사 이 버 보 안 캡 스 톤 디 자 인

스마트폰과 스마트워치를 활용한 사용자 인증

1395030 엄시우
1791224 심민주

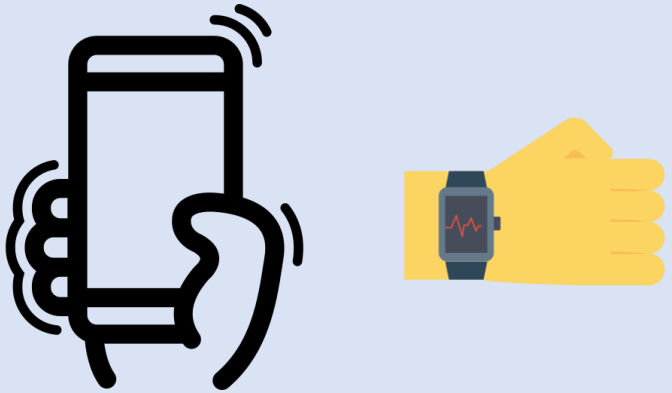
https://youtu.be/ksMr3n67U4g

01 프로젝트소개

1차 인증

지문 인식

2차 인증



스마트폰 & 스마트워치
를 활용한 사용자 인증

02 향후 개발 진행 순서

단계 1. 기기를 흔들어 사용자를 인증

단계 2. 정해진 모션을 통해 사용자 인증

단계 3. 서로 다른 사용자가 같은 모션을 취해도 사용자를 구분하여 인증

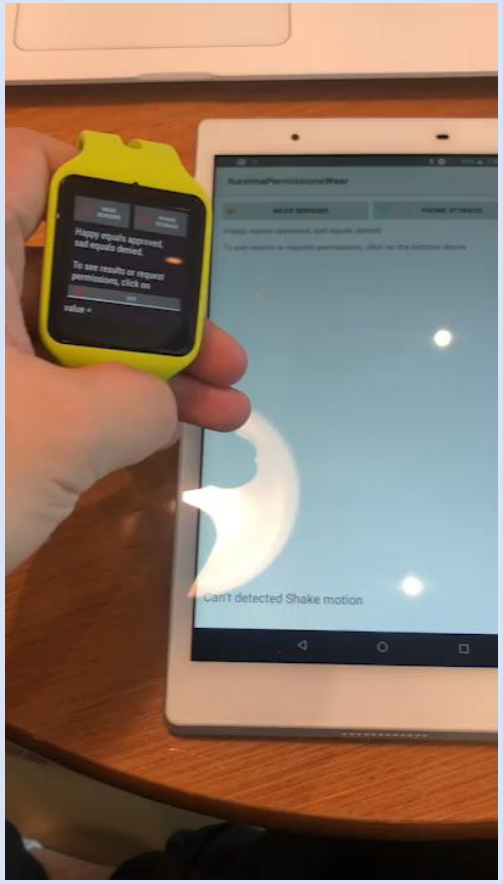
03 수행 내역

1. 워치와 스마트폰의 움직임이 감지되면 인증되는 부분 구현

2. 스마트 워치에서 스마트폰으로 신호 전달

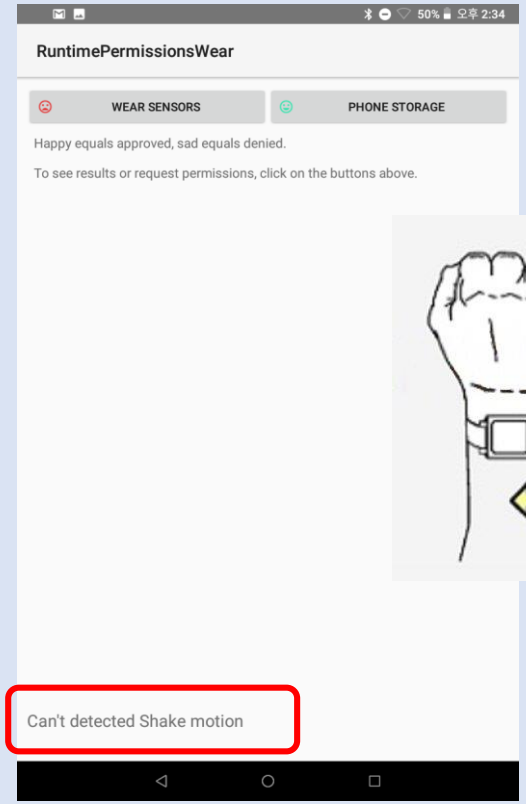
3. 스마트 워치와 스마트 폰의 센서값 수집(DB저장)

