Supporting Information

Open-source fluorescence spectrometer for noncontact scientific research and education

Hyejeong Jeong,[†] Suyeon Shin,[‡] Jihun Hwang,[†] Yoon-Jin Kim,[‡] and Sungyoung Choi[†,‡]

[†]Department of Biomedical Engineering, Hanyang University, Seoul 04763, Republic of Korea

[‡]Department of Electronic Engineering, Hanyang University, Seoul 04763, Republic of Korea

Contents:

1. Supplementary figures

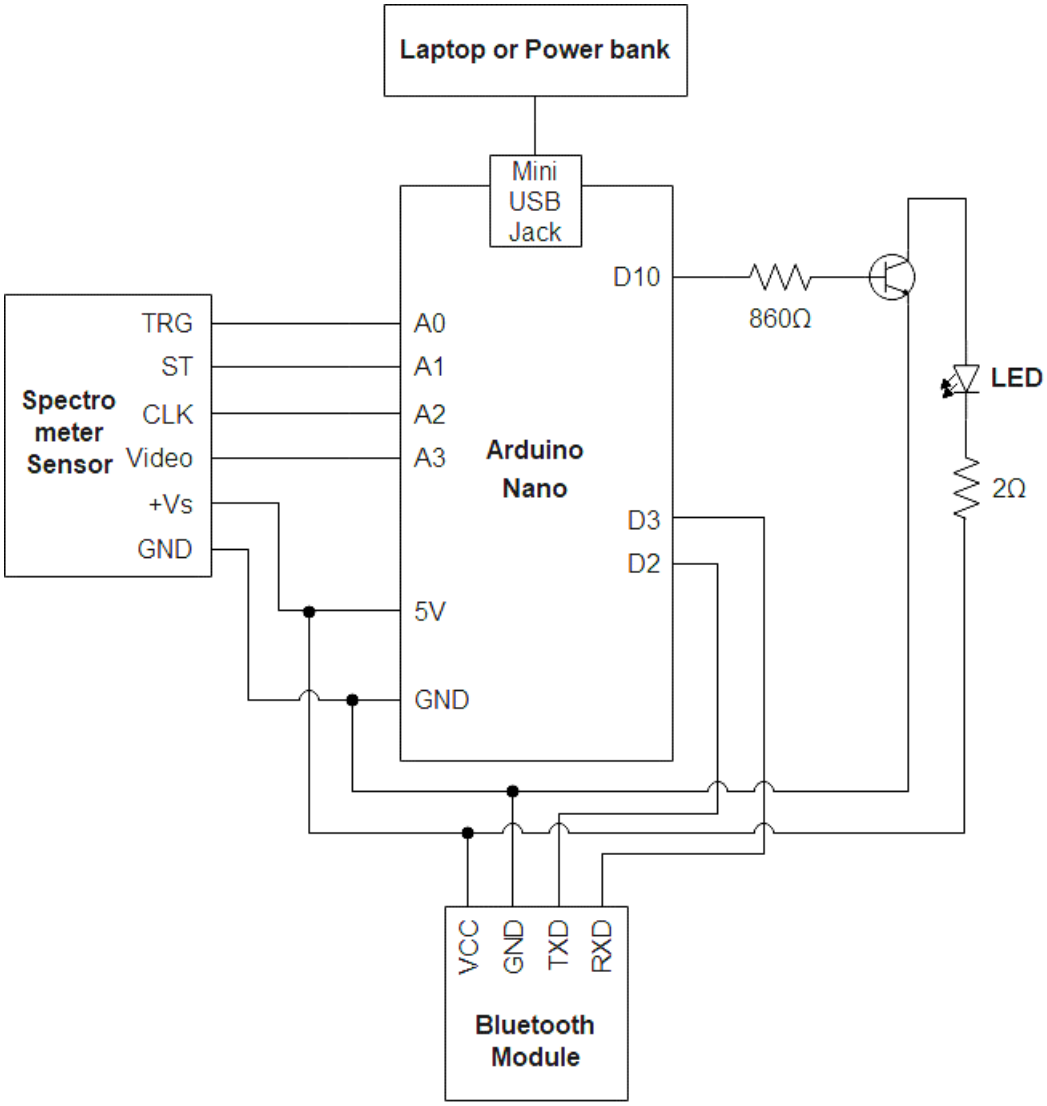


Figure S1. Circuit diagram of OpenFS indicating the connections between the Arduino Nano, spectrometer sensor, Bluetooth module, LED, and power source (laptop or power bank).

