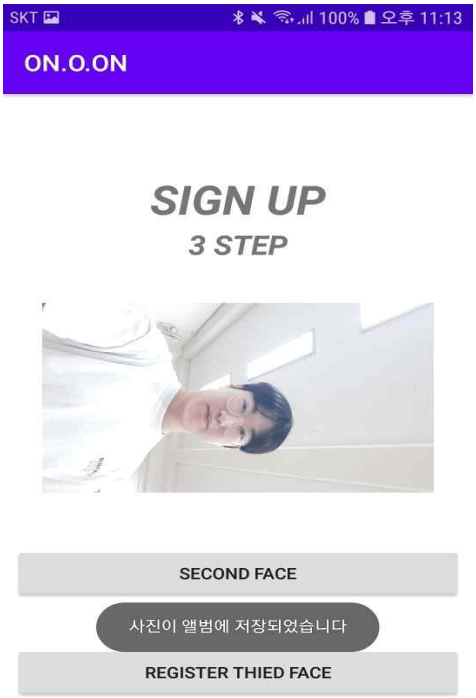


1. 첫 로그인 페이지

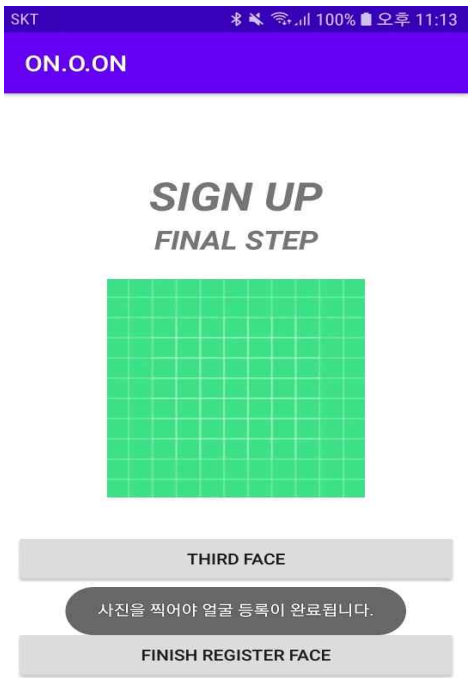


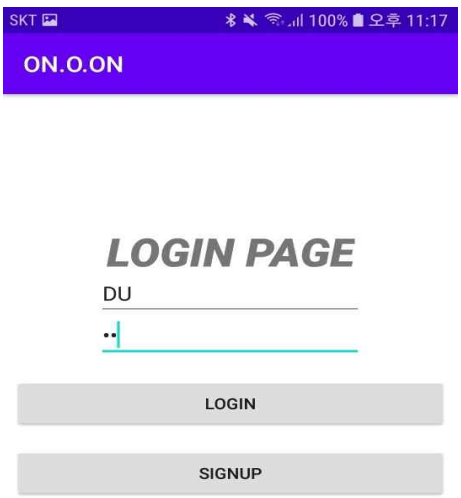
2. SIGN UP 버튼을 누르게 되면 카메라를 찍기 위해 권한을 허용





3. 사용자의 사진을 찍어서 장고 데이터 베이스에 저장합니다.
   총 3번의 사진을 찍는다.