04 작동 영상

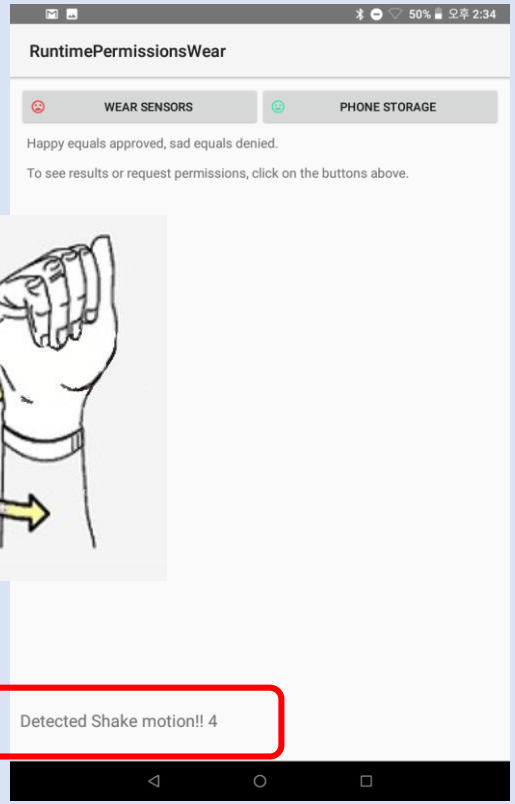

Happy equals approved,
sad equals denied.
To see results or request
permissions, click on
value =
Can't detected Shake motion

