<커스터마이징 자동 삼각대 – 블루투스 기능 보고서(3 월 4 주차)>

블루투스 기능 확인을 위해 아두이노 우노, 레오나르도 보드 와 HC-06 블루투스 모듈 이용.

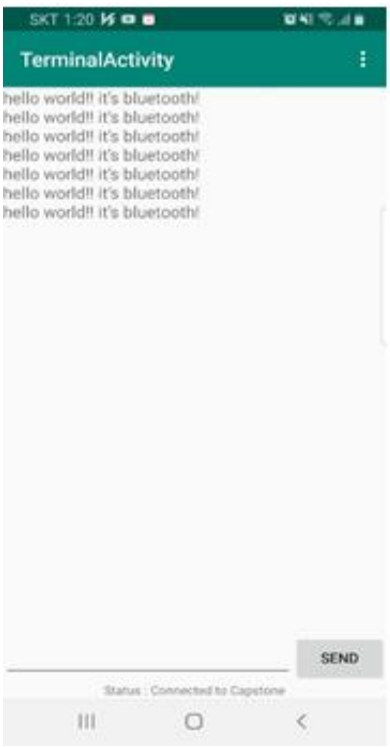
안드로이드의 블루투스 기능을 위해 'BluetoothSSPLibrary' 라이브러리 사용.

참고 자료: https://github.com/akexorcist/Android-BluetoothSPPLibrary

1) 아두이노 -> 안드로이드로의 간단한 문자열 전송은 확인.

2) 그러나 안드로이드 -> 아두이노로의 데이터 전송 시 어플이 자동 종료되는 오류가 남.

- 확인 결과, 'BluetoothSSPLibrary' 라이브러리 데모에서도 또한 같은 오류 발생.
- 따라서 오류 파악 및 다른 블루투스 라이브러리 또한 조사 필요.
- 추가로 안드로이드에서 아두이노와 같은 MCU 로 데이터를 전송하여 제어하는 전반적인 자료 조사 필요.



<아두이노 -> 안드로이드 문자열 전송>

<안드로이드 -> 아두이노로 데이터 전송 시>

<커스텀마이징 자동 삼각대 – 블루투스 기능 보고서(4 월 2 주차)>

<개발 현황>

4 월 1 주차에 블루투스 통신 페어링 기능 구현, 그러나 좌표값 송신 불가. -> 이에 따라 코드의 구조 분석 및 정리 작업 실시.

결과적으로 코드 구조 분석을 통해 얼굴 정보를 받는 역할을 하는 FrameReturn 인터페이스 설정 및 OnFrame 메소드 생성. OnFrame 메소드 인자를 Bitmap, FirebaseVisionFace, FrameMetadata, GraphicOverlay 로 선언.

FrameReturn 인터페이스(OnFrame 메소드)는 얼굴 감지 프로세스가 동작할 때 Frame 단위로 지속적으로 찍어 image 의 형태로 얼굴 정보를 MainActivity(앱 동작 부분)에 가져오도록 함.

