종합설계 프로젝트 수행 보고서

프로젝트명	건강 모니터링을 위한 스마트 반려동물 의류
팀번호	**S1-7**
문서제목	**수행계획서( )** **2차발표 중간보고서( )** **3차발표 중간보고서( )** **최종결과보고서(O)**

2020.12.03

팀원 : 류승재 (팀장), 김영서, 이상은
지도교수 : 최진구 교수 (인)

문서 수정 내역

작성일	대표작성자	버전(Revision)	수정내용	
2020.01.21	류승재(팀장)	1.0	수행보고서	최초작성
2020.02.28	류승재(팀장)	2.0	2차발표자료	설계서추가
2020.04.30	류승재(팀장)	3.0	3차발표자료	시험결과추가
2020.06.25	류승재(팀장)	4.0	4차발표자료	시험결과 수정
2020.12.03	류승재(팀장)	최종	최종보고서	논문 및 보고서

문서 구성

진행단계	프로젝트 계획서 발표	중간발표1 (2월)	중간발표2 (4월)	학기말발표 (6월)	최종발표 (10월)
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~3) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~4) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~5) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~7) 참고자료	Ⅰ Ⅱ Ⅲ

이 문서는 한국산업기술대학교 컴퓨터공학부의 “종합설계”교과목에서
프로젝트 “건강 모니터링을 위한 스마트 반려동물 의류” 을 수행하는
(S1-7 류승재, 김영서, 이상은)들이 작성한 것으로
사용하기 위해서는 팀원들의 허락이 필요합니다.

목 차

Ⅰ. 서론

1. 작품선정 배경 및 필요성

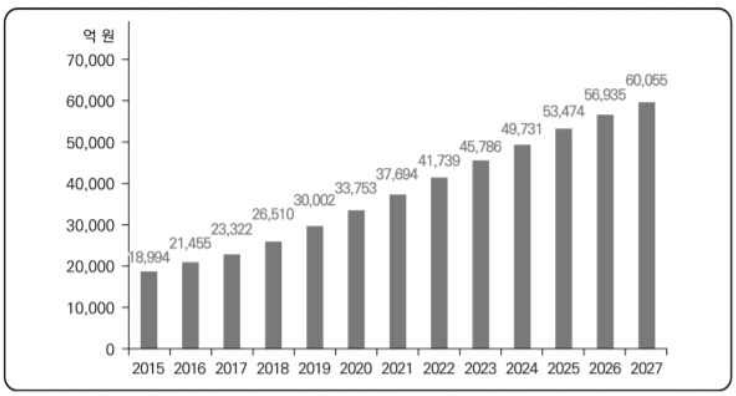
현대사회에는 핵가족화와 1인 가구의 증가, 저출산 및 고령화 현상으로 인해 반려동물과 삶을 함께하는 가구의 수가 급증하였다. 2017년 기준 전국 1,952만 가구 중 29.4%인 574만 가구에서 개 632만 마리, 고양이 243만 마리를 기르고 있는 것으로 추정되었다. 반려동물의 수는 꾸준히 증가하여 2027년에는 1,320만 마리에 이를 것으로 추정된다.

또한, 병원을 가지 않고도 건강관리를 할 수 있는 센서와 결합한 웨어러블 스마트 기기도 크게 발전했다. 인간을 위한 스마트 기기는 많은 발전을 했지만, 반려동물의 건강관리에 초점을 맞춘 스마트 기기는 상용화되어 있지 않다는 것을 알게 되었다.

이러한 현대 사회 배경을 이유로 반려동물과 함께하는 가정 수의 증가에 따라 반려인과 수의사가 반려동물의 건강상태를 실시간으로 모니터링 할 수 있는 스마트 기기가 필요하다고 생각했다.

반려동물과 함께하는 가정의 수 증가와 함께 사회의 소득도 증가하면서 반려동물 케어와 같은 반려동물 관련 산업은 크게 성장하고 있음을 볼 수 있다. 이는 반려인의 지출과 연관이 되는데 반려동물을 위한 지출의 증가를 의미한다. 이러한 사회 현상 속에서 반려동물의 헬스케어를 위한 스마트 기기를 제작한다면 적지 않은 수익도 얻을 수 있을 것으로 전망된다.

〈반려동물 연관산업 규모 전망〉



대부분 반려동물은 외적으로는 많은 털을 가지고 있고, 내적으로도 인간과 차이점이 있기 때문에 기존 사람을 위한 건강관리 기기를 이용한 건강관리가 현실적으로 불가능하므로 반려동물 건강관리를 위한 새로운 스마트 기기의 제작이 필요하다.

2. 기존 연구/기술동향 분석

반려동물을 위한 스마트 기기와 관련된 논문을 먼저 살펴보았다. 우리가 제작하려는 스마트 기기와 가장 가깝다고 볼 수 있는 논문 Smart Pet Clothing for Monitoring of Health and Mood(Yu-Jin Lin 등, 2018)에서는 스마트 의류 제작을 통한 호흡 신호와 심전도 신호 탐지, 반려동물의 기분 탐지를 목표로 하였고 CNN 알고리즘을 이

용한 신호 탐지 정밀도 향상까지 이루어내었다.

그다음 기존 출시되어있는 기기 및 기술을 분석해보았다. 첫째로 SKT의 T-pet이 있다. 반려동물의 목줄에 연결하여 반려인의 휴대폰으로 반려동물의 현재 위치를 확인할 수 있고 활동량 및 휴식량 분석을 통해 건강상태를 확인할 수 있다. 둘째로는 Fit Bark가 있다. 반려동물의 목에 채우는 장치를 통해 측정된 데이터를 반려인의 휴대폰으로 실시간으로 확인할 수 있고 운동량과 체중에 맞는 식사량 추천을 해준다. 셋째로는 Whistle이 있다. 이것은 반려동물의 목줄에 부착할 수 있는 피트니스 트래킹 디바이스로 반려동물의 위치 정보 및 활동량을 기록해준다.

기존 기기와 기술 분석을 통해 대부분의 장치는 반려동물의 목에 위치시키게 되어있는 것을 확인할 수 있었고 활동량, 휴식량, 위치 정보 등을 기록하는 것을 알 수 있었다.

3. 개발 목표

반려동물의 건강 및 현재 상태를 반려인과 수의사가 실시간으로 모니터링 할 수 있는 스마트 의류를 개발할 것이다.

반려동물에게 심장병은 종종 나타나는 질병 중 하나이다. 통계 보고서에 따르면 개의 11%와 고양이의 15%가 선천적으로 또는 노화에 따른 심장병의 증상으로 고통받고 있다고 한다. 고령의 고양이는 비대성 심근 병증으로 고통받는 경우가 있고 특정 증상이 없이 갑작스러운 사망을 유발할 수가 있으며, 고령의 개는 심장 판막 협착증이 발생하는 경우가 있다.

이러한 통계를 통해 심장병 조기 탐지 및 추적이 필요하다고 생각해 반려동물의 심전도를 측정하고 카메라로 실시간 영상송출을 할 수 있는 디바이스를 제작하고 측정 결과를 저장하는 서버를 구현하여 반려인과 수의사가 측정된 건강상태를 실시간으로 확인할 수 있는 어플리케이션을 제작할 것이다.

반려동물의 피부에는 사람의 털과는 비교할 수 없을 만큼의 많은 양의 털이 존재한다. 생체 신호 감지를 위한 센서를 부착하는 것이 털로 인해 방해될 수 있으므로 비교적 털이 적고 혈류가 많이 흐르는 귀와 같은 곳에 집게형으로 제작된 ppg센서를 통해 심박 수를 측정한다.

의류에 부착된 카메라로는 실시간 스트리밍을 통해 반려동물의 주변 상태를 확인할 수 있게 할 것이다.

이 반려동물 스마트 의류를 통하여 반려동물의 상태를 즉시 확인하고 위험 상황 발생 시 어플리케이션 알림을 통해 빠른 대처를 가능하게 할 것이다. 이를 통하여 반려인의 반려동물을 향한 불안감도 해소될 것으로 예측한다.

4. 팀 역할 분담

이 름	학 번	주 요 역 할
류승재(팀장)	2014156013	심박 센서 구현, 하드웨어 전반 관리
김영서	2017156006	어플리케이션 제작, 보고서 작성
이상은	2017156023	라즈베리파이 소켓통신 구현, 아두이노 구현

5. 개발 일정

2020년 1월 6일 ~ 2월 23일 : 라즈베리파이 소켓통신 구현, fpv 카메라 실시간 스트리밍 구현, 하드웨어 제작, 어플리케이션 제작

2020년 2월 24일 ~ 2월 29일 : 디바이스 및 어플리케이션 테스트

2020년 3월 1일 ~ : 디바이스 및 어플리케이션 보수작업

2020년 4월 10일 ~ : 기존 프런트 엔드 ECG회로를 PPG로 변경, 프로토타입 완성

6. 개발 환경

1) Application

Android Studio를 이용하고 Java언어를 사용하여 application을 구현한다.

Android 6.0부터 8.0버전까지 구현한다.

2) Server

PyCharm을 이용해 라즈베리파이 원격 빌드 환경을 설정한다.

3) Hardware

Raspberry Pi 4 B+ + PPG(심장박동) 센서 + V2(카메라)모듈 + 아두이노 나노를 사용한 측정 디바이스를 개발한다.

심장박동 센서는 PPG센서로 구성된다. PPG센서는 혈액이 몰리는 곳, 강아지의 경우 귀 또는 실험적으로 심장박동측정이 잘되는 곳에 클립식이나 밴드형식으로 부착한다.

카메라 영상은 휴대폰의 핫스팟을 이용하고 V2모듈을 사용해 실시간 스트리밍을 진행한다. 코드는 python언어를 사용한다.

라즈베리파이에는 ADC(anlog digital convert)가 없으므로 아두이노에서 센서를 측정한다.

아두이노와 라즈베리파이는 usb연결을 통한 시리얼 통신, 라즈베리파이와 어플리케이션은 소켓통신을 한다.

아두이노 코드는 C언어를 사용한다.

Ⅱ. 본론

1. 개발 내용

1) Software

① Application

어플리케이션을 켜고 소켓 통신할 ip(내부 ip)를 쓰고 연결을 누르면 FPV카메라의 실시간 영상이 상단에 표시된다. 라즈베리파이에서 보내는 데이터를 받아서 표시하게 된다. 이 데이터는 아두이노에서 받아온 심장박동 데이터로 건강상태 모니터링으로 사용된다. 실시간 그래프로 심장박동도를 확인할 수 있게 개발한다.

② Server

생체로부터 하드웨어 장치가 받아온 심박도의 연결 상태를 확인할 수 있도록 개발한다.

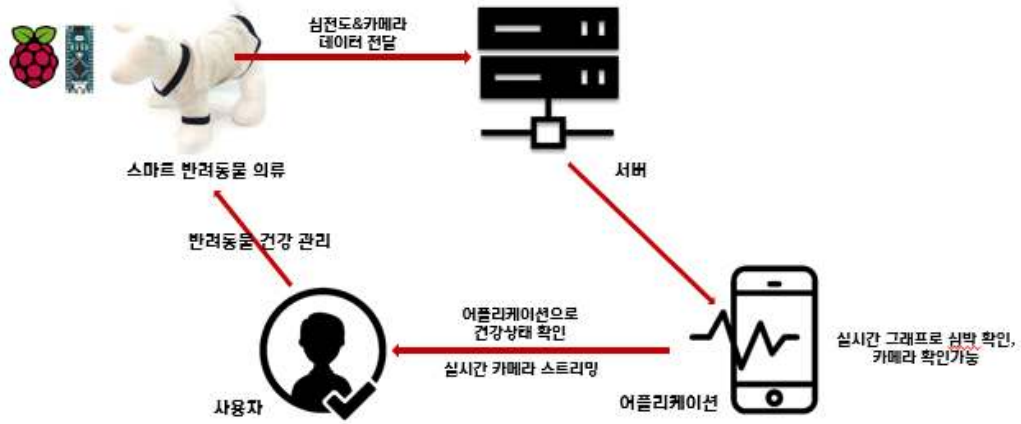
2) Hardware

Raspberry Pi 4 B+ + PPG 센서 + FPV 카메라 + 아두이노 나노를 이용한 반려동물 건강상태 측정 디바이스를 개발한다.

심장박동 센서는 PPG센서로 구성된다. PPG센서는 혈액이 몰리는 곳, 강아지의 경우 귀 또는 실험적으로 심장박동측정이 잘되는 곳에 클립식이나 밴드형식으로 부착한다.

V2 카메라 모듈을 사용하여 어플리케이션과 실시간 스트리밍을 진행한다.

어플리케이션과 라즈베리파이는 휴대폰의 와이파이(핫스팟)로 연결된다. 라즈베리파이에서 공유기가 되어 휴대폰으로 접속할 수 있으면 좋지만, 그 부품은 구하기가 힘들어 휴대폰의 핫스팟으로 대체한다. 휴대폰과 라즈베리파이는 내부 ip로 접속할 수 있다. 휴대폰 어플리케이션에서 라즈베리파이가 카메라를 스트리밍 하고 있는 서버에 접속하면 실시간 카메라를 볼 수 있고, 아두이노로 강아지에게 측정한 전압 데이터(심전도)를 라즈베리파이로 전송하면 라즈베리파이에서 이 데이터를 휴대폰으로 소켓 통신을 통해 보내 어플리케이션에서 확인할 수 있다.