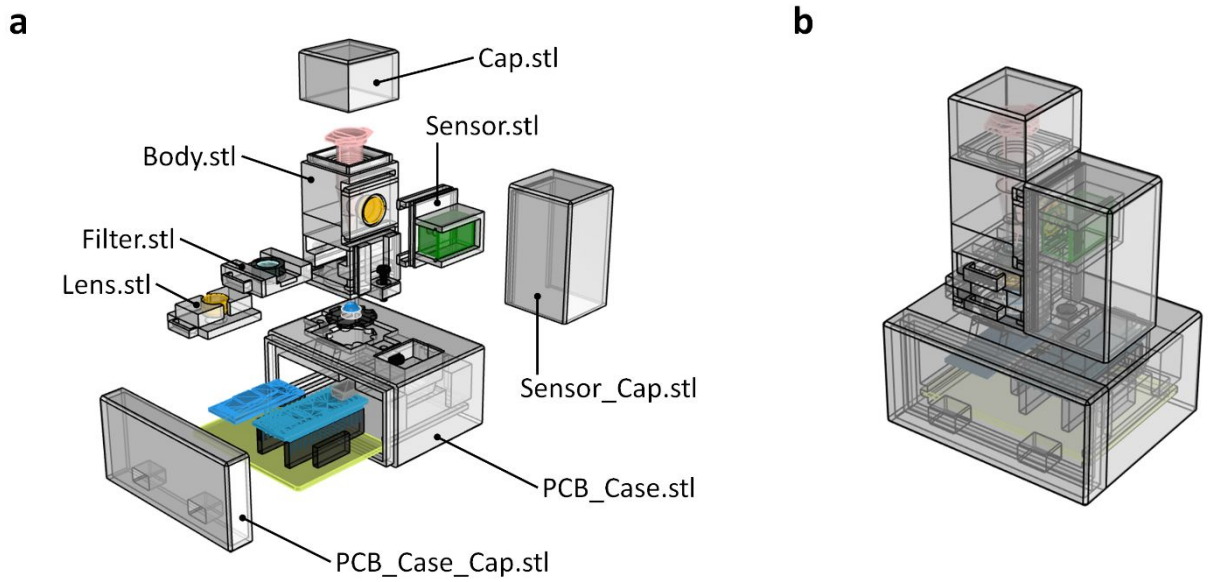


Figure S2. (a) STL file names for 3D-printing models and (b) assembly of the 3D-printing models. When using a cuvette as a sample container, Body.stl and Cap.stl can be replaced with Cuvette_Body.stl and Cuvette_Cap.stl.

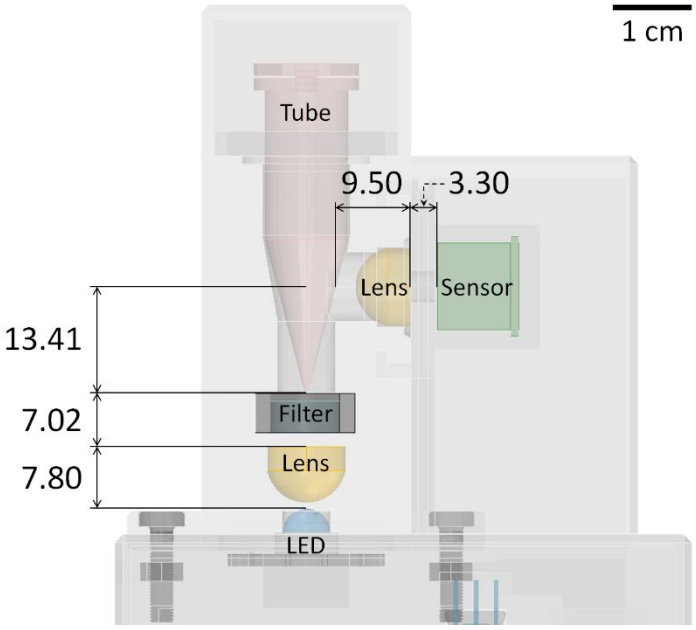


Figure S3. Configuration of OpenFS indicating the distances between the optical components. All distances are expressed in mm. Two condenser lenses used in the OpenFS were used to focus light and increase the detected fluorescence intensity.