05 작동 이미지

스마트 워치 흔들때 마다 스마트 폰으로 값 전달

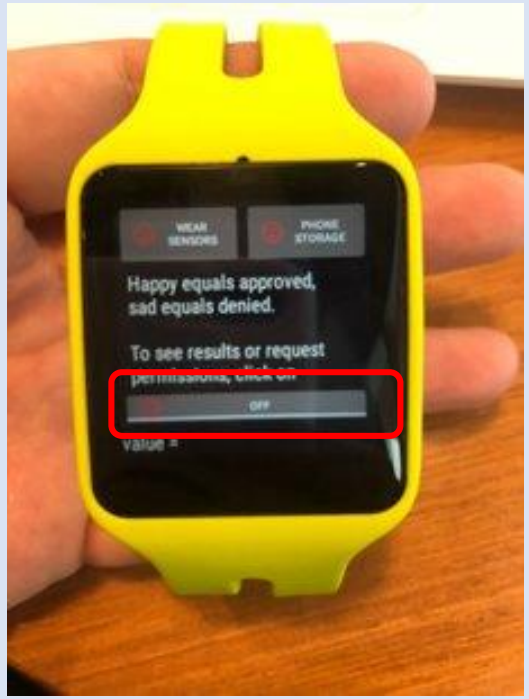


워치 shake 전 스마트폰



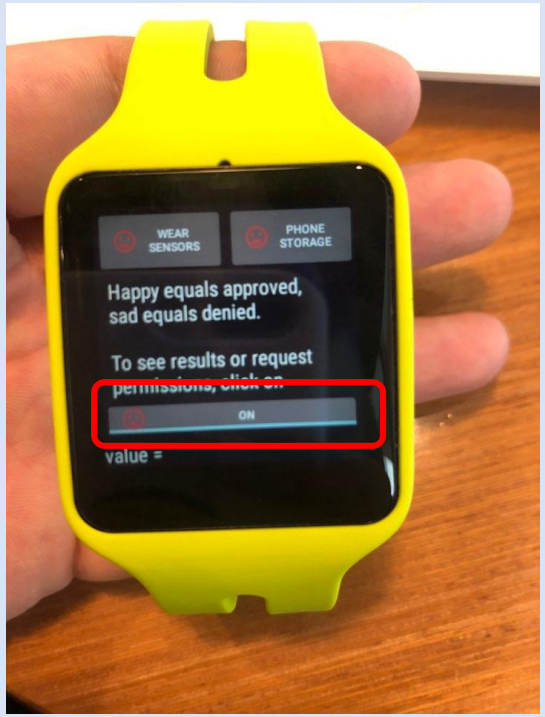
워치 shake 후 스마트폰

05 작동 이미지



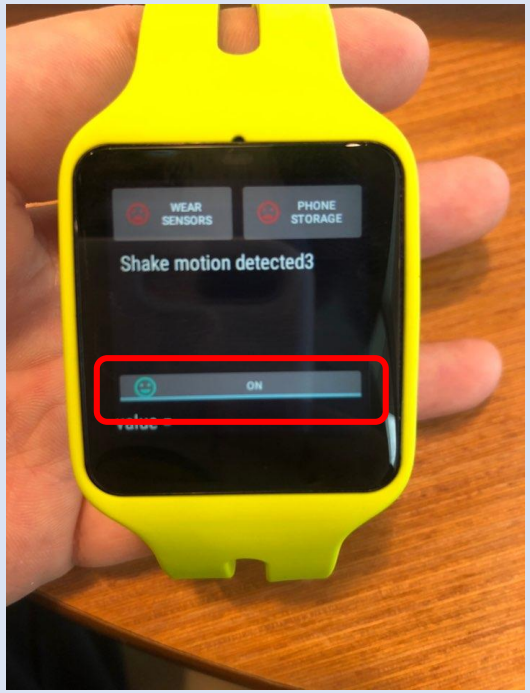
off 상태

워치 shake 신호 전달 불가



on 상태

워치 shake 신호 전달 가능 상태

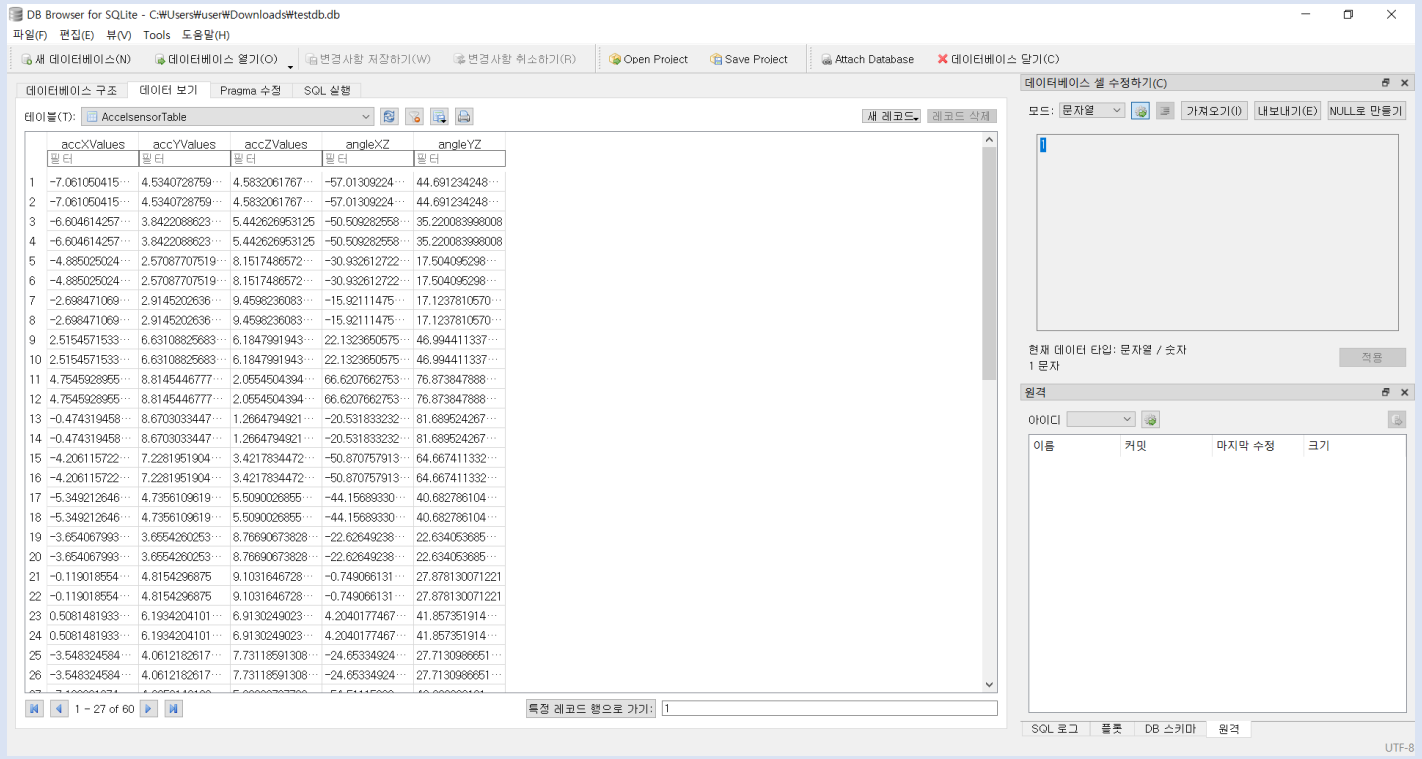


on 상태

워치 shake 신호 전달 진행 중

05 작동 이미지

가속도 센서 값 DB에 저장



06 코드 (워치)

```
float gravityX = axisX / SensorManager.GRAVITY_EARTH;
float gravityY = axisY / SensorManager.GRAVITY_EARTH;
float gravityZ = axisZ / SensorManager.GRAVITY_EARTH;
Float f = gravityX * gravityY * gravityZ *gravityX * gravityY * gravityZ;
double squareadD = Math.sqrt(f.doubleValue());
gForce = (float) squareadD;

if(gForce>SHAKE_THRESHOLD_GRAVITY){
    long currentTime = System.currentTimeMillis();
    Log.e( tag: "shake", Float.toString(gForce));
    if(mShakeTime + SHAKE_SKIP_TIME > currentTime){
        return;
    }

    mShakeTime = currentTime;
    if(oncheckedtgbutton){
        Shakecount += 1;
        logToUi("Shake motion detected" + Shakecount);
        DataMap dataMap = new DataMap();
        dataMap.putInt(Constants.KEY_COMM_TYPE, Constants.COMM_TYPE_SHAKE_RESULT_DATA );
        mshake_result_button.setCompoundDrawablesWithIntrinsicBounds(
                R.drawable.ic_permission_approved, top: 0, right: 0, bottom: 0);
        sendMessage(dataMap);
    }
```

06 코드 (워치)

```