2. 문제 및 해결방안

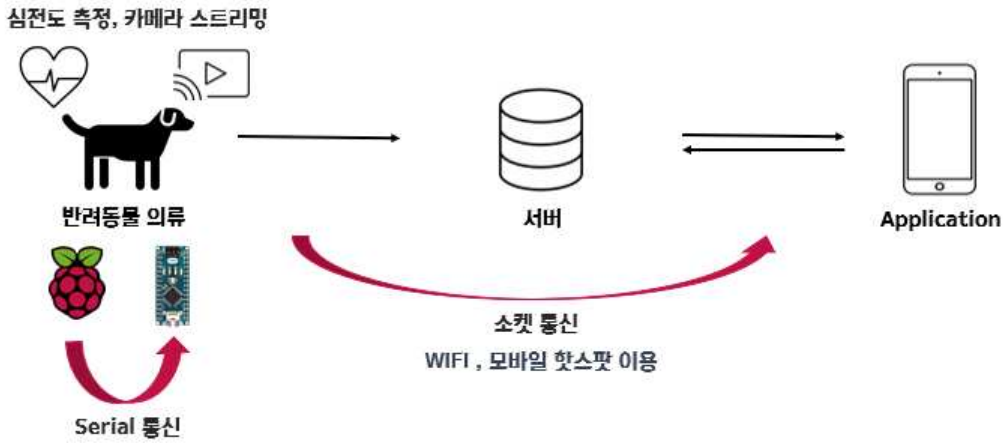
1) 반려동물의 피부에는 사람의 털과는 비교할 수 없을 만큼의 많은 양의 털이 존재한다. 생체 신호 감지를 위해 센서를 부착하는 것이 털로 인해 방해될 수 있으므로 이를 계측기와 증폭기 두 개의 필터로 특수 설계 구성된 프런트 엔드 ECG 회로를 통해 공통모드 노이즈를 제거하여 생체 신호를 증폭시켜 문제를 극복하고자 한다.
→ 하지만 계발 단계에서 문제가 발생하여 PPG센서를 이용하게 되었다. (Ⅱ5.1.1에서 계속)

2) 라즈베리파이에는 ADC(anlog digital convert)가 없으므로 아두이노 나노를 사용하여 아두이노에서 센서를 측정한다.

3) 어플리케이션과 라즈베리파이를 연결할 때 휴대폰의 핫스팟으로 접속할 수 있도록 한다.

3. 시험시나리오

PPG 센서를 사용하여 반려동물의 심장박동을 측정한다.
V2 카메라모듈을 사용하여 어플리케이션과 실시간 스트리밍을 진행한다.
어플리케이션과 라즈베리파이는 휴대폰의 와이파이(핫스팟)으로 연결된다. 라즈베리파이에서 공유기가 되어 휴대폰으로 접속할 수 있으면 좋지만, 그 부품은 구하기가 힘들어 휴대폰의 핫스팟으로 대체한다. 휴대폰과 라즈베리파이는 내부 ip로 접속할 수 있다. 휴대폰 어플리케이션에서 라즈베리파이가 카메라를 스트리밍 하는 서버에 접속하면 실시간 카메라를 볼 수 있고, 아두이노로 강아지에게 측정한 데이터(심박)를 라즈베리파이로 전송하면 라즈베리파이에서 이 데이터를 휴대폰으로 소켓 통신을 통해 보내 어플리케이션에서 그래프로 표기한다.
어플리케이션을 켜고 소켓 통신할 ip(내부 ip)를 쓰고 연결을 누르면 카메라의 실시간 영상이 상단에 표시된다. 라즈베리파이에서 보내는 데이터를 받아서 표시하게 된다. 이 데이터는 아두이노에서 받아온 심전도 데이터로 건강상태 모니터링으로 사용된다. 실시간 그래프로 심장박동 상태를 확인할 수 있다.



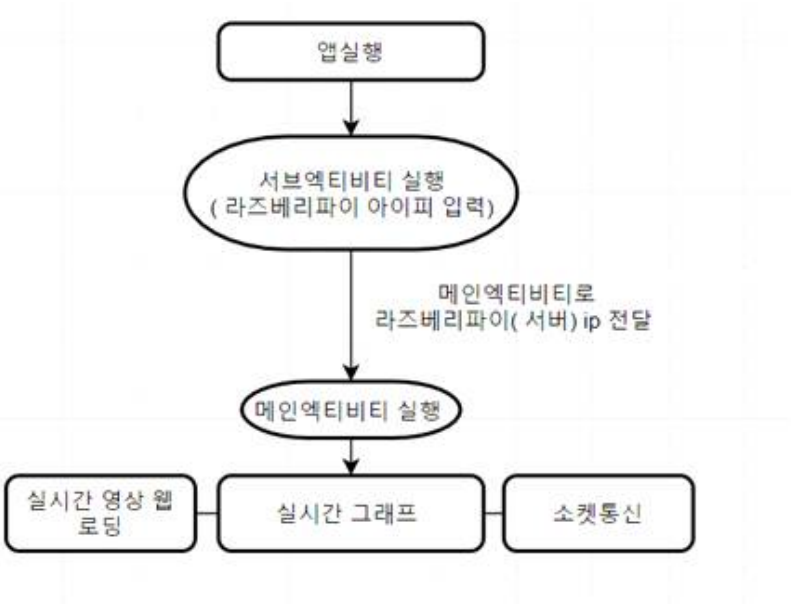
4. 상세 설계

1) Software

① Application

안드로이드 스튜디오 사용 (java언어 사용)

- 구성도

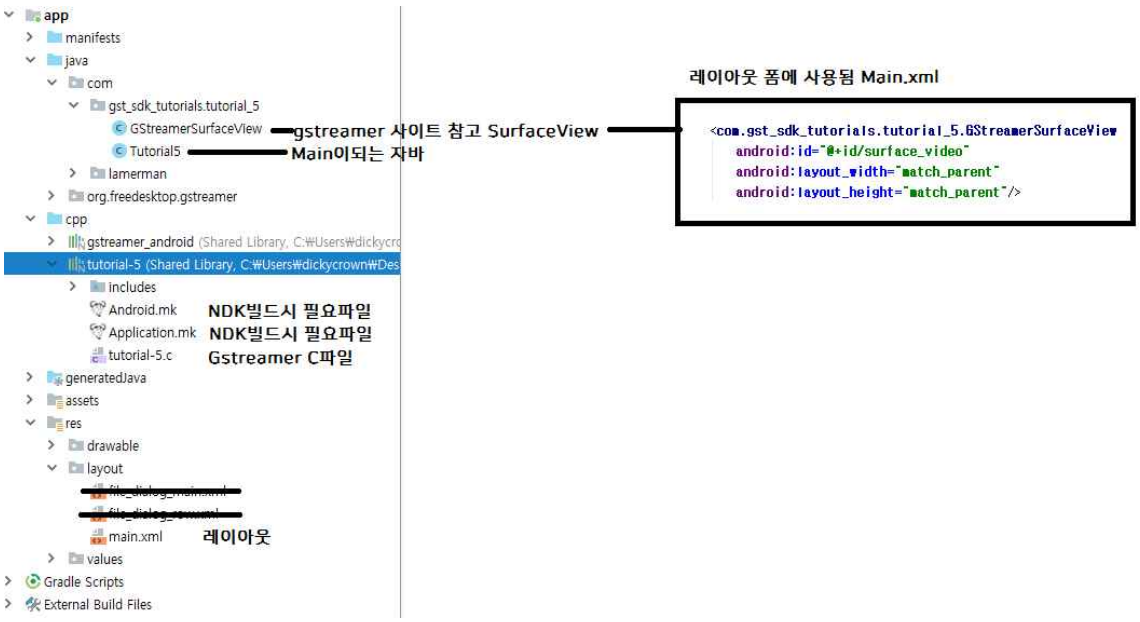


· Gstreamer

→ JAVA용 라이브러리가 없음.

Gstreamer를 돌리기 위해서는 C코드를 JAVA로 빌드해줘야한다.(NDK 빌드)

- 전체 구성



· Tutorial-5.c

```
static void gst_native_init (JNIEnv* env, jobject thiz, jstring addr, jstring strport) {
  CustomData *data = g_new0 (CustomData, 1);
  SET_CUSTOM_DATA (env, thiz, custom_data_field_id, data);
  const gchar *char_uri1 = (gchar*)(*env)->GetStringUTFChars (env, addr, NULL);
  const gchar *char_uri2 = (gchar*)(*env)->GetStringUTFChars (env, strport, NULL);
  pipeline=g_strjoin ("","tcpclientsrc host=", char_uri1, " port=", char_uri2, " ! gdpdepay ! rtph264depay ! avdec_h264 ! videoconvert ! autovideosink sync=false",NULL);
  (*env)->ReleaseStringUTFChars (env, addr, char_uri1);
  (*env)->ReleaseStringUTFChars (env, strport, char_uri2);
  GST_DEBUG_CATEGORY_INIT (debug_category, "tutorial-5", 0, "Android tutorial 5");
  gst_debug_set_threshold_for_name("tutorial-5", GST_LEVEL_DEBUG);
  GST_DEBUG ("Created CustomData at %p", data);
  data->app = (*env)->NewGlobalRef (env, thiz);
  GST_DEBUG ("Created GlobalRef for app object at %p", data->app);
  pthread_create (&gst_app_thread, NULL, &app_function, data);
```

→ 매개변수 서버의 IP, PORT를 입력받아 해당 pipeline을 실행할 수 있도록 함.

· Tutorial5.java

```
public class Tutorial5 extends Activity implements SurfaceHolder.Callback {
    private native void nativeInit(String addr, String vport);     // Initialize native code, build pipeline, etc
    private native void nativeFinalize(); // Destroy pipeline and shutdown native code
    private native void nativeSetUri(String uri); // Set the URI of the media to play
    private native void nativePlay();     // Set pipeline to PLAYING
    private native void nativePause();    // Set pipeline to PAUSED
    private static native boolean nativeClassInit(); // Initialize native class: cache Method IDs for callbacks
    private native void nativeSurfaceInit(Object surface); // A new surface is available
    private native void nativeSurfaceFinalize(); // Surface about to be destroyed
    private long native_custom_data;      // Native code will use this to keep private data
```

그림 7 tutorial-5.c 의 함수 선언.

```
private String server  = "211.198.35.6";  //서버아이피
private String vport="5001";              //비디오 서버포트

private String host = "211.198.35.6";
private String port = "5004";             //소켓통신 서버포트
```

그림 8 서버의 외부아이피와 서버의 포트 설정

```
//--------------------GraphView
GraphView graph = (GraphView) findViewById(R.id.graph);
mSeries1 = new LineGraphSeries<>();
mSeries1.setThickness(10); //그래프 선두께
mSeries1.setDataPointsRadius(10);
mSeries1.setOnDataPointTapListener(new OnDataPointTapListener() {
    @Override
    public void onTap(Series series, DataPointInterface dataPoint) {
        Toast.makeText(getApplicationContext(), text: "Point clicked: "+dataPoint.getY(), Toast.LENGTH_SHORT).show();
    }
});
graph.addSeries(mSeries1);
graph.getViewport().setXAxisBoundsManual(true); //setminX setmaxX를 사용하기위해 메뉴얼로 바꿔줌.
graph.getViewport().setMinX(0);
graph.getViewport().setMaxX(200); //그래프 범위지정
graph.getGridLabelRenderer().setHorizontalLabelsVisible(false); //가로축 라벨가리기
graph.getLegendRenderer().setVisible(false); //필요없는 그리드 없애기
graph.getGridLabelRenderer().setGridStyle(GridLabelRenderer.GridStyle.HORIZONTAL); //그리드 가로만 켜기
graph.getGridLabelRenderer().setVerticalAxisTitle("Bpm"); //그래프 축 이름 설정

graph.getViewport().setScalable(true); // Scroll enable
graph.getViewport().setScalableY(true);
```

그림 9 그래프뷰 셋팅

```
Thread thread = new Thread(new Runnable() {
    @Override
    public void run() { //쿼드코어CPU처럼 메인쓰레드가아닌 다른쓰레드를 이용해서 소켓 데이터요청
        while(flag) {
            try {
                MyClientTask myClientTask = new MyClientTask(host, Integer.parseInt(port), message: "request");//request라는 소켓데이터를  라즈베리파이로 전달
                myClientTask.execute(); //객체의 함수를 실행
                handler.post(new Runnable() {
                    @Override
                    public void run() {

                    }
                });
                Thread.sleep( millis: 1000); //1초마다
            } catch (InterruptedException e) { //오류나면 멈춤
                break;

            }
        }
    }
});
thread.start();
```

그림 10 1초마다 서버로 request(TCP통신) 요청 보냄
이후 "request" 메시지를 받은 서버는 센서의 상태, BPM을 송신함.

```
mTimer1 = new Runnable() {
    @Override
    public void run() { //그래프 일정시간마다 출력하기
        //bpm_text.setText("BPM : "+BPM);
        graph2LastXValue += 1d;
        mSeries1.appendData(new DataPoint(graph2LastXValue,y), scrollToEnd: true,  maxDataPoints: 300);
        mHandler.postDelayed( r: this,  delayMillis: 200); //출력시간 interval
    }
};
mHandler.postDelayed(mTimer1,  delayMillis: 200);
```

그림 11 그래프뷰 200ms마다 BPM데이터인 y축을 업그레이드함.

```
nativeInit(server, vport); //실시간 영상 initialize
nativePlay(); //실시간 영상 play
```

그림 12 Gstreamer 서버와 포트입력으로
초기화 및 영상 플레이

```
static { //C라이브러리 추가
    System.loadLibrary( libname: "gstreamer_android");
    System.loadLibrary( libname: "tutorial-5");
    nativeClassInit();

}
```