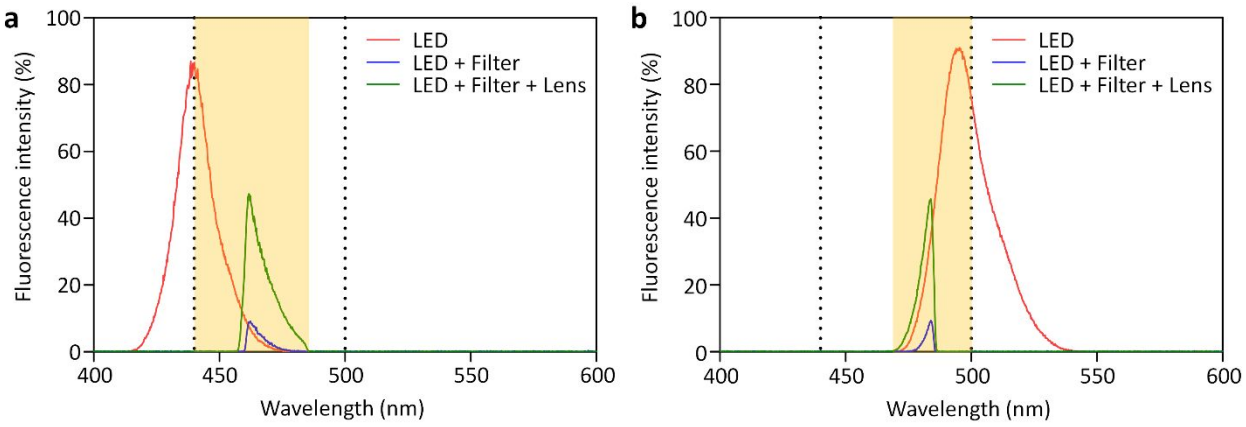


Figure S4. Spectral measurement of different combinations of optical elements using (a) a blue LED and (b) a cyan LED. The two dotted lines represent 440 nm and 500 nm, respectively. The yellow part represents the area where the spectrum of each LED overlaps with the range of interest (440 to 500 nm).

2. Supplementary tables

Table S1. Cost analysis of OpenFS

Item	Cost
3D print material	$30.88
3W blue LED	$3.00
Aspheric condenser lens (2 pcs)	$45.00
Bandpass filter	$140.00
Spectrometer sensor	$295.99
Arduino Nano	$20.28
Bluetooth module	$3.25
Transistor	$0.14
1.5 mL microcentrifuge tube	$0.02
Total	**$538.56**

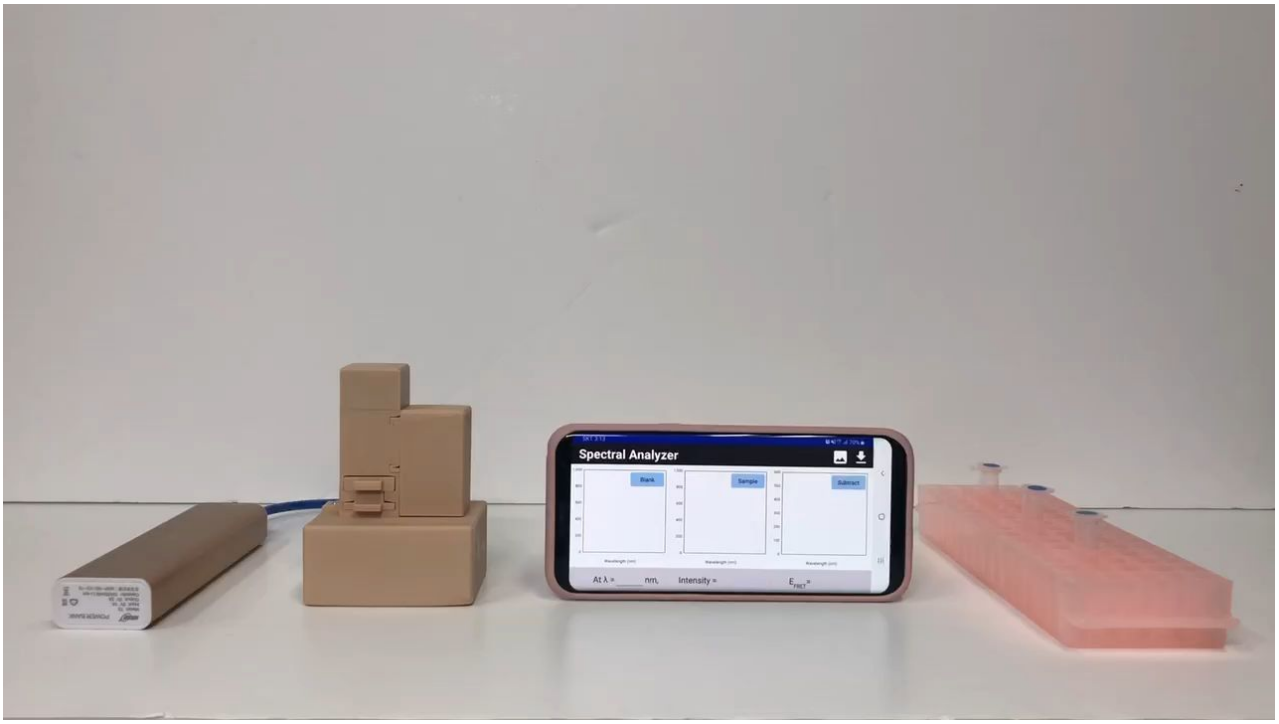
Table S2. Comparison of commercial fluorescence spectrometers and OpenFS