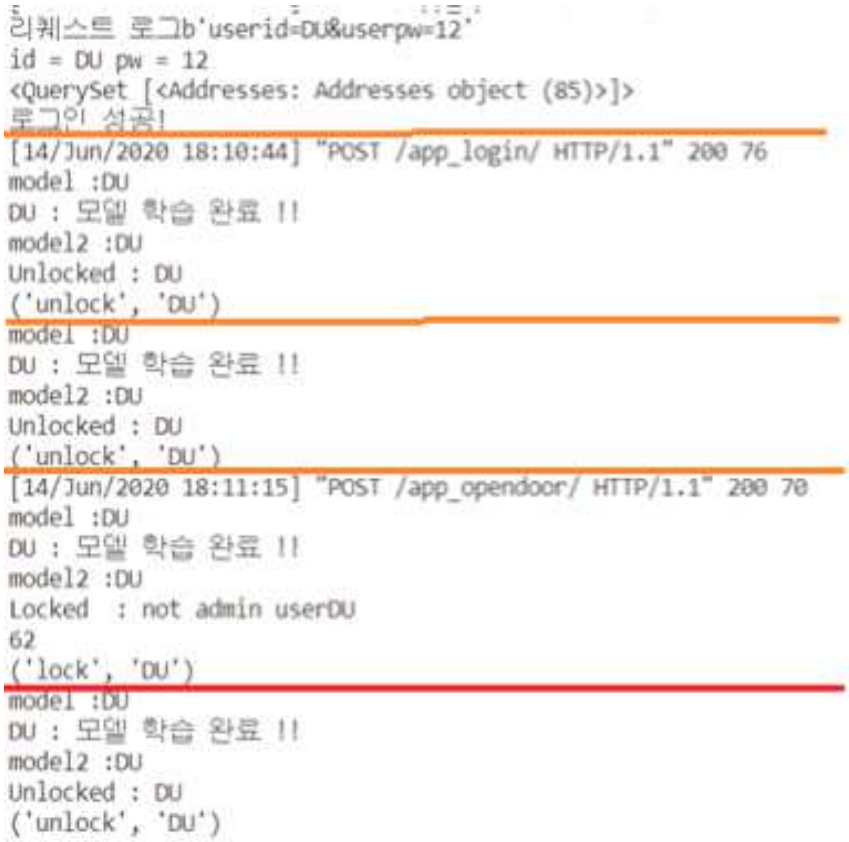


4. 마지막으로 사진을 찍고 로그인

5. 로그인이 되면 OPEN DOOR와 EXIT화면 발생

6. OPEN DOOR버튼을 눌러서 등록된 사용자일 경우 “문이 열렸습니다” 등록된 사용자가 아닐 경우 “얼굴 인식에 실패 하였습니다” 가 화면에 출력

4.2 Django 처리 현황



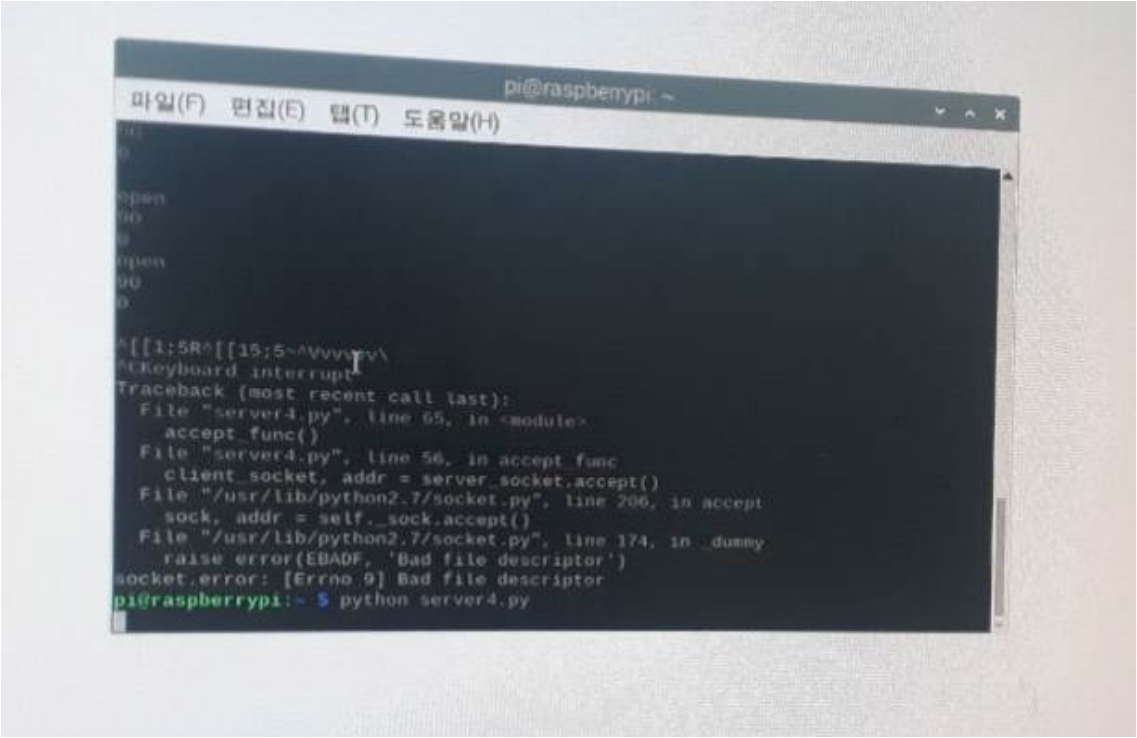
-첫 번째 주황색 라인: 로그인 성공

-두 번째 주황색 라인: 1 단계 얼굴 인증 성공 UNLOCK
(OPEN DOOR 눌렀을때의 사진과 DB 비교)