그림 13 Tutorial-5.c 라이브러리 추가 및 클래스 초기화

```
//소켓통신 class
public class MyClientTask extends AsyncTask<Void, Void, Void> {
    String dstAddress; //ipadress
    int dstPort; //포트명
    String response = ""; //받는메세지
    String myMessage = ""; //송신메세지

    //constructor
    MyClientTask(String addr, int port, String message){
        dstAddress = addr;//ipadress
        dstPort = port; //포트명
        myMessage = message; //메세지
    }
```

```
    @Override
    protected Void doInBackground(Void... arg0) {

        Socket socket = null;
        myMessage = myMessage.toString();
        try {
            socket = new Socket(dstAddress, dstPort);
            //송신
            OutputStream out = socket.getOutputStream(); //송신측 설정
            out.write(myMessage.getBytes()); //송신메세지 사이즈와 함께전달

            //수신
            ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream( size: 1024); //수신측 1024사이즈
            byte[] buffer = new byte[1024]; //받을데이터 사이즈 선택하고
            int bytesRead;
            InputStream inputStream = socket.getInputStream(); //소켓에서 받은데이터를 inputStream에넣음

            while ((bytesRead = inputStream.read(buffer)) != -1){
                byteArrayOutputStream.write(buffer,  off: 0, bytesRead);
                response += byteArrayOutputStream.toString( charsetName: "UTF-8"); //UTF-8으로 컨버트
            }
            response =response; //받은데이터

        } catch (UnknownHostException e) { //소켓통신오류
```

```
            e.printStackTrace();
            flag=false;
            response = "UnknownHostException: " + e.toString();
        } catch (IOException e) {
            e.printStackTrace();

            flag=false;
            response = "IOException: " + e.toString();
        }finally{
            if(socket != null){
                try {
                    socket.close();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    flag=false;
                }
```

```
                ,
            }
        }
        return null;
    }

    @Override
    protected void onPostExecute(Void result) { //소켓 수신받으면 발생
        try{
            ImageView image_temp;
            TextView text_temp;
            if(orientation_vertical){
                image_temp = imageView;
                text_temp = textview_status;
            }else{
                image_temp = imageView_hor;
                text_temp = textView_hor;
            }
            switch(response){
                case "none": // receive == none
                    image_temp.setImageResource(R.drawable.red);
                    text_temp.setText("Sensor Undetected");
                    y=0;
                    break;
                case "cal": // receive == cal
                    image_temp.setImageResource(R.drawable.orange);
```

→ 소켓 통신 클래스를 Asynctask로 해야 한다. 아니면 메인에서는 소켓통신을 사용 할 수 없다. 받은 데이터를 통해 UI를 업그레이드하고 센서 상태에 따라 image를 red orange green으로 만들어준다.

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT){ // 세로 전환시{
        orientation_vertical = true;
        textView_hor.setVisibility(View.INVISIBLE);
        imageView_hor.setVisibility(View.INVISIBLE);
        relativeLayout.setLayoutParams(paramsNotFullscreen);

    }
    else if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {// 가로 전환시
        orientation_vertical=false;
        paramsNotFullscreen =new LinearLayout.LayoutParams(relativeLayout.getLayoutParams());
        textView_hor.setVisibility(View.VISIBLE);
        imageView_hor.setVisibility(View.VISIBLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);
        //paramsNotFullscreen=(RelativeLayout.LayoutParams)relativeLayout.getLayoutParams();
        LinearLayout.LayoutParams params = new LinearLayout.LayoutParams( relativeLayout.getLayoutParams());
        params.height=ViewGroup.LayoutParams.MATCH_PARENT;
        params.width=ViewGroup.LayoutParams.MATCH_PARENT;
        relativeLayout.setLayoutParams(params);


        }
}
```

그림 18 가로회전시 UI상태변환

→ 가로 회전 시 비디오뷰가 속한 relativeLayout을 휴대폰 화면에 꽉 차게 만들어주고 가로 회전 전(세로)에서의 상태를 변수에 저장해놨다가 세로 전환 시 원래대로 되돌려준다.

가로 회전 시 센서값과 BPM을 비디오화면 위에 표시하기 위해 VISIBLE로 상태 변환하여 표시한다. 세로일 때는 그래프뷰에 BPM이 표시되므로 비디오화면 위에 표시했던 textview와 imageview를 INVISIBLE상태로 바꾸어준다.

② Server

소켓 통신을 통해 생체로부터 하드웨어 장치가 받아온 심박도의 연결 상태를 확인할 수 있도록 개발한다.

```
def transfer_socket():
    global  bpmdata
    global count_
    count_ =0
    while True:
        conn, addr = sock_transfer.accept()
        data = conn.recv(1024)
        data = data.decode("utf-8").strip()
        res = bpmdata
        if res =='N':
            res = "none"
        elif res =='C':
            res = "cal"
        conn.sendall(res.encode("utf-8"))
        if count_ == 0 :
            count_ =1
            main_dialog.transfer_tB_set_text("TRANSFER : {} >> {}".format(res, addr))
        elif count_ ==1:
            count_=0
            main_dialog.transfer_tB_set_text("*TRANSFER : {} >> {}".format(res, addr))

        conn.close()
t2 = threading.Thread(target=transfer_socket)
t2.daemon = True
t2.start()
```

```
def read_socket():
    global bpmdata
    global count
    count=0
    while True:
        data, addr = sock_receive.recvfrom(1024)

        bpmdata = data.decode('utf-8')
        if count == 0:
            main_dialog.receive_tB_set_text("RECEIVED : {} << {}".format(data.decode('utf-8'), addr))
            count = 1
        elif count ==1:
            main_dialog.receive_tB_set_text("*RECEIVED : {} << {}".format(data.decode('utf-8'), addr))
            count = 0
        if bpmdata != 'N' and bpmdata != 'C':
            main_dialog.set_SEG(bpmdata)
            main_dialog.set_light("green")
        elif bpmdata.find('N') != -1 :
            main_dialog.set_light("red")
        elif bpmdata.find('C') != -1:
            main_dialog.set_light("orange")
```

2) Hardware

① PPG 센서

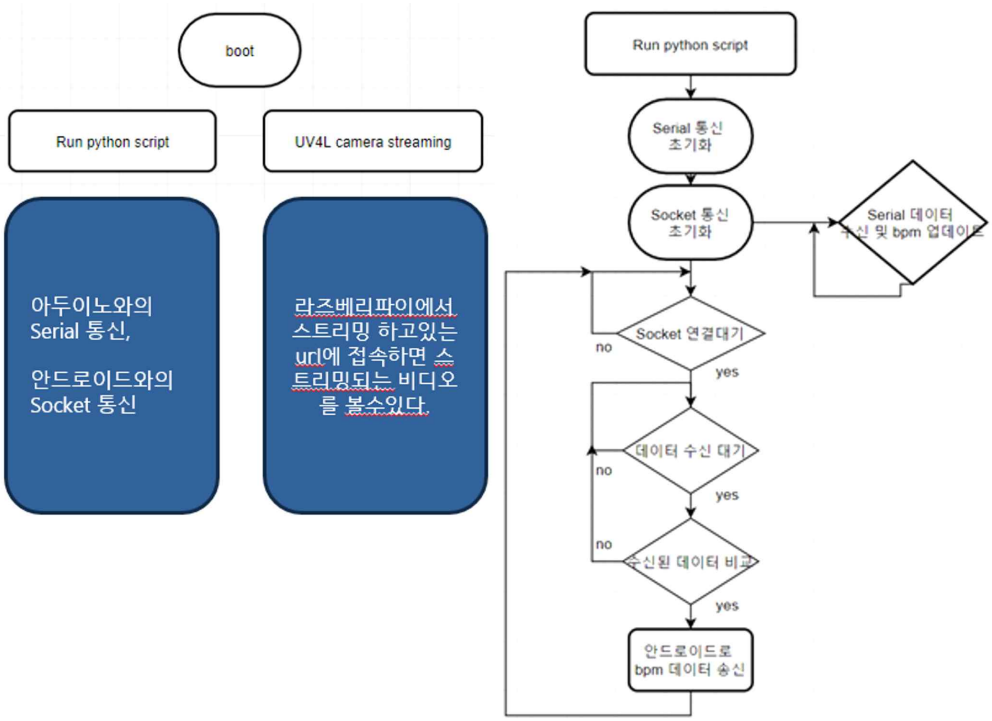
② ADC (Analog To Digital Converter)

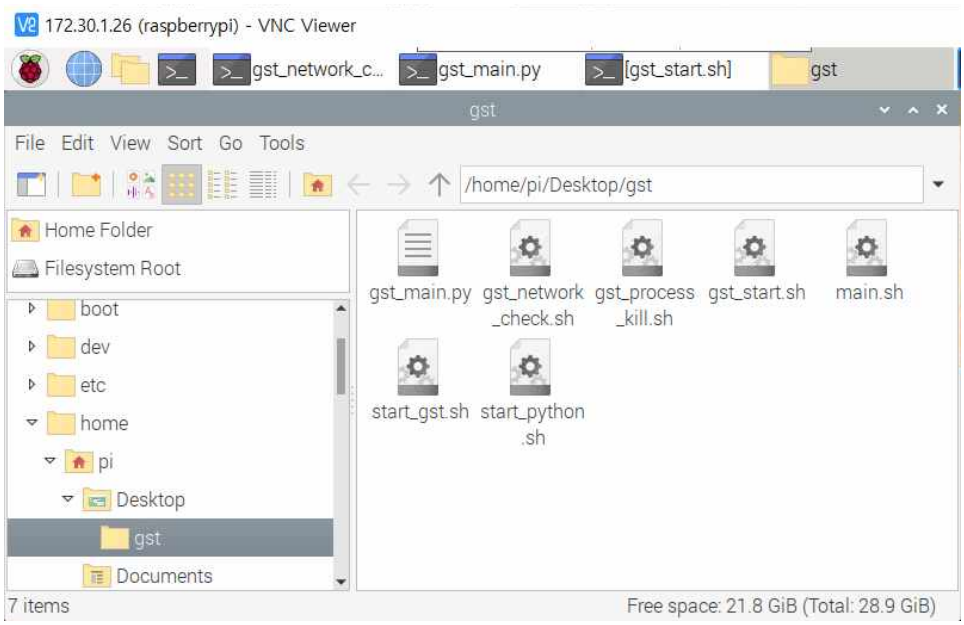
- 아두이노를 사용해서 비교기를 통해 들어온 심박신호의 텀을 계산하여 라즈베리파이로 시리얼통신을 통해 넘겨준다.

- 코드(.ino)

```
const int heart_rate_sensor = A0;
unsigned long duration;

void setup(){
Serial.begin(115200);    //라즈베리파이와이 시리얼 시간 맞추기
pinMode( heart_rate_sensor, INPUT_PULLUP);
}
void loop(){
 duration = pulseIn(heart_rate_sensor, HIGH);    //핀이 HIGH될때까지 기다림 심박 있을경우
float bpm_float = 60000/duration;    // bpm 계산 mm단위
int bpm = int(bpm_float);    //소수점 버림
Serial.println(bpm);    //Serial통신으로 라즈베리파이에 bpm 수치 전달
}
```

③ 라즈베리파이





- 아두이노와의 시리얼 통신

→ 시리얼로 받은 데이터를 디코딩해준다.(원하는 값만 추출)

```
def read_serial():
    global serial_data
    while True:
        read_ser=arduino.readline()
```

```
            serial_data =read_ser.decode()[:-2]

read_serial_thread = threading.Thread(target=read_serial)
read_serial_thread.start()
```

- 안드로이드와의 소켓 통신

→ 소켓통신이 메인 쓰레드에서 돌아가므로 아두이노에서 무작위로 들어오는 시리얼 통신된 데이터를 계속해서 받아오도록 서브쓰레딩으로 값을 받아온다. (메인과 서브쓰레드는 직렬이 아닌 병렬로 돌아가므로 가능)

```
def do_some_stuffs_with_input(input_string):
    global serial_data
    if input_string == "request":
        input_string= serial_data