float gravityX = axisX / SensorManager.GRAVITY_EARTH;
float gravityY = axisY / SensorManager.GRAVITY_EARTH;
float gravityZ = axisZ / SensorManager.GRAVITY_EARTH;
Float f = gravityX * gravityY * gravityZ *gravityX * gravityY * gravityZ;
double squareadD = Math.sqrt(f.doubleValue());
gForce = (float) squareadD;
```

- X, Y, Z 값 이용
- 워치를 흔들었을 때, 중력가속도 계산
- 계산한 값 > 중력, 워치를 흔든 것으로 가정하여 중력가속도를 계산

*gForce 가 중력가속도 값을 의미

06 코드 (워치)

```
float gravityX = axisX / SensorManager.GRAVITY_EARTH;
float gravityY = axisY / SensorManager.GRAVITY_EARTH;
float gravityZ = axisZ / SensorManager.GRAVITY_EARTH;
Float f = gravityX * gravityY * gravityZ *gravityX * gravityY * gravityZ;
double squareadD = Math.sqrt(f.doubleValue());
gForce = (float) squareadD;

if(gForce>SHAKE_THRESHOLD_GRAVITY){
    long currentTime = System.currentTimeMillis();
    Log.e( tag: "shake", Float.toString(gForce));
    if(mShakeTime + SHAKE_SKIP_TIME > currentTime){
        return;
    }

    mShakeTime = currentTime;
    if(oncheckedtgbutton){
        Shakecount += 1;
        logToUi("Shake motion detected" + Shakecount);
        DataMap dataMap = new DataMap();
        dataMap.putInt(Constants.KEY_COMM_TYPE, Constants.COMM_TYPE_SHAKE_RESULT_DATA );
        mshake_result_button.setCompoundDrawablesWithIntrinsicBounds(
                R.drawable.ic_permission_approved, top: 0, right: 0, bottom: 0);
        sendMessage(dataMap);
    }
```