- MainActivity 에 implement 한 FrameReturn 인터페이스의 OnFrame 메소드 코드

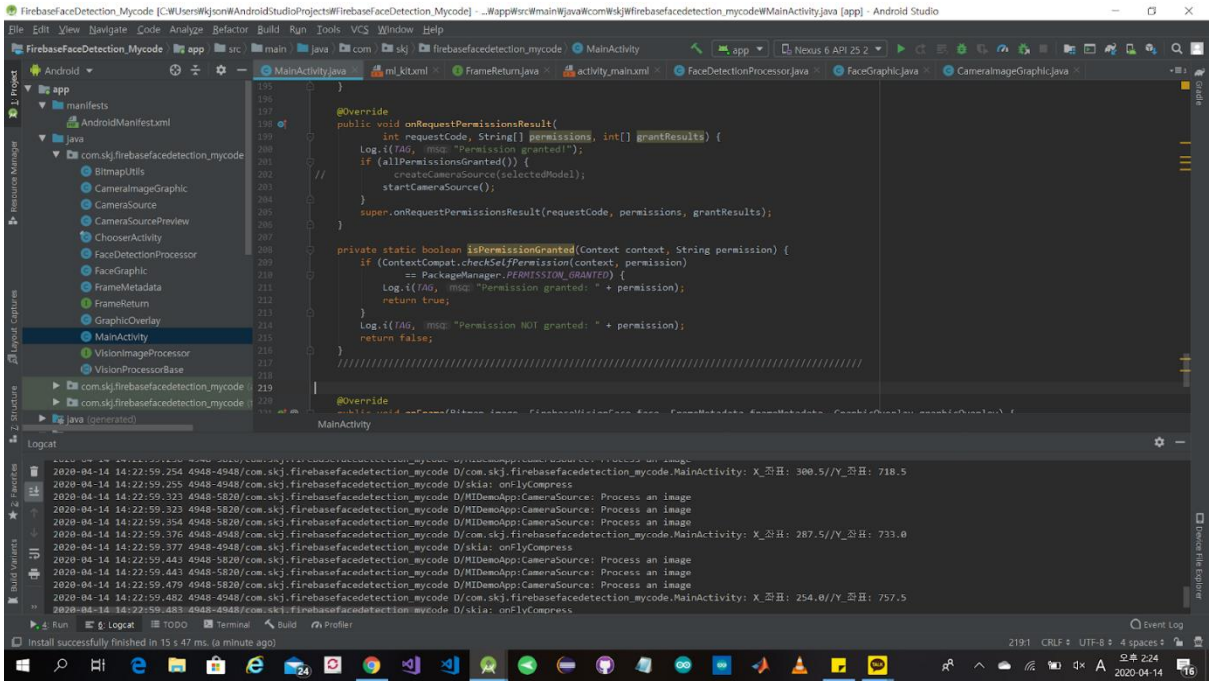
```
-    @Override
public void onFrame(Bitmap image, FirebaseVisionFace face, FrameMetadata
    frameMetadata, GraphicOverlay graphicOverlay) {
    originalImage = image;
    FaceGraphic faceGraphic = new FaceGraphic(graphicOverlay);
    float onFrame_X = faceGraphic.translateX(face.getBoundingBox().exactCenterX());
    float onFrame_Y = faceGraphic.translateY(face.getBoundingBox().exactCenterY());

    X_f = Float.toString(onFrame_X);
    Y_f = Float.toString(onFrame_Y);

    Log.d(this.getClass().getName(), "X_좌표: " + X_f + "//" + "Y_좌표: " + Y_f);
}
```

즉, 얼굴에 대한 정보들을(좌표, 눈의 감김 정도, 얼굴 id 등) MainActivity 에서 다룰 수 있도록 함.

- 얼굴 좌표 Log 출력한 화면. (하단에 Logcat 화면 참조.)



FrameReturn 인터페이스를 이용함으로 BitmapUtils, CameraImageGraphic 클래스 생성 및 VisionProcessorBase, FaceDetectionProcessor 클래스 수정.

- **클래스 생성**

1. BitmapUtils 클래스

   FrameReturn 의 OnFrame 메소드 인자 중 Bitmap 인자를 받게 해주도록 background 에서 돌아가는 클래스.

   즉, 얼굴 감지를 위한 image process(이미지 생성 부분.)와 detectInVisionImage(이미지 생성을 감지하여 ML Kit 가 실제적으로 실행되는 부분.)이 실행되는 VisionProcessorBase 클래스에 선언. Image process 작업에 image 를 만들어주는 역할.

2. CameraImageGraphic 클래스

   어플이 작동 후 얼굴 감지 시 카메라를 통한 이미지 생성 및 현재 이미지의 bitmap 정보를 받아 제공해주는 클래스. (클래스 내 drawBitmap 명령어 참조.)

   쉽게 정리하면 실시간 동기화 개념으로 생각하면 될 듯.

- **클래스 수정**

1. VisionProcessorBase 클래스

   크게 processing image/framemetadata 와 latest image/framemetadata 부분으로 접근.

   즉 processing image/framemetadata 가 null 인지 아닌지 기준으로 image 를 생성.

   이때 latest Image/framemetadata 변수를 통해 즉각 image 생성 동작을 제어. 따라서 카메라 실행 시 약간 끊기는 현상 발생. 크게 무리가 되지 않을 것으로 예상.

   결국 하나의 frame 으로의 동작을 취하기 위해 image 생성 -> processing image/framemetadata null 유무 파악.

   detectInVisionImage(이미지 생성을 감지하여 ML Kit 가 실제적으로 실행되는 부분.) 메소드 수정. FrameReturn 인터페이스의 OnFrame 메소드 동작을 위해 메소드 인자를 맞춰줌. (Bitmap 변수 추가.)

   detectInVisionImage 메소드 내 OnSuccessListener 의 Success 시, 즉 firebasevision image 가 감지 시 동작하는 OnSuccess 메소드에도 Bitmap 변수 추가.