    return input_string
while True:
    conn, addr = s.accept() #
    print("Connected by ", addr)
    data = conn.recv(1024)
    data = data.decode("utf8").strip()
    if not data: break
    print("Received: " + data)
    res = do_some_stuffs_with_input(data)
    print("transmit: "+ res)
    conn.sendall(res.encode("utf-8"))
    conn.close()
s.close()
```

5. Prototype 구현

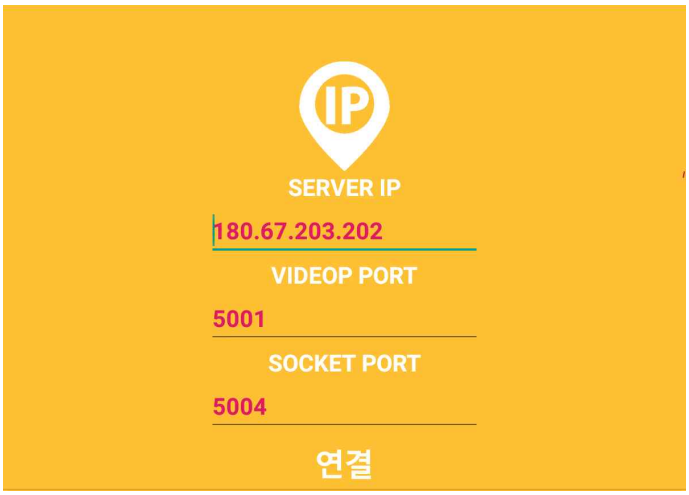
1) 구현 현황

① 문제점 및 해결방안

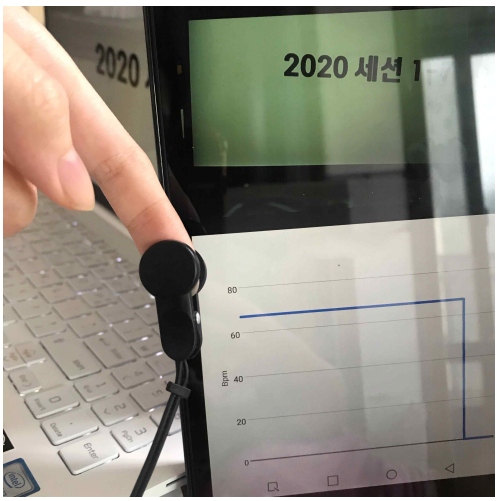
- 기존에는 ECG 센서를 이용하여 심전도 신호를 출력하려 했다. 하지만 제작 과정 중 ECG 신호 출력은 되지만 반려동물의 사소한 움직임에도 모션 아티팩트와 전극 특성상 신호가 매우 출렁거려 제대로 된 심장박동 피크를 잡는 것이 불가능하다는 결론을 내리게 되었다. 이러한 이유로 ECG 대신 PPG(광소자를 활용해 심장의 활동을 해석) 센서를 이용하였고 클립 형식으로 제작해 동물의 귀나 실험적으로 심장박동 측정이 잘 되는 곳에 부착하는 방식으로 변경하여 작품 제작을 진행하게 되었다.

② Application

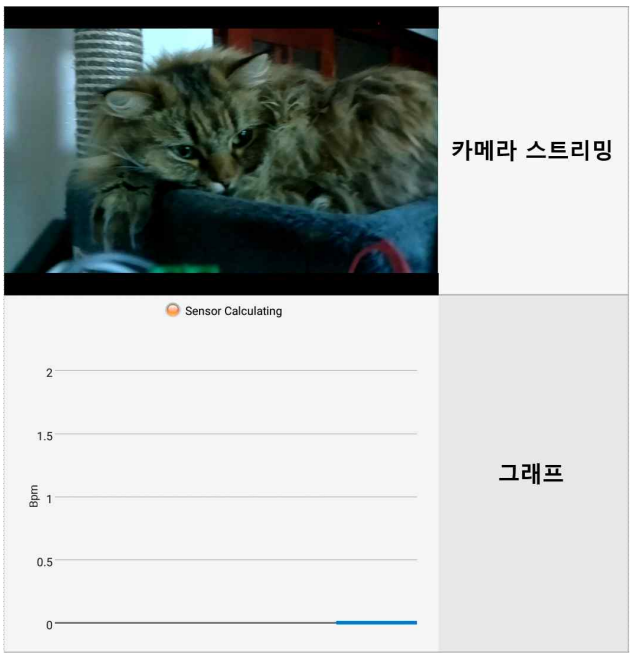
❶ 라즈베리파이의 전원을 켜고 현재 연결할 ip를 입력한다.

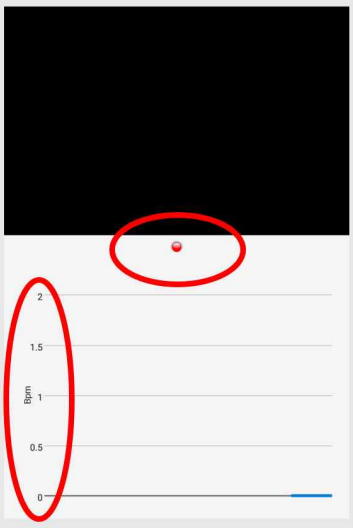
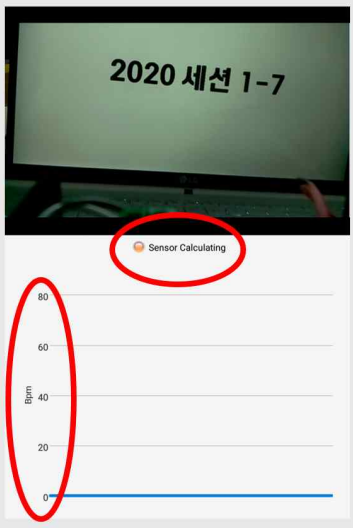
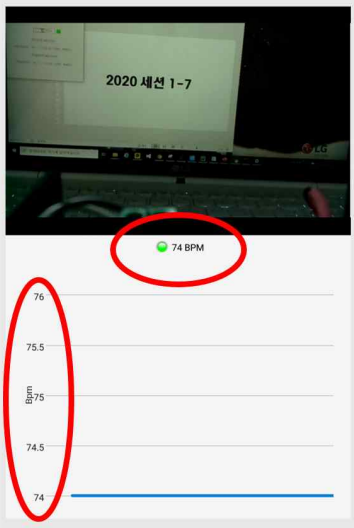


❷ 하드웨어의 클립형 PPG센서와 혈류가 흐르는 신체 부위를 연결시킨다.



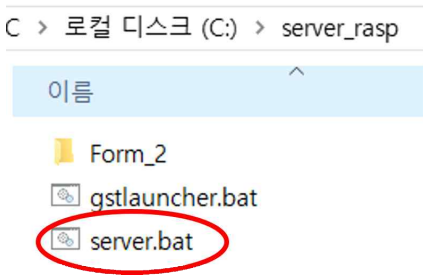
❸ 화면의 카메라 스트리밍 화면과 그래프를 통해 연결 상태를 확인한다.



연결 받지 않은 상태 (라즈베리파이 off 상태)	연결 중 (라즈베리파이 on 센서 연결)	연결 완료 후 BPM 연결 정상화
		

③ Server

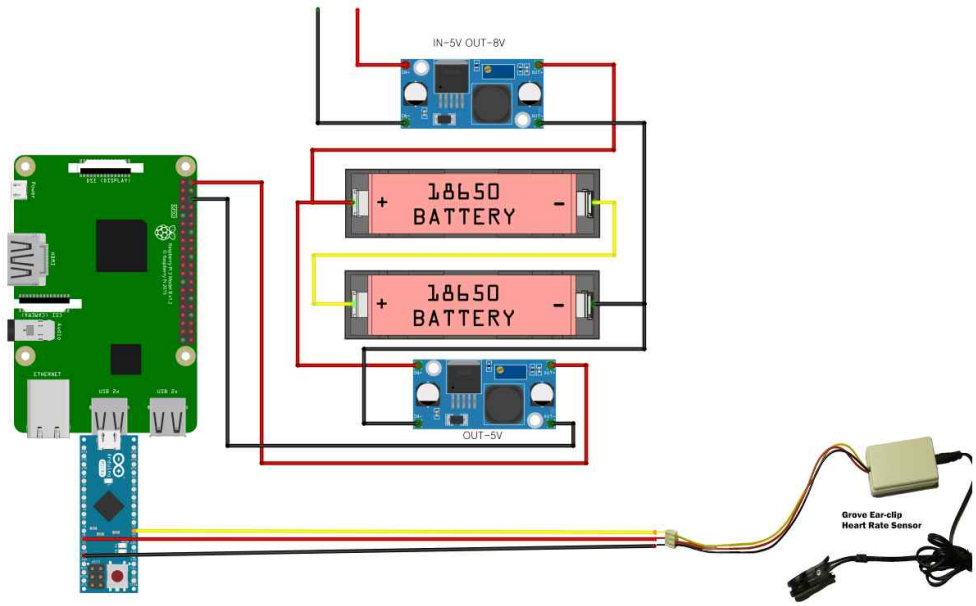
❶ 서버를 실행한다.



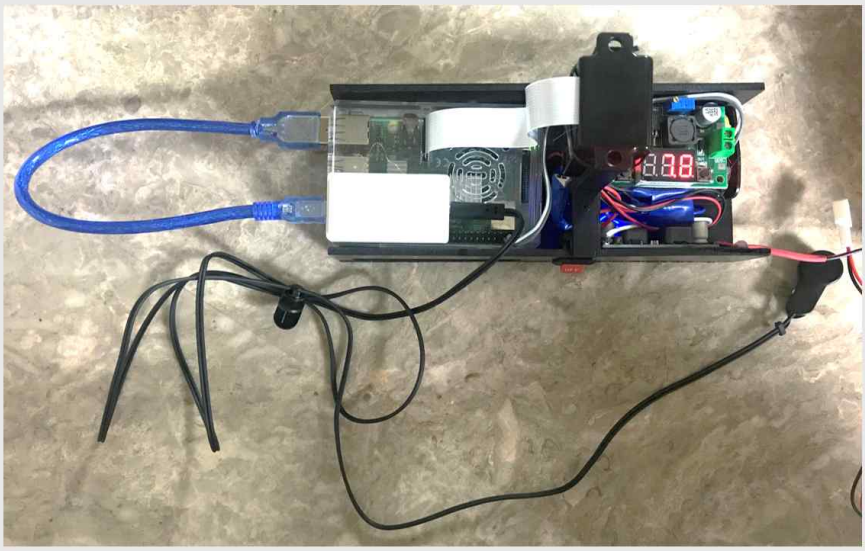
❷ 실행하면 나오는 창을 통해 연결이 원활히 이루어지고 있는지 확인한다.

BPM 0 (우측 사각형 붉은색)	BPM 받는 중 (우측 사각형 노란색)	BPM 받음 (우측 사각형 초록색)
		

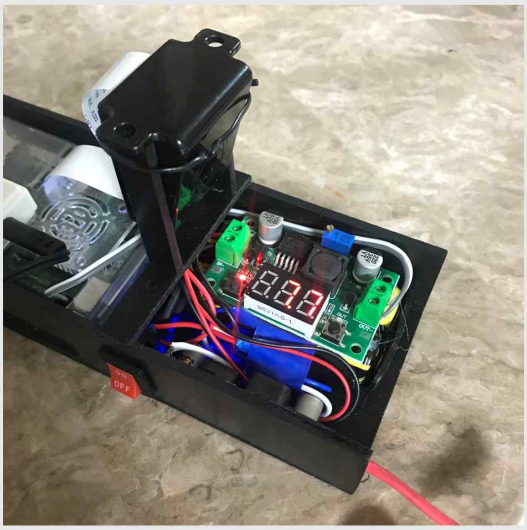
⑤ Hardware



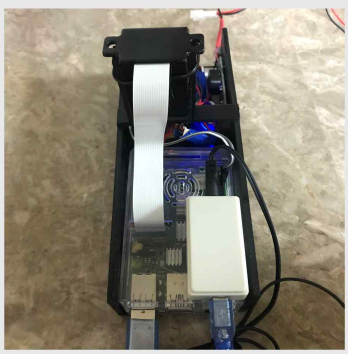
하드웨어 전체



카메라 모듈, 아두이노 나노, 배터리 & 전압 관리



라즈베리파이



6. 시험/ 테스트 결과

- 카메라 실시간 스트리밍은 문제없이 가능하였다.

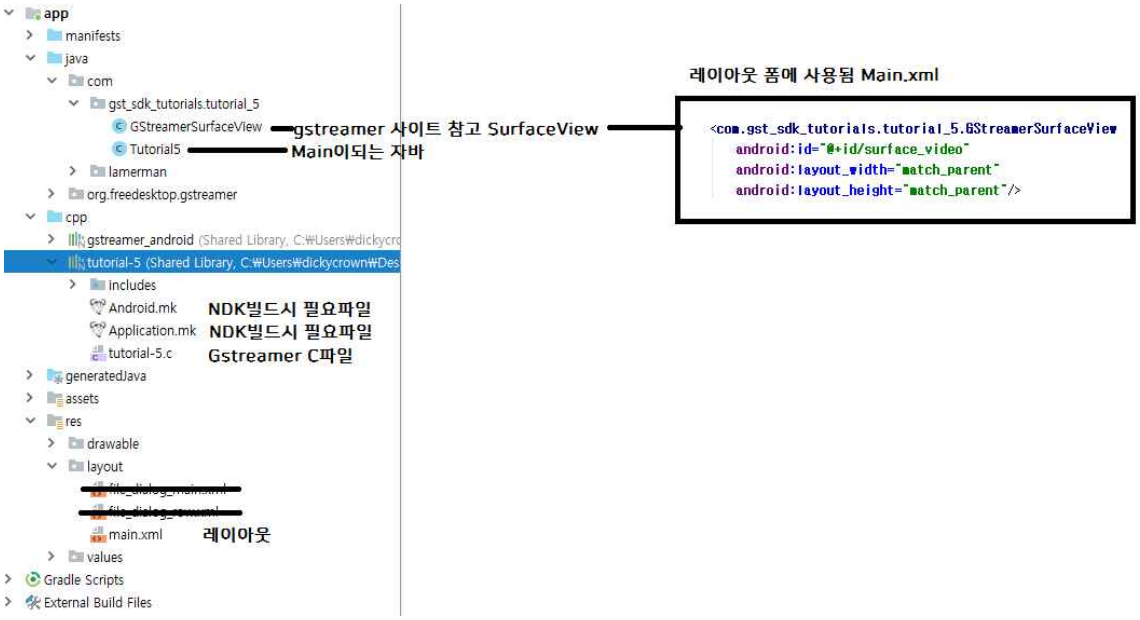
- ppg 센서를 이용한 심박수 측정 테스트는 사람의 손가락과 동물의 귀를 이용해 측정 진행하였다. 그래프와 숫자를 통한 데이터 전달(어플리케이션과 서버)은 문제없이 가능했다.

7. Coding & DEMO

1) Coding

① Application

- 전체구성



· GStreamerSurfaceView.java

```
package com.study.gst.gst_first_example;

import android.content.Context;
import android.util.AttributeSet;
import android.util.Log;
import android.view.SurfaceView;
import android.view.View;

// A simple SurfaceView whose width and height can be set from the outside
public class GStreamerSurfaceView extends SurfaceView {
    public int media_width = 320;
    public int media_height = 240;

    // Mandatory constructors, they do not do much
    public GStreamerSurfaceView(Context context, AttributeSet attrs,
                                int defStyle) {
        super(context, attrs, defStyle);
    }

    public GStreamerSurfaceView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public GStreamerSurfaceView (Context context) {
        super(context);
    }