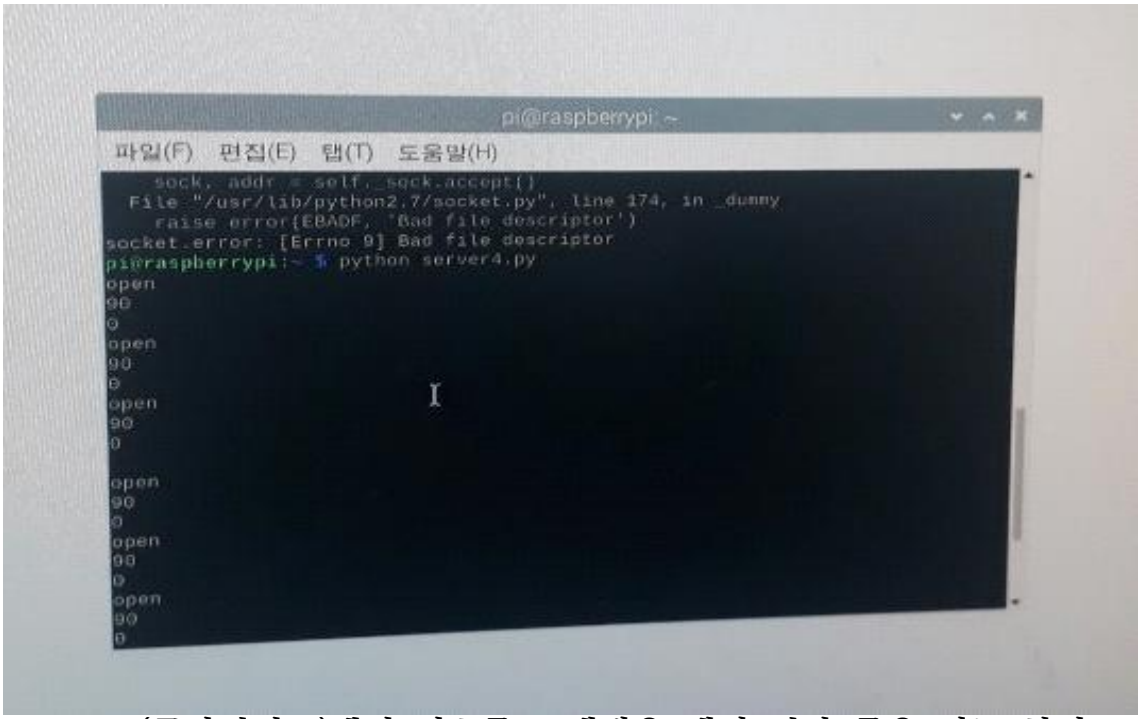
-세 번째 주황색 라인: 2 단계 얼굴 인증 성공 UNLOCK
(SIGN UP에 등록된 사진과 DB 비교)

-첫 번째 빨강색 라인: 등록되지 않은 사용자가 왔을 때 LOCK

4.3 Raspberrypi 처리 현황



-Raspberrypi(서버)가 열렸을 때의 화면



-Django(클라이언트)에서 신호를 보내줬을 때의 결과 문을 여는 화면

5. 적용방안 및 기대효과

● 인증장치(카드키,열쇠) 가지고 다닐 필요 없다. 비밀번호를 잊어버릴 일이 없다.

● 비접촉식 생체인식을 통해 위생 관련 안전성, 편리성

● 사진을 이용한 회원가입을 통해 개인만의 고유한 생체 인식으로 효율성을 높인다.

● 누구나 가지고 있는 핸드폰의 어플리케이션을 이용해 쉽게 접근 가능하다.