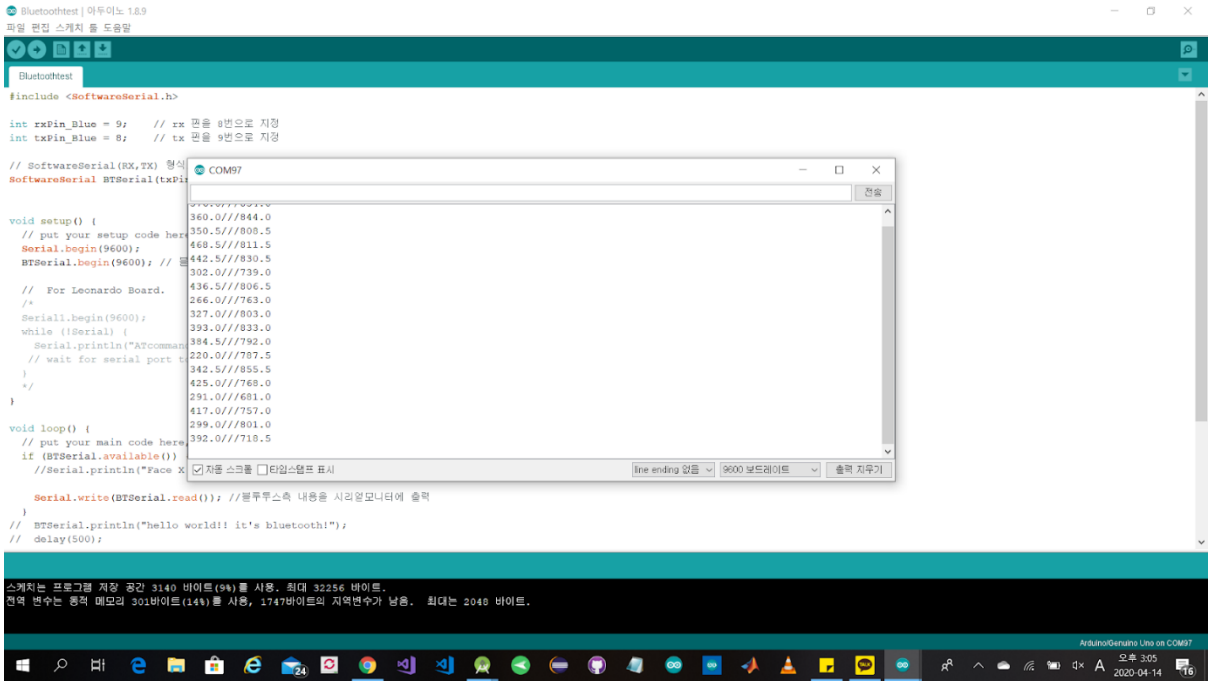
2. FaceDetectionProcessor 클래스

   VisionProcessBase 클래스를 기반으로 얼굴 감지가 실질적으로 작동되는 클래스. 때문에 FrameReturn 인터페이스가 여기에서 선언되어 돌아가야 함. 전반적인 동작을 위해 위에서 설명한 CameraImageGraphic 클래스도 여기에 선언.

   FaceDetectionProcessor 클래스 내의 OnSuccess 메소드. (VisionProcessBase 클래스의 확장.)

<결과>

Logcat 을 통해 좌표가 print 되는 것을 확인 후, 아두이노 보드를 이용한 hc-06 블루투스 모듈로 간단한 얼굴 좌표 블루투스 전송을 실험함.



위 자료를 통해 안드로이드 -> mcu 보드간 얼굴 좌표 값 블루투스 송신이 제대로 이루어지는 것을 확인.

또한 카메라 실행 시 끊김 현상이 발생했지만 google ML Kit 문서를 통해 FAST 모드를 지정할 수 있다는 것을 파악. FirebaseVisionFaceDetectorOptions 객체에서 FAST 모드 설정하여 실행 결과 끊김의 정도가 많이 내려가는 것을 확인.

<다음 개발 과제>

블루투스 송신을 버튼 이벤트 형식으로 눌렀을 때 송신되는 동작으로 구현함.

그러나 어플 실행 후 얼굴 감지 시 자동으로 좌표값을 송신하는 동작 구현이 필요.

버튼 이벤트는 추후 얼굴 트래킹이 자동으로 되었지만 좀 더 정확한 조정이 필요할 때 사용되는 최적화 기능으로 두는 방향으로 생각.

<개발 참고 자료>

Github: https://github.com/ateymoori/android_face_detection

Demo video: https://www.youtube.com/watch?v=J4-t1SiGXXQ&feature=youtu.be