    // Called by the layout manager to find out our size and give us some ru
les.
    // We will try to maximize our size, and preserve the media's aspect rati
o if
    // we are given the freedom to do so.
```

```
    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int width = 0, height = 0;
        int wmode = View.MeasureSpec.getMode(widthMeasureSpec);
        int hmode = View.MeasureSpec.getMode(heightMeasureSpec);
        int wsize = View.MeasureSpec.getSize(widthMeasureSpec);
        int hsize = View.MeasureSpec.getSize(heightMeasureSpec);

        Log.i ("GStreamer", "onMeasure called with " + media_width + "x" + m
edia_height);
        // Obey width rules
        switch (wmode) {
            case View.MeasureSpec.AT_MOST:
                if (hmode == View.MeasureSpec.EXACTLY) {
                    width = Math.min(hsize * media_width / media_height, ws
ize);
                    break;
                }
            case View.MeasureSpec.EXACTLY:
                width = wsize;
                break;
            case View.MeasureSpec.UNSPECIFIED:
                width = media_width;
        }

        // Obey height rules
        switch (hmode) {
            case View.MeasureSpec.AT_MOST:
                if (wmode == View.MeasureSpec.EXACTLY) {
                    height = Math.min(wsize * media_height / media_width, h
size);
                    break;
                }
```

```
            case View.MeasureSpec.EXACTLY:
                height = hsize;
                break;
            case View.MeasureSpec.UNSPECIFIED:
                height = media_height;
        }

        // Finally, calculate best size when both axis are free
        if (hmode == View.MeasureSpec.AT_MOST && wmode == View.Measur
eSpec.AT_MOST) {
            int correct_height = width * media_height / media_width;
            int correct_width = height * media_width / media_height;

            if (correct_height < height)
                height = correct_height;
            else
                width = correct_width;
        }

        // Obey minimum size
        width = Math.max (getSuggestedMinimumWidth(), width);
        height = Math.max (getSuggestedMinimumHeight(), height);
        setMeasuredDimension(width, height);
    }

}
```

· Tutorial5.java

```
package com.gst_sdk_tutorials.tutorial_5;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.net.UnknownHostException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;

import android.annotation.SuppressLint;
import android.annotation.TargetApi;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.res.Configuration;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.PowerManager;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.view.Window;
```

```
import android.view.WindowManager;
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.TextView;
import android.widget.Toast;


import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.GridLabelRenderer;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.DataPointInterface;
import com.jjoe64.graphview.series.LineGraphSeries;
import com.jjoe64.graphview.series.OnDataPointTapListener;
import com.jjoe64.graphview.series.Series;
import org.freedesktop.gstreamer.GStreamer;

public class Tutorial5 extends Activity implements SurfaceHolder.Callback {
    private native void nativeInit(String addr, String vport);      // Initialize nati
ve code, build pipeline, etc
    private native void nativeFinalize(); // Destroy pipeline and shutdown nativ
e code
    private native void nativeSetUri(String uri); // Set the URI of the media to
play
    private native void nativePlay();       // Set pipeline to PLAYING
    private native void nativePause();      // Set pipeline to PAUSED
    private static native boolean nativeClassInit(); // Initialize native class: cac
he Method IDs for callbacks
```

```
    private native void nativeSurfaceInit(Object surface); // A new surface is a
vailable
    private native void nativeSurfaceFinalize(); // Surface about to be destroye
d
    private long native_custom_data;      // Native code will use this to keep
private data


    private boolean is_playing_desired;   // Whether the user asked to go to
PLAYING
    private int position;                 // Current position, reported by native
code
    private int duration;                 // Current clip duration, reported by
native code
    private boolean is_local_media;       // Whether this clip is stored locally
or is being streamed
    private int desired_position;         // Position where the users wants to
seek to
    private String mediaUri;              // URI of the clip being played

    private final String defaultMediaUri = "http://docs.gstreamer.com/media/sin
tel_trailer-368p.ogv";


    private EditText editTextIPAddress;
    private InputMethodManager imm;
    private String server  = "211.198.35.6";  //서버아이피
    private String vport="5001";              //비디오 서버포트

    private String host = "211.198.35.6";
    private String port = "5004";             //소켓통신 서버포트
    private Socket socket = null;
    Handler handler = new Handler();
```

```
private final Handler mHandler = new Handler();
private Runnable mTimer1;
private LineGraphSeries<DataPoint> mSeries1;
private double graph2LastXValue = 5d;
public int y=0;
private OutputStream outs;


//-------------UI
LinearLayout.LayoutParams paramsNotFullscreen;
ImageView imageView,imageView_hor;
TextView textview_status,textView_hor;
SurfaceView sv;
LinearLayout relativeLayout;
boolean flag = true;
boolean orientation_vertical = true;
// Called when the activity is first created.
@TargetApi(Build.VERSION_CODES.HONEYCOMB)
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);

    // Initialize GStreamer and warn if it fails
    try {
        GStreamer.init(this);
    } catch (Exception e) {
        Toast.makeText(this, e.getMessage(), Toast.LENGTH_LONG).show();
        finish();
        return;
    }
    setContentView(R.layout.main);
```

```
        //-----------------------UI
        textview_status=findViewById(R.id.textView_status);
        textView_hor= findViewById(R.id.textView_hor);
        imageView_hor=findViewById(R.id.imageView_hor);
        imageView=findViewById(R.id.imageView);
        relativeLayout = findViewById(R.id.relativelayout);
        sv = (SurfaceView) this.findViewById(R.id.surface_video);
        SurfaceHolder sh = sv.getHolder();
        sh.addCallback(this);

        //--------------------GraphView
        GraphView graph = (GraphView) findViewById(R.id.graph);
        mSeries1 = new LineGraphSeries<>();
        mSeries1.setThickness(10); //그래프 선두께
        mSeries1.setDataPointsRadius(10);
        mSeries1.setOnDataPointTapListener(new OnDataPointTapListener() {
            @Override
            public void onTap(Series series, DataPointInterface dataPoint) {
                Toast.makeText(getApplicationContext(), "Point clicked: "+dataP
oint.getY(), Toast.LENGTH_SHORT).show();
            }
        });
        graph.addSeries(mSeries1);
        graph.getViewport().setXAxisBoundsManual(true);
        graph.getViewport().setMinX(0);
        graph.getViewport().setMaxX(200); //그래프 범위지정
        graph.getGridLabelRenderer().setHorizontalLabelsVisible(false);   //가로축
라벨가리기
        graph.getLegendRenderer().setVisible(false); //필요없는 그리드 없애기
        graph.getGridLabelRenderer().setGridStyle(GridLabelRenderer.GridStyle.H
ORIZONTAL); //그리드 가로만 켜기
        graph.getGridLabelRenderer().setVerticalAxisTitle("Bpm"); //그래프 축 이
름 설정
```

```
        graph.getViewport().setScalable(true); // Scroll enable
        graph.getViewport().setScalableY(true);


        Thread thread = new Thread(new Runnable() {
            @Override
            public void run() { //쿼드코어CPU처럼 메인쓰레드가아닌 다른쓰레드를
이용해서 소켓 데이터요청
                while(flag) {
                    try {
                        MyClientTask myClientTask = new MyClientTask(host, I
nteger.parseInt(port), "request");//request라는 소켓데이터를  라즈베리파이로 전달
                        myClientTask.execute(); //객체의 함수를 실행
                        handler.post(new Runnable() {
                            @Override
                            public void run() {

                            }
                        });
                        Thread.sleep(1000); //1초마다
                    } catch (InterruptedException e) { //오류나면 멈춤
                        break;

                    }
                }
            }
        });
        thread.start();

        mTimer1 = new Runnable() {
            @Override
            public void run() { //그래프 일정시간마다 출력하기
```

```
                //bpm_text.setText("BPM : "+BPM);
                graph2LastXValue += 1d;
                mSeries1.appendData(new DataPoint(graph2LastXValue,y), true,
300);
                mHandler.postDelayed(this, 200); //출력시간 interval
            }
        };
        mHandler.postDelayed(mTimer1, 200);




        nativeInit(server, vport); //실시간 영상 initialize
        nativePlay(); //실시간 영상 play
    }
    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT){ //
세로 전환시{
            orientation_vertical = true;
            textView_hor.setVisibility(View.INVISIBLE);
            imageView_hor.setVisibility(View.INVISIBLE);
            relativeLayout.setLayoutParams(paramsNotFullscreen);

        }
        else if (newConfig.orientation == Configuration.ORIENTATION_LANDSCA
PE) {// 가로 전환시
            orientation_vertical=false;
            paramsNotFullscreen =new LinearLayout.LayoutParams(relativeLay
out.getLayoutParams());
            textView_hor.setVisibility(View.VISIBLE);
```

```
            imageView_hor.setVisibility(View.VISIBLE);
            getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSC
REEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);
            //paramsNotFullscreen=(RelativeLayout.LayoutParams)relativeLayou
t.getLayoutParams();
            LinearLayout.LayoutParams params = new LinearLayout.LayoutPar
ams( relativeLayout.getLayoutParams());
            params.height=ViewGroup.LayoutParams.MATCH_PARENT;
            params.width=ViewGroup.LayoutParams.MATCH_PARENT;
            relativeLayout.setLayoutParams(params);



            }
    }
    protected void onSaveInstanceState (Bundle outState) {
        Log.d ("GStreamer", "Saving state, playing:" + is_playing_desired + " po
sition:" + position +
                " duration: " + duration + " uri: " + mediaUri);
        outState.putBoolean("playing", is_playing_desired);
        outState.putString("mediaUri", mediaUri);
    }

    protected void onDestroy() {
        nativeFinalize();
        super.onDestroy();
    }

    // Called from native code. This sets the content of the TextView from th
e UI thread.
    private void setMessage(final String message) {
//          final TextView tv = (TextView) this.findViewById(R.id.textview_messag
e);
        runOnUiThread (new Runnable() {
```

```
            public void run() {
            }
        });
    }


    // Called from native code. Native code calls this once it has created its pipeline and
    // the main loop is running, so it is ready to accept commands.
    private void onGStreamerInitialized () {
        Log.i ("GStreamer", "GStreamer initialized:");
        Log.i ("GStreamer", "  playing:" + is_playing_desired + " position:" + position + " uri: " + mediaUri);

        // Restore previous playing state
        //setMediaUri ();
        //nativeSetPosition (position);
        if (is_playing_desired) {
            nativePlay();
        } else {
            nativePause();
        }

        // Re-enable buttons, now that GStreamer is initialized
        final Activity activity = this;
        runOnUiThread(new Runnable() {
            public void run() {
                nativePlay();
            }
        });
    }
```

```
    static { //C라이브러리 추가
        System.loadLibrary("gstreamer_android");
        System.loadLibrary("tutorial-5");
        nativeClassInit();

    }

    public void surfaceChanged(SurfaceHolder holder, int format, int width,
                                    int height) {
        Log.d("GStreamer", "Surface changed to format " + format + " width "
                + width + " height " + height);
        nativeSurfaceInit (holder.getSurface());
    }

    public void surfaceCreated(SurfaceHolder holder) {
        Log.d("GStreamer", "Surface created: " + holder.getSurface());
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
        Log.d("GStreamer", "Surface destroyed");
        nativeSurfaceFinalize ();
    }

    // Called from native code when the size of the media changes or is firs
t detected.
    // Inform the video surface about the new size and recalculate the layou
t.
    private void onMediaSizeChanged (int width, int height) {
        Log.i ("GStreamer", "Media size changed to " + width + "x" + height);
        final GStreamerSurfaceView gsv = (GStreamerSurfaceView) this.findVie
wById(R.id.surface_video);
        gsv.media_width = width;
        gsv.media_height = height;
```

```
        runOnUiThread(new Runnable() {
            public void run() {
                gsv.requestLayout();
            }
        });
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent dat
a)
    {
        try {

            String sndOpkey = "[close]";
            outs.write(sndOpkey.getBytes("UTF-8"));
            outs.flush();
        } catch (IOException e) {

            e.printStackTrace();
        }
    }

    //소켓통신 class
    public class MyClientTask extends AsyncTask<Void, Void, Void> {
        String dstAddress; //ipadress
        int dstPort; //포트명
        String response = ""; //받는메세지
        String myMessage = ""; //송신메세지

        //constructor
        MyClientTask(String addr, int port, String message){
            dstAddress = addr;//ipadress
            dstPort = port; //포트명
```

```
        myMessage = message; //메세지
    }

    @Override
    protected Void doInBackground(Void... arg0) {

        Socket socket = null;
        myMessage = myMessage.toString();
        try {
            socket = new Socket(dstAddress, dstPort);
            //송신
            OutputStream out = socket.getOutputStream(); //송신측 설정
            out.write(myMessage.getBytes()); //송신메세지 사이즈와 함께전달

            //수신
            ByteArrayOutputStream byteArrayOutputStream = new ByteArr
ayOutputStream(1024); //수신측 1024사이즈
            byte[] buffer = new byte[1024]; //받을데이터 사이즈 선택하고
            int bytesRead;
            InputStream inputStream = socket.getInputStream(); //소켓에서
받은데이터를 inputStream에넣음

            while ((bytesRead = inputStream.read(buffer)) != -1){
                byteArrayOutputStream.write(buffer, 0, bytesRead);
                response += byteArrayOutputStream.toString("UTF-8"); //UT
F-8으로 컨버트
            }
            response =response; //받은데이터
        } catch (UnknownHostException e) { //소켓통신오류
            e.printStackTrace();
            flag=false;
            response = "UnknownHostException: " + e.toString();
        } catch (IOException e) {
```