Specifications	Duetta	USB4000-FL-450	OpenFS
Manufacturer	HORIBA	Ocean Optics	In this paper
Cost	$34,100	$11,090	$540
Size (cm)	43.2 x 51.8 x 36.6	8.9 x 12.0 x 3.4	7.3 × 6.1 × 10.2
Weight	20.4 kg	310 g	180 g
Measurement	Absorbance, Emission	Emission	Emission
Wavelength range	250 - 1,100 nm	360 - 1,000 nm	340 - 850 nm
Spectral resolution (FWHM)	~1 nm	~10 nm	12 - 15 nm
Light source	75W Xenon arc lamp	470 nm LED	3W blue LED
Sample holder	1 cm path length cuvette	1 cm path length cuvette	1.5 mL micro tube
Power cable	110V	USB	USB
Data transfer	USB	USB	USB or Bluetooth
Analysis	PC	PC	PC or Smartphone

Table S3. Oligonucleotide sequences

DNA	Sequence
Cy3-labeled DNA probe	5′-GAC GCG ATC ACC ACG-3′-Cy3
Cy5-labeled DNA probe	Cy5-5′-ACC AGC CAG CTG AGC-3′
Target DNA	5′-TAG GCT CAG CTG GCT GGT CGT GGT GAT CGC GTC CAA-3′

3. Supplementary Video



Movie S1. Wireless FRET-based detection of target DNA using OpenFS.

4. Student handout

Student Handout

Open-source fluorescence spectrometer for noncontact scientific research and education

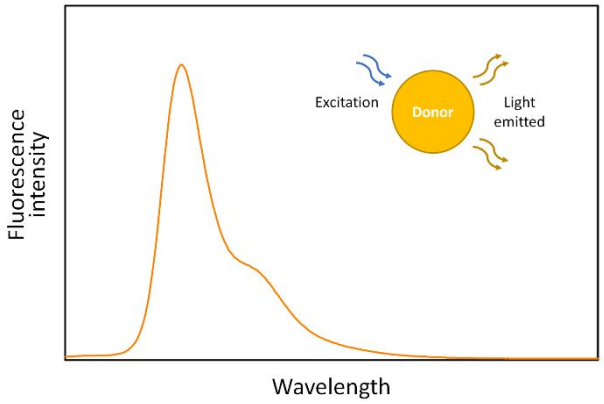
Conventional fluorescence spectrometers are difficult to use in resource-limited settings due to several factors such as size, cost, and professional manpower. To enable noncontact research of non-expert users (i.e. students) even in places with limited equipment, open-source fluorescence spectrometer (OpenFS) was developed by compactly assembling optical and electronic elements in a 3D-printed housing. In this experiment, by using OpenFS and a customized Android application, students first observe fluorescence emission spectra for different concentrations of fluorophore. Then, students integrate OpenFS with a FRET-based DNA sensor and compare the spectral shape with and without FRET.