● 따로 카메라 설치 없이 휴대폰에 어플리케이션만 깔면 되기에, 비용 절감

6. S.W.O.T 분석

Strength

* 앱을 이용해 사용자 접근이 쉬움
* 앱을 이용한 비용절감
* 분실, 도난 걱정 없음
* 복제, 해킹 가능성 낮음

Weakness

* 비용, 시간적 제한으로 모터로 대체
* 조명, 각도 등에 따른 오인식률
* 하나의 도어락만 제어 가능

Opportunity

* 도어락 뿐 아니라 은행, 쇼핑, 의료 등 다양한 분야에서 얼굴인식을 이용해 본인인증, 결제 등에 접목시킬수 있다.

Threat

* 사진도용으로 인한 보안성 문제

6.1 SWOT 분석을 통한 경영 전략

 # SO(강점-기회) 전략

- 데이터베이스에 이용자 정보를 추가하여, 다양한 분야에 적용할 수 있게 하고, 얼굴인식을 활용하여 도어락 뿐 아니라 본인 인증, 결제 등, 보다 포괄적이고 다양한 방면으로 활용성을 높일 수 있다.

 # ST(강점-위협) 전략

- 더 많은 데이터를 수집하고 학습시켜 정확도를 높여 보안성을 강화시켜 효율성을 높인다.

 # WO(약점-기회) 전략

- 좀 더 세밀한 기능들을 추가하여 하나의 도어락 뿐 아니라, 하나의 애플리케이션으로도 더 많은 도어락을 제어 할 수 있게 하여 실용성을 높일 수 있다.

 # SW(강점-약점) 전략

- 다양한 각도와 조명에서 데이터를 수집하여 학습시켜 오인식률을 감소시키고, 앱을 사용함으로써 절감한 비용을 투자하여 실제 도어락을 구현하여 상업성을 높일 수 있다.

7. 프로젝트 세부일정

7.1 팀원담당업무 및 일정

이름	담당업무
강경문	머신러닝 / 머신러닝 <-> Django서버 통신
이효섭	안드로이드 앱개발 / Django <-> 라즈베리파이 소켓통신
이채영	Django 서버구축/ 안드로이드 앱 개발/ 머신러닝
강두영	라즈베리파이 서버 구축/ 안드로이드 앱 개발 및 디자인

7.2 세부일정

	1주차	2주차	3주차	4주차
아이디어 회의 및 역할 분담				
라즈베리파이 분석 및 개발				
안드로이드 앱 개발				
서버 구축				
레트로핏 통신				
머신러닝 서버				
Socket 통신				
코드 추합 및 전체 연결				
앱 디자인				
최종 작동 확인				
최종 보고서 작성				

Xproject(OnOoN)

<일정>
2020/05/30
* 주제 선정
(스마트폰으로 얼굴을 인식하는 스마트 도어락)

2020/06/01
* 제안서 작성 (발표 : 강두영)
* 제안서 발표 URL :
https://www.youtube.com/watch?v=evSF0tX5LFg&feature=youtu.be
* 제안서 ppt URL :
https://drive.google.com/drive/folders/1y7Ew28heeLEZUsjv6ksNjD4UYYz5jewY

<역할분담>
* 강경문 : 머신러닝 서버 구축 / 머신러닝 <-> Django서버 통신
* 강두영 : 라즈베리파이 서버 구축/ 안드로이드 앱 개발 및 디자인
* 이효섭 : 안드로이드 앱개발 / Django <-> 라즈베리파이 소켓통신
* 이채영 : Django 서버구축/ 안드로이드 앱 개발/ 머신러닝

2020/06/08
* 서보모터 동작 확인하기 --ok
* 로그인, 회원가입 구현하기 --ok
* django서버 제작 --ok
* django 서버와 연결하기 --ok
* 머신러닝 공부 --ok

~2020/06/10
* 회원가입시 서버랑 통신 --ok
* 앱으로 사진을 찍은 후 저장되는 것 확인 --ok
* 사진을 여러장 찍을 수 있는지 확인 --ok
* 실제 도어락처럼 움직이는지 확인 --ok
* 얼굴인식 관련 머신런닝 공부 --ok

* 스위치와 서보모터 라즈베리파이에 연결해서 서보모터 제어 구현 --ok
* 팀 깃허브 만들기 (팀 깃허브 Url : https://github.com/Xproject-Team5)
    - 기존의 django github url: https://github.com/chea-young/OnOnO_django/commits/master
    - 기존의 android studio url : https://github.com/chea-young/OnOoN2