```
            e.printStackTrace();
            flag=false;
            response = "IOException: " + e.toString();
        }finally{
            if(socket != null){
                try {
                    socket.close();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    flag=false;
                }
            }
        }
        return null;
    }
    @Override
    protected void onPostExecute(Void result) { //소켓 수신받으면 발생
        try{
            ImageView image_temp;
            TextView text_temp;
            if(orientation_vertical){
                image_temp = imageView;
                text_temp = textview_status;
            }else{
                image_temp = imageView_hor;
                text_temp = textView_hor;
            }
            switch(response){
                case "none": // receive == none
                    image_temp.setImageResource(R.drawable.red);
                    text_temp.setText("Sensor Undetected");
                    y=0;
```

```
                    break;
                case "cal": // receive == cal
                    image_temp.setImageResource(R.drawable.orange);
                    text_temp.setText("Sensor Calculating");

                    break;
                default:
                    try{ //receive == BPM data
                        int bpm = Integer.parseInt(response);
                        text_temp.setText(response + " BPM");
                        image_temp.setImageResource(R.drawable.green);
                        y=bpm;
                    }catch (NumberFormatException nfe){
                    }
                    break;
            }
        }catch ( Exception e){
        }
        super.onPostExecute(result);
    }
  }
}
```

· tutorial-5.c

```
#include <string.h>
#include <stdint.h>
#include <jni.h>
#include <android/log.h>
#include <android/native_window.h>
#include <android/native_window_jni.h>
#include <gst/gst.h>
#include <gst/video/video.h>
#include <gst/video/videooverlay.h>
#include <pthread.h>

GST_DEBUG_CATEGORY_STATIC (debug_category);
#define GST_CAT_DEFAULT debug_category

/*
 * These macros provide a way to store the native pointer to CustomData, w
hich might be 32 or 64 bits, into
 * a jlong, which is always 64 bits, without warnings.
 */
#if GLIB_SIZEOF_VOID_P == 8
# define GET_CUSTOM_DATA(env, thiz, fieldID) (CustomData *)(*env)->GetLong
Field (env, thiz, fieldID)
# define SET_CUSTOM_DATA(env, thiz, fieldID, data) (*env)->SetLongField (env,
thiz, fieldID, (jlong)data)
#else
# define GET_CUSTOM_DATA(env, thiz, fieldID) (CustomData *)(jint)(*env)->GetL
ongField (env, thiz, fieldID)
# define SET_CUSTOM_DATA(env, thiz, fieldID, data) (*env)->SetLongField (env,
thiz, fieldID, (jlong)(jint)data)
#endif

/* Do not allow seeks to be performed closer than this distance. It is visually
```

```
useless, and will probably
 * confuse some demuxers. */
#define SEEK_MIN_DELAY (500 * GST_MSECOND)

/* Structure to contain all our information, so we can pass it to callbacks */
typedef struct _CustomData {
    jobject app;                  /* Application instance, used to call its met
hods. A global reference is kept. */
    GstElement *pipeline;         /* The running pipeline */
    GstElement *video_sink; /* The video sink element which receives XOverla
y commands */
    GMainContext *context;        /* GLib context used to run the main loop
*/
    GMainLoop *main_loop;         /* GLib main loop */
    gboolean initialized;         /* To avoid informing the UI multiple times a
bout the initialization */
    ANativeWindow *native_window; /* The Android native window where vide
o will be rendered */
    GstState state;               /* Current pipeline state */
    GstState target_state;        /* Desired pipeline state, to be set once buf
fering is complete */
    gint64 duration;              /* Cached clip duration */
    gint64 desired_position;      /* Position to seek to, once the pipeline is r
unning */
    GstClockTime last_seek_time;  /* For seeking overflow prevention (throttli
ng) */
    gboolean is_live;             /* Live streams do not use buffering */
} CustomData;

gchar *pipeline;


/* playbin2 flags */
```

```
typedef enum {
    GST_PLAY_FLAG_TEXT = (1 << 2)  /* We want subtitle output */
} GstPlayFlags;

/* These global variables cache values which are not changing during executi
on */
static pthread_t gst_app_thread;
static pthread_key_t current_jni_env;
static JavaVM *java_vm;
static jfieldID custom_data_field_id;
static jmethodID set_message_method_id;
//static jmethodID set_current_position_method_id;
static jmethodID on_gstreamer_initialized_method_id;
static jmethodID on_media_size_changed_method_id;

/*
 * Private methods
 */

/* Register this thread with the VM */
static JNIEnv *attach_current_thread (void) {
    JNIEnv *env;
    JavaVMAttachArgs args;

    GST_DEBUG ("Attaching thread %p", g_thread_self ());
    args.version = JNI_VERSION_1_4;
    args.name = NULL;
    args.group = NULL;

    if ((*java_vm)->AttachCurrentThread (java_vm, &env, &args) < 0) {
        GST_ERROR ("Failed to attach current thread");
        return NULL;
    }
```

```
    return env;
}

/* Unregister this thread from the VM */
static void detach_current_thread (void *env) {
    GST_DEBUG ("Detaching thread %p", g_thread_self ());
    (*java_vm)->DetachCurrentThread (java_vm);
}

/* Retrieve the JNI environment for this thread */
static JNIEnv *get_jni_env (void) {
    JNIEnv *env;

    if ((env = pthread_getspecific (current_jni_env)) == NULL) {
        env = attach_current_thread ();
        pthread_setspecific (current_jni_env, env);
    }

    return env;
}

/* Change the content of the UI's TextView */
static void set_ui_message (const gchar *message, CustomData *data) {
    JNIEnv *env = get_jni_env ();
    GST_DEBUG ("Setting message to: %s", message);
    jstring jmessage = (*env)->NewStringUTF(env, message);
    (*env)->CallVoidMethod (env, data->app, set_message_method_id, jmessag
e);
    if ((*env)->ExceptionCheck (env)) {
        GST_ERROR ("Failed to call Java method");
        (*env)->ExceptionClear (env);
    }
```

```
    (*env)->DeleteLocalRef (env, jmessage);
}

/* Retrieve errors from the bus and show them on the UI */
static void error_cb (GstBus *bus, GstMessage *msg, CustomData *data) {
    GError *err;
    gchar *debug_info;
    gchar *message_string;

    gst_message_parse_error (msg, &err, &debug_info);
    message_string = g_strdup_printf ("Error received from element %s: %s",
GST_OBJECT_NAME (msg->src), err->message);
    g_clear_error (&err);
    g_free (debug_info);
    set_ui_message (message_string, data);
    g_free (message_string);
    data->target_state = GST_STATE_NULL;
    gst_element_set_state (data->pipeline, GST_STATE_NULL);
}

/* Called when the End Of the Stream is reached. Just move to the beginnin
g of the media and pause. */
static void eos_cb (GstBus *bus, GstMessage *msg, CustomData *data) {
    data->target_state = GST_STATE_PAUSED;
    data->is_live = (gst_element_set_state (data->pipeline, GST_STATE_PAUSE
D) == GST_STATE_CHANGE_NO_PREROLL);
    //execute_seek (0, data);
}

/* Called when buffering messages are received. We inform the UI about the
current buffering level and
 * keep the pipeline paused until 100% buffering is reached. At that point, se
t the desired state. */
```

```
static void buffering_cb (GstBus *bus, GstMessage *msg, CustomData *data) {
    gint percent;

    if (data->is_live)
        return;

    gst_message_parse_buffering (msg, &percent);
    if (percent < 100 && data->target_state >= GST_STATE_PAUSED) {
        gchar * message_string = g_strdup_printf ("Buffering %d%%", percen
t);
        gst_element_set_state (data->pipeline, GST_STATE_PAUSED);
        set_ui_message (message_string, data);
        g_free (message_string);
    } else if (data->target_state >= GST_STATE_PLAYING) {
        gst_element_set_state (data->pipeline, GST_STATE_PLAYING);
    } else if (data->target_state >= GST_STATE_PAUSED) {
        set_ui_message ("Buffering complete", data);
    }
}
//----------------------
/* Called when the clock is lost */
static void clock_lost_cb (GstBus *bus, GstMessage *msg, CustomData *data) {
    if (data->target_state >= GST_STATE_PLAYING) {
        gst_element_set_state (data->pipeline, GST_STATE_PAUSED);
        gst_element_set_state (data->pipeline, GST_STATE_PLAYING);
    }
}
/* Retrieve the video sink's Caps and tell the application about the media siz
e */
static void check_media_size (CustomData *data) {
    JNIEnv *env = get_jni_env ();
    //GstElement *video_sink;
    GstPad *video_sink_pad;
```

```
    GstCaps *caps;
    GstVideoInfo info;
    /* Retrieve the Caps at the entrance of the video sink */
    g_object_get (data->pipeline, "video-sink", &data->video_sink, NULL);
    video_sink_pad = gst_element_get_static_pad (data->video_sink, "sink");
    //caps = gst_pad_get_negotiated_caps (video_sink_pad);
    caps = gst_pad_get_current_caps (video_sink_pad);
    if (gst_video_info_from_caps(&info, caps)) {
        info.width = info.width * info.par_n / info.par_d;
        GST_DEBUG ("Media size is %dx%d, notifying application", info.width, i
nfo.height);
        (*env)->CallVoidMethod (env, data->app, on_media_size_changed_meth
od_id, (jint)info.width, (jint)info.height);
        if ((*env)->ExceptionCheck (env)) {
            GST_ERROR ("Failed to call Java method");
            (*env)->ExceptionClear (env);
        }
    }
    gst_caps_unref(caps);
    gst_object_unref (video_sink_pad);
    gst_object_unref(data->video_sink);
}

/* Notify UI about pipeline state changes */
static void state_changed_cb (GstBus *bus, GstMessage *msg, CustomData *d
ata) {
    GstState old_state, new_state, pending_state;
    gst_message_parse_state_changed (msg, &old_state, &new_state, &pendin
g_state);
    /* Only pay attention to messages coming from the pipeline, not its child
ren */
    if (GST_MESSAGE_SRC (msg) == GST_OBJECT (data->pipeline)) {
        data->state = new_state;
```

```
        gchar *message = g_strdup_printf("State changed to %s", gst_element
_state_get_name(new_state));
        set_ui_message(message, data);
        g_free (message);

        /* The Ready to Paused state change is particularly interesting: */
        if (old_state == GST_STATE_READY && new_state == GST_STATE_PAU
SED) {
            /* By now the sink already knows the media size */
            check_media_size(data);
        }
    }
}

/* Check if all conditions are met to report GStreamer as initialized.
* These conditions will change depending on the application */
static void check_initialization_complete (CustomData *data) {
    JNIEnv *env = get_jni_env ();
    if (!data->initialized && data->native_window && data->main_loop) {
        GST_DEBUG ("Initialization complete, notifying application. native_wind
ow:%p main_loop:%p", data->native_window, data->main_loop);
        gst_video_overlay_set_window_handle (GST_VIDEO_OVERLAY (data->vid
eo_sink), (guintptr)data->native_window);
        (*env)->CallVoidMethod (env, data->app, on_gstreamer_initialized_meth
od_id);
        if ((*env)->ExceptionCheck (env)) {
            GST_ERROR ("Failed to call Java method");
            (*env)->ExceptionClear (env);
        }
        data->initialized = TRUE;
    }
}
```

```
/* Main method for the native code. This is executed on its own thread. */
static void *app_function (void *userdata) {
    JavaVMAttachArgs args;
    GstBus *bus;
    CustomData *data = (CustomData *)userdata;
    GSource *bus_source;
    GError *error = NULL;
    guint flags;

    GST_DEBUG ("Creating pipeline in CustomData at %p", data);

    /* Create our own GLib Main Context and make it the default one */
    data->context = g_main_context_new ();
    g_main_context_push_thread_default(data->context);

    data->pipeline = gst_parse_launch(pipeline, &error);

    if (error) {
        gchar *message = g_strdup_printf("Unable to build pipeline: %s", error
->message);
        g_clear_error (&error);
        set_ui_message(message, data);
        g_free (message);
        return NULL;
    }

    /* Disable subtitles */
    g_object_get (data->pipeline, "flags", &flags, NULL);
    flags &= ~GST_PLAY_FLAG_TEXT;
    g_object_set (data->pipeline, "flags", flags, NULL);

    /* Set the pipeline to READY, so it can already accept a window handle, i
f we have one */
```

```
    data->target_state = GST_STATE_READY;
    gst_element_set_state(data->pipeline, GST_STATE_READY);

    data->video_sink = gst_bin_get_by_interface(GST_BIN(data->pipeline), GST_
TYPE_VIDEO_OVERLAY);


    /* Instruct the bus to emit signals for each received message, and conne
ct to the interesting signals */
    bus = gst_element_get_bus (data->pipeline);
    bus_source = gst_bus_create_watch (bus);
    g_source_set_callback (bus_source, (GSourceFunc) gst_bus_async_signal_fu
nc, NULL, NULL);
    g_source_attach (bus_source, data->context);
    g_source_unref (bus_source);
    g_signal_connect (G_OBJECT (bus), "message::error", (GCallback)error_cb, d
ata);
    g_signal_connect (G_OBJECT (bus), "message::eos", (GCallback)eos_cb, dat
a);
    g_signal_connect (G_OBJECT (bus), "message::state-changed", (GCallback)st
ate_changed_cb, data);
    //g_signal_connect (G_OBJECT (bus), "message::duration", (GCallback)durati
on_cb, data);
    g_signal_connect (G_OBJECT (bus), "message::buffering", (GCallback)bufferi
ng_cb, data);
    g_signal_connect (G_OBJECT (bus), "message::clock-lost", (GCallback)clock_l
ost_cb, data);
    gst_object_unref (bus);

    /* Create a GLib Main Loop and set it to run */
    GST_DEBUG ("Entering main loop... (CustomData:%p)", data);
    data->main_loop = g_main_loop_new (data->context, FALSE);
    check_initialization_complete (data);
```