Introduction

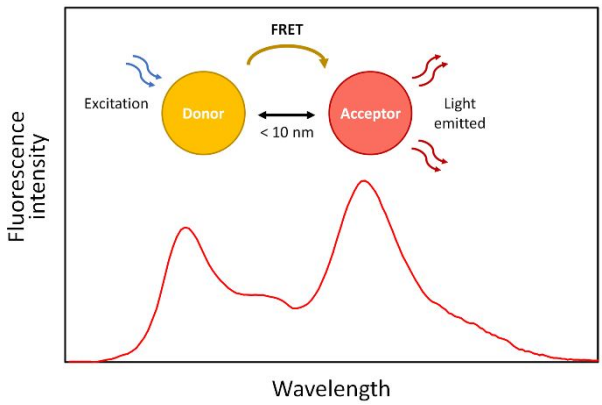
Fluorescence, a type of luminescence, refers to the emission of light that occurs when a molecule rises to an excited state by absorbing a certain wavelength of light and falls back into a low-energy ground state. In general, the wavelength of light emitted is longer than the excitation wavelength. Every fluorophore has characteristic excitation and emission range of wavelengths. Fluorescence spectroscopy analyzes fluorescence from a molecule based on its fluorescent properties and it can be used to measure compounds in a solution.

Förster Resonance Energy Transfer (FRET) is a phenomenon describing energy transfer between two neighboring fluorescent molecules called FRET pair. An excited donor group transfers energy to an acceptor group through a non-radiative process. FRET is detected by a spectrofluorometer, which measures the fluorescence of acceptor or donor. The FRET efficiency is a proportion of energy transfer from donor to acceptor fluorophore. The closer the distance between the donor and the acceptor, the larger the overlap area between the emission spectrum of the donor and the excitation spectrum of the acceptor, the more parallel or antiparallel their dipole moments, the higher the FRET efficiency.

Fluorescence Emission Spectrum
< No FRET >



Fluorescence Emission Spectrum
< FRET >



Procedure

1. Thaw DNA samples dissolved in distilled water at a concentration of 100 pmol/μL at room temperature.
2. Dilute DNA samples to an appropriate concentration in a 1X Tris-EDTA buffer (50 mM Tris-HCl (pH 7.5), 100 mM NaCl, 5 mM KCl, 1 mM MgCl2, and 0.1 mM EDTA).
3. Skip this process if the sample is not FRET-based DNA. To measure FRET-based DNA, a hybridization process should be performed. Prepare a sample by mixing donor-labeled DNA probes, acceptor-labeled DNA probes, and target DNA molecules in a suitable ratio. Then, heat the sample at 95°C for 5 min in a heat block, and cool the solution slowly to room temperature.
4. Put 500 μL of distilled water in a 1.5 mL microcentrifuge tube and use as a blank.
5. Put 500 μL of DNA sample in another 1.5 mL microcentrifuge tube and use as a test sample.
6. After inserting a microcentrifuge tube containing a blank sample (distilled water), press the “Blank” button to receive and display the blank signal.
7. After inserting another tube containing a test sample (DNA sample), press the “Sample” button to obtain the fluorescence signal from the test sample, and the final emission spectrum (Sample – Blank) is obtained by pressing the “Subtract” button.

Data analysis

1. By entering a desired wavelength at the bottom of the app, a user can check the fluorescence intensity at that wavelength.
2. The result screen can be saved as an image by pressing the screenshot button, and the graph data can be saved as a text file by pressing the download button.
3. If necessary, the user can quickly check the FRET efficiency automatically calculated by the following equation: $E_{FRET} = I_A/(I_D + I_A)$, where I_A and I_D are the fluorescence peak intensities of the acceptor and donor, respectively.

Results

Spectral analysis of fluorescence emission

Contents	Results		
Fluorophore name			
Sample concentration			
Fluorescence peak intensity (In case of Cy3, λ = 565.2 nm)			

FRET-based DNA sensor

Contents	Results		
FRET pair (Donor/Acceptor)			
Target concentration			
Fluorescence peak intensity of donor (I_D) (In case of Cy3, λ = 565.2 nm)			
Fluorescence peak intensity of acceptor (I_A) (In case of Cy5, λ = 668.9 nm)			
FRET efficiency ($E_{FRET} = I_A / (I_D+I_A)$)			