~2020/06/13
* 카메라 서버에 저장하기 --ok
* 로그인 실패시 화면 이동 안되게 하기 --ok
* 회원가입시 로그인 중복시 다시 페이지 이동 안되게 하기--ok
* 라즈베리파이 연결을 위한 socket 공부 --ok
* 사진 및 얼굴 data 수집 후, 머신러닝 모델 학습 --ok

~2020/06/16
* 최종보고서 앱, 서버부분 작성 시작
* socket 통신 제작
* open door 버튼 누룰 때 사진과 id 정보 서버로 보내기 --ok
* 사진 1,2,3 한 객체로 묶기 --ok
* 머신러닝 django에 연결하기
* 앱 꾸미기 시작

~2020/06/19
* 머신러닝연결 --ok
* 서버와 앱 통신 연결 --ok
* 최종보고서 일부분 작성하기 --ok

~2020/06/22
* 머신러닝 정확도 높이기 --ok
* 서보모터와 소켓 통신 확인해 보기(django 연결전에) --ok
* 라즈베리파이와 서버 --ok
* 최종보고서 작성 --ok

~2020/06/23

* 시연 영상 찍기
* 앱 디자인 하기
* 최종보고서 마무리
* 대본쓰기

7.3 참고문헌

웹	https://www.youtube.com/watch?v=9EJH7qC2mcQ
안드로이드	https://www.youtube.com/watch?v=9EJH7qC2mcQ
Django	https://12teamtoday.tistory.com/33
머신러닝	https://github.com/ukayzm/opencv/tree/master/face_recognition
하드웨어	https://www.kocoafab.cc/tutorial/view/354
소켓통신	https://seolin.tistory.com/98

8. 프로젝트 결론 및 느낀 점

3주간의 팀 프로젝트가 마무리 되었다.

팀원들 모두 이런 장기적인 팀 프로젝트는 개념이 생소하고 경험이 적었기에 어떻게 시작해야 할지 감이 잘 오지 않았다. 그래서 이전에 프로젝트를 한 선배들의 보고서를 참고하면서 아이디어를 구상했다. 주제를 정하는 일부터 매우 어려워 몇 일간의 고민을 거듭하였다. 주제를 정하고 나서도 막막한 기분이었다. 일단 주제를 정하고 큰 대목부터 생각해보기로 했다. 장고서버, 라즈베리파이, 안드로이드를 통한 앱 구현, 머신러닝을 이용한 얼굴인식, 서버와 라즈베리파이 사이의 통신 등 처음 접해 보는 여러 가지 기술들을 구현하면서 팀원들 모두 자신이 맡은 파트에 머리를 싸매며 시간을 쏟아 부었다.

처음 해본 장기 팀 프로젝트인 만큼 시행착오도 많이 겪고 처음에 생각했던 만큼 완성도가 나오지는 않았던 것 같다. 팀원들 간의 소통은 원활하고 잘 되었는데 프로그램과 서버 간의 통신은 잘 안되었다. 만들다 보니 시간도 빠듯했고, 급하게 완성하다보니 정신이 없었던 것 같다. 막상 완성하고 보니 오인식률, 공간적 한계 등 문제점들도 발견되었다.
그럼에도 포기하지 않고 끝까지 찾고 수정하는 일을 수없이 반복하여 값진 결과를 이루었다.
끝내고 나니 무언가 프로젝트를 해냈다는 성취감이 있었고, 스스로 찾아보고 연구하며 수업시간에 배웠던 웹서버, 소켓통신에 대해 더 깊이 공부 할 수 있던 계기가 되었던 것 같다. 그리고 처음 접해보았던 안드로이드 앱, 레트로핏, 머신러닝, 등 새로운 분야에 대해서도 배우게 되었고, 관심을 가지게 되었다.

이토록 하나의 일에 여럿이 오랜 시간 공들인다는 것이 쉽지 않으면서도 참 보람 찬 일이었다. 집단 지성, 팀워크, 효율성 등 코드를 작성하고 통신을 하고 하드웨어를 연결하는 일 외에도 중요한 요소가 참으로 많은 것이 팀 프로젝트임을 알게 되었다. 다음에도 프로젝트를 하게 된다면 이번 프로젝트를 바탕으로 더 좋은 결과를 얻을 수 있을 것 같다.

*Git 주소 : **https://github.com/Xproject-Team5**
*시연 영상 Youtube 주소 : **https://www.youtube.com/watch?v=3112z-fH1Xg&t**