06 코드 (워치)

```
if(gForce>SHAKE_THRESHOLD_GRAVITY){
    long currentTime = System.currentTimeMillis();
    Log.e( tag: "shake", Float.toString(gForce));
    if(mShakeTime + SHAKE_SKIP_TIME > currentTime){
        return;
    }

    mShakeTime = currentTime;
```

- 워치를 흔들때마다 계속 카운트 X

- 한번 흔든 후, 0.5초의 시간간격을 두고 흔들어야 카운트가 증가

06 코드 (워치)

```
float gravityX = axisX / SensorManager.GRAVITY_EARTH;
float gravityY = axisY / SensorManager.GRAVITY_EARTH;
float gravityZ = axisZ / SensorManager.GRAVITY_EARTH;
Float f = gravityX * gravityY * gravityZ *gravityX * gravityY * gravityZ;
double squareadD = Math.sqrt(f.doubleValue());
gForce = (float) squareadD;

if(gForce>SHAKE_THRESHOLD_GRAVITY){
    long currentTime = System.currentTimeMillis();
    Log.e( tag: "shake", Float.toString(gForce));
    if(mShakeTime + SHAKE_SKIP_TIME > currentTime){
        return;
    }

    mShakeTime = currentTime;
    if(oncheckedtgbutton){
        Shakecount += 1;
        logToUi("Shake motion detected" + Shakecount);
        DataMap dataMap = new DataMap();
        dataMap.putInt(Constants.KEY_COMM_TYPE, Constants.COMM_TYPE_SHAKE_RESULT_DATA );
        mshake_result_button.setCompoundDrawablesWithIntrinsicBounds(
                R.drawable.ic_permission_approved, top: 0, right: 0, bottom: 0);
        sendMessage(dataMap);
    }
```

06 코드 (워치)

```
if(oncheckedtgbutton){
    Shakecount += 1;
    logToUi("Shake motion detected" + Shakecount);
    DataMap dataMap = new DataMap();
    dataMap.putInt(Constants.KEY_COMM_TYPE, Constants.COMM_TYPE_SHAKE_RESULT_DATA );
    mshake_result_button.setCompoundDrawablesWithIntrinsicBounds(
            R.drawable.ic_permission_approved,  top: 0,  right: 0,  bottom: 0);
    sendMessage(dataMap);
```

- oncheckedtgbutton -> 토글버튼 on/off(on=true / off = false) 상태 확인 후, on 경우, Shakecount를 1 증가
- SendMessage를 통해 스마트폰으로 데이터를 전송
- 전송 데이터는 datamap을 활용하여 <key, value>처럼 활용합니다.
- Count 값 전달 X, 흔들렸다는 신호 전송

06 코드(스마트폰)

```
public void onMessageReceived(MessageEvent messageEvent) {
    Log.d(TAG, msg: "onMessageReceived(): " + messageEvent);

    String messagePath = messageEvent.getPath();

    if (messagePath.equals(Constants.MESSAGE_PATH_PHONE)) {
        DataMap dataMap = DataMap.fromByteArray(messageEvent.getData());

        int commType = dataMap.getInt(Constants.KEY_COMM_TYPE, i: 0);

        if (commType == Constants.COMM_TYPE_SHAKE_RESULT_DATA){
            shakecount += 1;
            mShake_result_textView.setText("Detected Shake motion!! " + shakecount);
        } else {
            Log.d(TAG, msg: "Unrecognized communication type received.");
        }
    }
}
```

워치로 부터 데이터를 전달받고 카운트 증가시키고 출력

07 개발 목표 대비 개발 진척도

 # 개발 목표

1단계 (흔들기) -> 2단계 (모션) -> 3단계 (사용자 인식)

 # 개발 진척도

- 모든 단계에 공통적으로 필요한 워치와 폰의 통신 부분 해결에 많은 시간 소비 -> 개발 진척도 제자리 걸음

- 현재, **통신부분 문제점 해결**

- 1단계(흔들기) 부분의 마무리 작업을 진행 중

**** 통신부분이 해결된 만큼 앞으로 진행속도가 빨리질 것이라 예상**

08 향후 개발 계획

		10주차	11주차	12주차	13주차	14주차	15주차
1단계	구현	O					
	테스트	O	O				
2단계	조사	O	O				
	구현		O	O			
	테스트			O	O		
3단계	조사		O	O			
	구현			O	O		
	테스트				O	O	
UI					O	O	
최종 발표 준비						O	O

THANK YOU