```
    g_main_loop_run (data->main_loop);
    GST_DEBUG ("Exited main loop");
    g_main_loop_unref (data->main_loop);
    data->main_loop = NULL;

    /* Free resources */
    g_main_context_pop_thread_default(data->context);
    g_main_context_unref (data->context);
    data->target_state = GST_STATE_NULL;
    gst_element_set_state (data->pipeline, GST_STATE_NULL);
    gst_object_unref (data->video_sink);
    gst_object_unref (data->pipeline);

    return NULL;
}
//--------------------------------------
/*
 * Java Bindings
 */

/* Instruct the native code to create its internal data structure, pipeline and t
hread */
static void gst_native_init (JNIEnv* env, jobject thiz, jstring addr, jstring strpor
t) {
    CustomData *data = g_new0 (CustomData, 1);
    SET_CUSTOM_DATA (env, thiz, custom_data_field_id, data);
    const gchar *char_uri1 = (gchar*)(*env)->GetStringUTFChars (env, addr, N
ULL);
    const gchar *char_uri2 = (gchar*)(*env)->GetStringUTFChars (env, strport,
NULL);
    pipeline=g_strjoin ("","tcpclientsrc host=", char_uri1, " port=", char_uri2, " !
gdpdepay ! rtph264depay ! avdec_h264 ! videoconvert ! autovideosink sync=fal
se",NULL);
```

```
    (*env)->ReleaseStringUTFChars (env, addr, char_uri1);
    (*env)->ReleaseStringUTFChars (env, strport, char_uri2);
    GST_DEBUG_CATEGORY_INIT (debug_category, "tutorial-5", 0, "Android tuto
rial 5");
    gst_debug_set_threshold_for_name("tutorial-5", GST_LEVEL_DEBUG);
    GST_DEBUG ("Created CustomData at %p", data);
    data->app = (*env)->NewGlobalRef (env, thiz);
    GST_DEBUG ("Created GlobalRef for app object at %p", data->app);
    pthread_create (&gst_app_thread, NULL, &app_function, data);
}



/* Quit the main loop, remove the native thread and free resources */
static void gst_native_finalize (JNIEnv* env, jobject thiz) {
    CustomData *data = GET_CUSTOM_DATA (env, thiz, custom_data_field_id);
    if (!data) return;
    GST_DEBUG ("Quitting main loop...");
    g_main_loop_quit (data->main_loop);
    GST_DEBUG ("Waiting for thread to finish...");
    pthread_join (gst_app_thread, NULL);
    GST_DEBUG ("Deleting GlobalRef for app object at %p", data->app);
    (*env)->DeleteGlobalRef (env, data->app);
    GST_DEBUG ("Freeing CustomData at %p", data);
    g_free (data);
    SET_CUSTOM_DATA (env, thiz, custom_data_field_id, NULL);
    GST_DEBUG ("Done finalizing");
}

/* Set playbin2's URI */
void gst_native_set_uri (JNIEnv* env, jobject thiz, jstring uri) {
    CustomData *data = GET_CUSTOM_DATA (env, thiz, custom_data_field_id);
```

```
    if (!data || !data->pipeline) return;
    const jbyte *char_uri = (*env)->GetStringUTFChars (env, uri, NULL);
    GST_DEBUG ("Setting URI to %s", char_uri);
    if (data->target_state >= GST_STATE_READY)
        gst_element_set_state (data->pipeline, GST_STATE_READY);
    g_object_set(data->pipeline, "uri", char_uri, NULL);
    (*env)->ReleaseStringUTFChars (env, uri, char_uri);
    data->duration = GST_CLOCK_TIME_NONE;
    data->is_live = (gst_element_set_state (data->pipeline, data->target_state)
== GST_STATE_CHANGE_NO_PREROLL);
}

/* Set pipeline to PLAYING state */
static void gst_native_play (JNIEnv* env, jobject thiz) {
    CustomData *data = GET_CUSTOM_DATA (env, thiz, custom_data_field_id);
    if (!data) return;
    GST_DEBUG ("Setting state to PLAYING");
    data->target_state = GST_STATE_PLAYING;
    data->is_live = (gst_element_set_state (data->pipeline, GST_STATE_PLAYIN
G) == GST_STATE_CHANGE_NO_PREROLL);
}

/* Set pipeline to PAUSED state */
static void gst_native_pause (JNIEnv* env, jobject thiz) {
    CustomData *data = GET_CUSTOM_DATA (env, thiz, custom_data_field_id);
    if (!data) return;
    GST_DEBUG ("Setting state to PAUSED");
    data->target_state = GST_STATE_PAUSED;
    data->is_live = (gst_element_set_state (data->pipeline, GST_STATE_PAUSE
D) == GST_STATE_CHANGE_NO_PREROLL);
}
```

```
/* Static class initializer: retrieve method and field IDs */
static jboolean gst_native_class_init (JNIEnv* env, jclass klass) {
    custom_data_field_id = (*env)->GetFieldID (env, klass, "native_custom_data
", "J");
    set_message_method_id = (*env)->GetMethodID (env, klass, "setMessage",
"(Ljava/lang/String;)V");
    //set_current_position_method_id = (*env)->GetMethodID (env, klass, "setC
urrentPosition", "(II)V");
    on_gstreamer_initialized_method_id = (*env)->GetMethodID (env, klass, "on
GStreamerInitialized", "()V");
    on_media_size_changed_method_id = (*env)->GetMethodID (env, klass, "on
MediaSizeChanged", "(II)V");

    if (!custom_data_field_id || !set_message_method_id || !on_gstreamer_initi
alized_method_id ||
        !on_media_size_changed_method_id ) {
        /* We emit this message through the Android log instead of the GStr
eamer log because the later
         * has not been initialized yet.
         */
        __android_log_print (ANDROID_LOG_ERROR, "tutorial-5", "The calling cl
ass does not implement all necessary interface methods");
        return JNI_FALSE;
    }
    return JNI_TRUE;
}

static void gst_native_surface_init (JNIEnv *env, jobject thiz, jobject surface) {
    CustomData *data = GET_CUSTOM_DATA (env, thiz, custom_data_field_id);
    if (!data) return;
    ANativeWindow *new_native_window = ANativeWindow_fromSurface(env, s
urface);
    GST_DEBUG ("Received surface %p (native window %p)", surface, new_nati
```

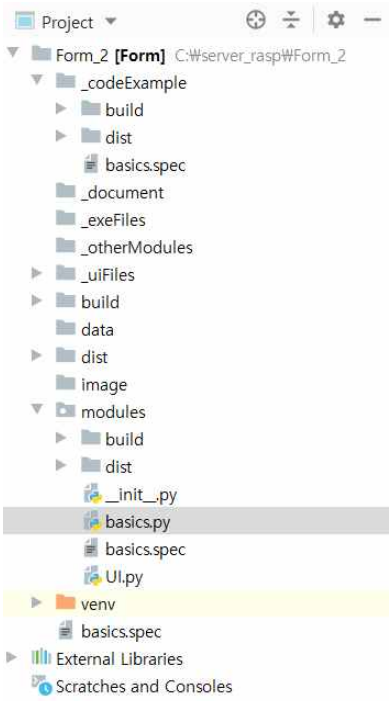
```
ve_window);
    //gst_video_overlay_prepare_window_handle (GST_VIDEO_OVERLAY (data->v
ideo_sink));
    if (data->native_window) {
        ANativeWindow_release (data->native_window);
        if (data->native_window == new_native_window) {
            GST_DEBUG ("New native window is the same as the previous on
e : %p", data->native_window);
            if (data->video_sink) {
                //gst_video_overlay_expose(GST_VIDEO_OVERLAY (data->pipelin
e));
                //gst_video_overlay_expose(GST_VIDEO_OVERLAY (data->pipelin
e));
                gst_video_overlay_expose(GST_VIDEO_OVERLAY (data->video_si
nk));
                gst_video_overlay_expose(GST_VIDEO_OVERLAY (data->video_si
nk));
            }
            return;
        } else {
            GST_DEBUG ("Released previous native window %p", data->native_
window);
            data->initialized = FALSE;
        }
    }
    data->native_window = new_native_window;
    check_initialization_complete (data);
}


static void gst_native_surface_finalize (JNIEnv *env, jobject thiz) {
    CustomData *data = GET_CUSTOM_DATA (env, thiz, custom_data_field_id);
    if (!data) return;
```

```
    GST_DEBUG ("Releasing Native Window %p", data->native_window);
    if (data->video_sink) {
        //gst_video_overlay_set_window_handle (GST_VIDEO_OVERLAY (data->pi
peline), (guintptr)NULL);
        //gst_video_overlay_prepare_window_handle (GST_VIDEO_OVERLAY (dat
a->video_sink));
        gst_video_overlay_set_window_handle (GST_VIDEO_OVERLAY (data->vid
eo_sink), (guintptr)NULL);
        gst_element_set_state (data->pipeline, GST_STATE_READY);
    }
    ANativeWindow_release (data->native_window);
    data->native_window = NULL;
    data->initialized = FALSE;
}

/* List of implemented native methods */
static JNINativeMethod native_methods[] = {
        { "nativeInit", "(Ljava/lang/String;Ljava/lang/String;)V", (void *) gst_nativ
e_init},
        { "nativeFinalize", "()V", (void *) gst_native_finalize},
        { "nativeSetUri", "(Ljava/lang/String;)V", (void *) gst_native_set_uri},
        { "nativePlay", "()V", (void *) gst_native_play},
        { "nativePause", "()V", (void *) gst_native_pause},
        { "nativeSurfaceInit", "(Ljava/lang/Object;)V", (void *) gst_native_surface
_init},
        { "nativeSurfaceFinalize", "()V", (void *) gst_native_surface_finalize},
        { "nativeClassInit", "()Z", (void *) gst_native_class_init}
};

jint JNI_OnLoad(JavaVM *vm, void *reserved) {
    JNIEnv *env = NULL;
    java_vm = vm;
    if ((*vm)->GetEnv(vm, (void**) &env, JNI_VERSION_1_4) != JNI_OK) {
```

```
        __android_log_print (ANDROID_LOG_ERROR, "tutorial-5", "Could not ret
rieve JNIEnv");
        return 0;
    }
    jclass klass = (*env)->FindClass (env, "com/study/gst/gst_first_example/Mai
nActivity");
    (*env)->RegisterNatives (env, klass, native_methods, G_N_ELEMENTS(native
_methods));
    pthread_key_create (&current_jni_env, detach_current_thread);
    return JNI_VERSION_1_4;
}
```

② Server

- 전체 구성



· **basics.py**

```
import sys, UI
import os
import ctypes
import subprocess, threading
import re
import PyQt5
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
import socket
import time

video_start_cmd = ('gst-launch-1.0  -v udpsrc port=5000 ! application/x-rtp ! rtph264depay ! rtph264pay config-interval=1 pt=96 !gdppay  ! tcpserversink host=172.30.1.18 port=5001')
video_stop_cmd = ('taskkill /IM "gst-launch-1.0.exe" /F')
```

```
bpmdata =''

class MainDialog(QDialog, UI.Ui_Dialog):
    def __init__(self):
        global t
        QDialog.__init__(self, None)
        self.setupUi(self)

        self.pushButton2.clicked.connect(self.stop_button_clicked)

    def set_SEG(self,string):
        inteager = int(string)
        self.lcdNumber.display(inteager)
    def set_light(self,color):
        stylesheet =''
        if color == "red":
            stylesheet = "background-color:#FF0000"
        elif color == "orange":
            stylesheet = "background-color:#FFA500"
        elif color == "green":
            stylesheet = "background-color:#00FF00"
        self.label_3.setStyleSheet(stylesheet)
    def receive_tB_set_text(self,text):
        self.label_5.setText(text)
        #self.textBrowser_2.setLineWrapMode(0)

    def transfer_tB_set_text(self,text):
        self.label_6.setText(text)

    def cmd_command(self):
        os.system(video_start_cmd)
    def stop_button_clicked(self):
```

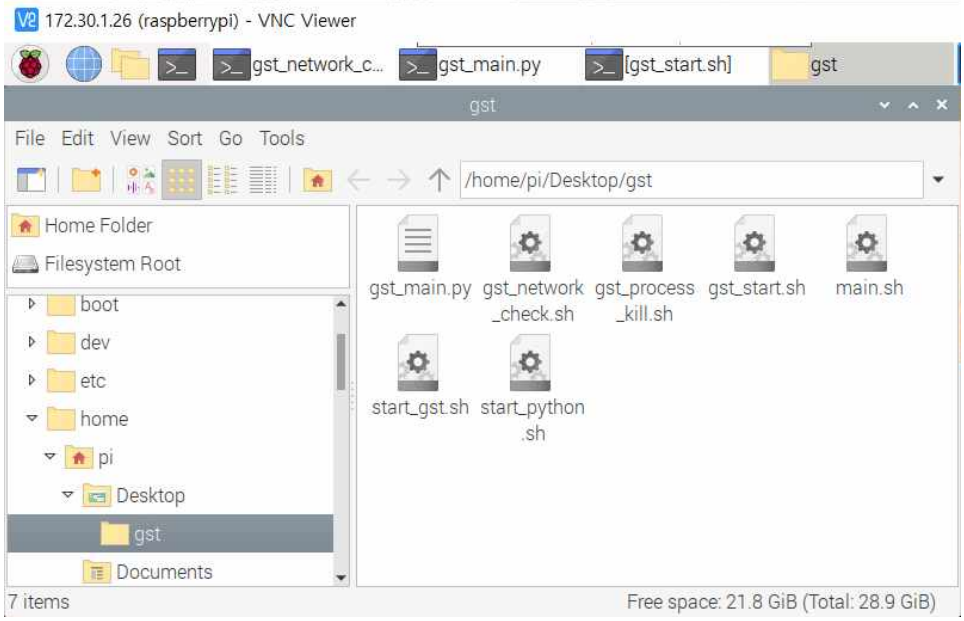
```
            os.system('taskkill /IM "gst-launch-1.0.exe" /F')
        def closeEvent(self, QCloseEvent):
            os.system(video_stop_cmd)
            t.stop()

app = QApplication(sys.argv)
main_dialog = MainDialog()

sock_receive = socket.socket(socket.AF_INET , socket.SOCK_DGRAM)
sock_receive.bind(('',5003))
main_dialog.receive_tB_set_text("Receive socket, port 5003 is opened")
sock_transfer = socket.socket()
sock_receive.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,1)
sock_transfer.bind(('',5004))
main_dialog.transfer_tB_set_text("Transfer socket, port 5004 is opened")
sock_transfer.listen()

time.sleep(2)

def transfer_socket():
    global  bpmdata
    global count_
    count_ =0
    while True:
        conn, addr = sock_transfer.accept()
        data = conn.recv(1024)
        data = data.decode("utf-8").strip()
        res = bpmdata
        if res =='N':
            res = "none"
        elif res =='C':
            res = "cal"
        conn.sendall(res.encode("utf-8"))
```

```
        if count_ == 0 :
            count_ =1
            main_dialog.transfer_tB_set_text("TRANSFER : {} >> {}".format(res,
addr))
        elif count_ ==1:
            count_=0
            main_dialog.transfer_tB_set_text("*TRANSFER : {} >> {}".format(re
s, addr))

        conn.close()
t2 = threading.Thread(target=transfer_socket)
t2.daemon = True
t2.start()

def read_socket():
    global bpmdata
    global count
    count=0
    while True:
        data, addr = sock_receive.recvfrom(1024)

        bpmdata = data.decode('utf-8')
        if count == 0:
            main_dialog.receive_tB_set_text("RECEIVED : {} << {}".format(data.d
ecode('utf-8'), addr))
            count = 1
        elif count ==1:
            main_dialog.receive_tB_set_text("*RECEIVED : {} << {}".format(data.
decode('utf-8'), addr))
            count = 0
        if bpmdata != 'N' and bpmdata != 'C':
            main_dialog.set_SEG(bpmdata)
            main_dialog.set_light("green")
```

```
            elif bpmdata.find('N') != -1 :
                main_dialog.set_light("red")
            elif bpmdata.find('C') != -1:
                main_dialog.set_light("orange")

t1 = threading.Thread(target=read_socket)
t1.daemon = True
t1.start()

# def start_video():
#       os.system(video_start_cmd)
#
# t = threading.Thread(target= start_video)
# t.daemon = True
# t.start()

main_dialog.show()
app.exec_()
```

③ Raspberry pi

- 전체구성



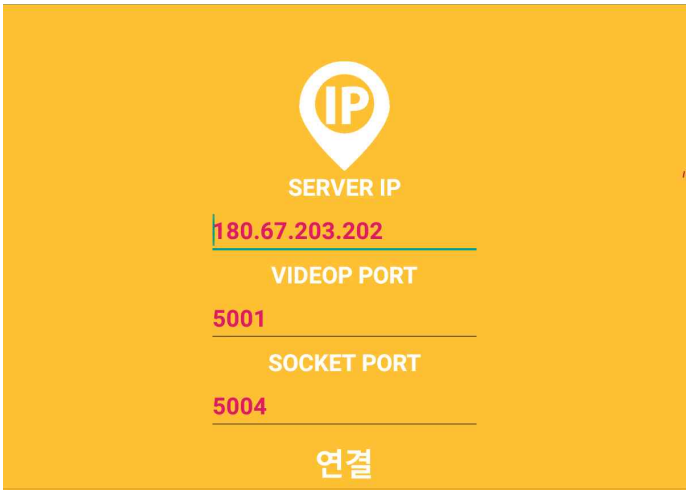
· **gst_main.py**

```
import socket
import serial
import posix
from fcntl import ioctl
import RPi.GPIO as GPIO
import time
import os
import threading
from random import *

serial_data = ''

arduino=serial.Serial("/dev/ttyUSB0",115200)
sock = socket.socket( socket.AF_INET , socket.SOCK_DGRAM)
def read_serial():
    global serial_data
    while True:
        read_ser=arduino.readline()
```

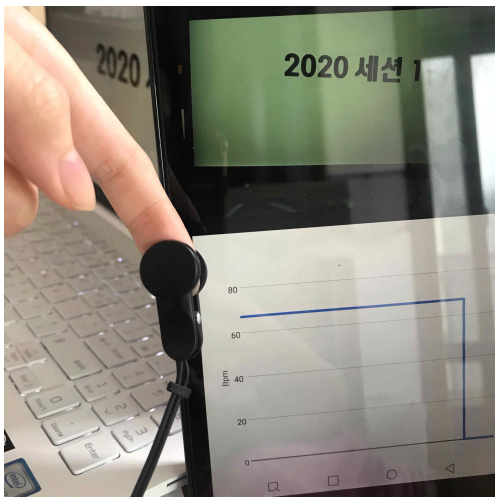
```
            serial_data =read_ser.decode()[:-2]
            print(serial_data)
            sock.sendto( serial_data.encode(),('180.67.203.202',5003))
            print("socket transfer : ",serial_data.encode('UTF-8'))


read_serial_thread = threading.Thread(target=read_serial)
read_serial_thread.start()

while True:

    time.sleep(1)
```

2) Demo

① 라즈베리파이의 전원을 켜고 현재 연결할 ip를 입력한다.



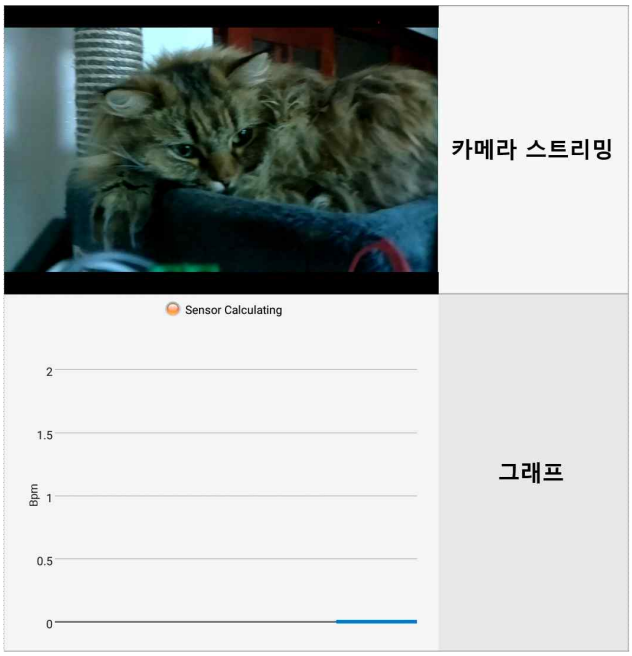
② 하드웨어의 클립형 PPG센서와 혈류가 흐르는 신체 부위를 연결시킨다.

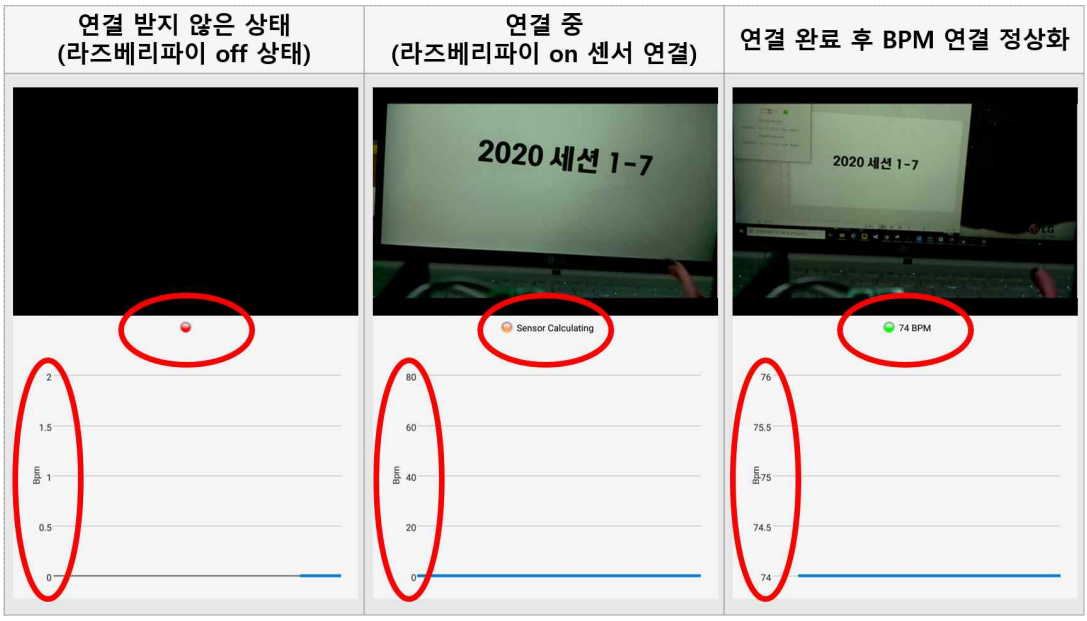


③

❶ Application

- 카메라 스트리밍 화면과 그래프를 통해 연결 상태를 확인한다.





❷ Server

- 서버 실행 후 서버 창을 통해 연결이 원활히 이루어지고 있는지 확인한다.

BPM 0 (우측 사각형 붉은색)	BPM 받는 중 (우측 사각형 노란색)	BPM 받음 (우측 사각형 초록색)
		

Ⅲ. 결론

1. 연구 결과

1) 구현 결과

① 카메라 실시간 스트리밍은 문제없이 가능하였다.

② ppg 센서를 이용한 심박수 측정 테스트는 사람의 손가락과 동물의 귀를 이용해 측정 진행하였다. 그래프와 숫자를 통한 데이터 전달(어플리케이션과 서버)은 문제없이 가능했다.

2) 문제점 및 향후 연구과제

ECG 센서를 활용하여 심전도를 측정한다면 의료적인 면에서 더 좋은 결과를 가져올 수 있었겠지만, 반려동물의 털 문제에 계측기와 증폭기 설계 과정에서 어려움이 있어 PPG 센서로 변경되었다. 또한 하드웨어의 크기를 최대한 줄여보았지만, 라즈베리파이와 카메라모듈 사용으로 인해 배터리 용량을 줄일 수 없었기에 대형견에 탈부착 가능한 수준으로만 제작을 하게 되었다. 이러한 문제점을 계측기와 증폭기에 대한 꾸준한 연구를 통해 극복해나갈 예정이다.

2. 작품제작 소요재료 목록

품목	수량	개당 가격	총 가격
5인치 라즈베리파이 HDMI 터치스크린 LCD	1	44,000	44,000
라즈베리파이 카메라모듈 V2	1	30,800	30,800
라즈베리파이 카메라 케이스	1	12,100	12,100
라즈베리파이4 B	1	80,300	80,300
3.6V 18650 리튬이온 충전 건전지	4	8,900	35,600
18650 x 4 리튬이온 배터리홀더	1	1,320	1,320
라즈베리파이4 B ABS 쿨러 케이스	1	4,400	4,400
마이크로SD카드32GB	1	8,800	8,800
Micro HDMI to HDMI 변환 젠더	2	770	1,540
5V 3A 어댑터	1	5,500	5,500
PCB 70x300	1	12,100	12,100
저항셋	1	6,600	6,600
아두이노 나노	2	5,500	11,000
18650 배터리 충전기	1	16,400	16,400
UA741CP	10	600	6,000

구리선	1	2,400	2,400
캐패시터 키트(극성)	1	8,800	8,800
캐패시터 키트(무극성)	1	13,200	13,200
최종금액			300,860

참고자료

[1] Smart Pet Clothing for Monitoring of Health and Mood
Yu-Jin Lin, Chen-Wei Chuang, Chun-Yueh Yen, Sheng-Hsin Huang and Shuenn-Yuh Lee, Senior Member, IEEEDepartment of Electronic Engineering, National Cheng Kung University, Tainan, Taiwan

[2] R. Brugarolas, T. Latif, J. Dieffenderfe, et. Al., "Wearable Heart Rate Sensor Systems for Wireless Canine Health Monitoring", IEEE Sensor Journal, vol. 16, no. 10, pp.3454-3464, May 2016.

[3] M. Brložnik, V. Avbelj, "A case report of long-term wireless electrocardiographic monitoring in a dog with dilated cardiomyopathy", Int. Conven. Infor. and Comm. Technology, Electronics and Microelectronics (MIPRO), May 22-26, 2017, pp.303-307.

[4] A. Krvavica, S. Likar, M. Brloznik, A. Domanjko-Petric, V. Avbelj, "Comparison of Wireless Electrocardiographic Monitoring and Standard ECG in Dogs", Int. Conven. Infor. and Comm. Technology, Electronics and Microelectronics (MIPRO), May 30- June 26, 2016, pp.396-399.

[5] M. Šarlija, F. Jurišić, S. Popović, "A Convolutional Neural Network Based Approach to QRS Detection", International Symposium on Image and Signal Processing and Analysis, 2017, pp.121-125.

[6] S. Kiranyaz, T. Ince, and M. Gabbouj,"Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks", IEEE Transactions on Biomedical Engineering, vol. 63, no. 3, pp.664-675, March 2016.

[7] 지인배, 김현중, 김원태, 서강철. (2017.10.). 반려동물 연관산업 발전방안 연구 (Development Strategies for the Companion Animal Industry) www.krei.re.kr

[8] 그래프뷰 셋팅 참조 : https://github.com/jjoe64/GraphView/wiki/Documentation

[9] NDK 참조 : https://yucaroll.tistory.com/1