Survey

No.	Question	Score				
1	(Experience) How much experience do you have with fluorescence spectroscopy? 1. Who has never studied or heard of spectrometers. 2. Who has studied or heard of spectrometers. 3. Who has used a spectrometer once or twice. 4. Who has used a spectrometer more than a dozen times. 5. Who has developed or researched a spectrometer.	5	4	3	2	1
2	(Usability) Is there no difficulty in understanding or handling the developed device (OpenFS)? (Usability)	5	4	3	2	1
3	(Portability) Is it easy to carry?	5	4	3	2	1
4	(Efficiency) Is it possible to measure and analyze quickly?	5	4	3	2	1
5	(Accuracy) Are the results obtained through the developed device (OpenFS) and the theoretical results consistent?	5	4	3	2	1
6	(Necessity) Are you willing to use this device (OpenFS) if it is open-source?	5	4	3	2	1
7	(Recommendation) Would you like to recommend it to others?	5	4	3	2	1
Strongly agree = 5, Agree = 4, Neutral = 3, Disagree = 2, Strongly disagree = 1						

5. Supplementary Codes

The following Arduino source codes were developed based on and by revising the C12880MA code posted on GitHub (https://github.com/groupgets/c12880ma). The Arduino source codes are used for device operation and data acquisition; the following Java code is used for graph display and spectral analysis in the Android application.

(1) Arduino source code for wired OpenFS (acquiring data through Arduino serial monitor on computer)

```
#define SPEC_TRG A0
#define SPEC_ST A1
#define SPEC_CLK A2
#define SPEC_VIDEO A3

#define SPEC_CHANNELS 288 // New Spec Channel
uint16_t data[SPEC_CHANNELS];

#define N 3   // MAF N
uint16_t filteredData[SPEC_CHANNELS];

#define INT_TIME 32   //Integration Time

int led_pin=10;
int value=255; //0~255

void setup()
{
  //Set desired pins to OUTPUT
  pinMode(SPEC_CLK, OUTPUT);
  pinMode(SPEC_ST, OUTPUT);
  pinMode(LASER_404, OUTPUT);
  pinMode(WHITE_LED, OUTPUT);
  pinMode(led_pin, OUTPUT);

  digitalWrite(SPEC_CLK, HIGH); // Set SPEC_CLK High
  digitalWrite(SPEC_ST, LOW); // Set SPEC_ST Low

  Serial.begin(115200); // Baud Rate set to 115200
}

/*
  This function reads spectrometer data from SPEC_VIDEO
  See the Timing Chart in the Datasheet for more info
*/

void readSpectrometer()
{
  int delayTime = 1;

  // Start clock cycle and set start pulse to signal start
  digitalWrite(SPEC_CLK, LOW);
  delayMicroseconds(delayTime); //10000us = 0.01s
  digitalWrite(SPEC_CLK, HIGH);
  delayMicroseconds(delayTime);
  digitalWrite(SPEC_CLK, LOW);
  digitalWrite(SPEC_ST, HIGH);
  delayMicroseconds(delayTime);
```

```
//Sample for a period of time
for (int i = 0; i < 10; i++) {
   digitalWrite(SPEC_CLK, HIGH);
   delayMicroseconds(delayTime);
   digitalWrite(SPEC_CLK, LOW);
   delayMicroseconds(delayTime);
}

//Clock cycles for integration time //1
for (int i = 0; i < INT_TIME; i++)
{
   digitalWrite(SPEC_CLK, HIGH);
   digitalWrite(SPEC_CLK, LOW);
}

//Clock cycles for integration time //2
for (int i = 0; i < INT_TIME; i++)
{
   digitalWrite(SPEC_CLK, HIGH);
   digitalWrite(SPEC_CLK, LOW);
}

//Clock cycles for integration time //3
for (int i = 0; i < INT_TIME; i++)
{
   digitalWrite(SPEC_CLK, HIGH);
   digitalWrite(SPEC_CLK, LOW);
}

//Clock cycles for integration time //4
for (int i = 0; i < INT_TIME; i++)
{
   digitalWrite(SPEC_CLK, HIGH);
   digitalWrite(SPEC_CLK, LOW);
}

//Set SPEC_ST to low
digitalWrite(SPEC_ST, LOW);

//Sample for a period of time
for (int i = 0; i < 87; i++)
{
   digitalWrite(SPEC_CLK, HIGH);
   delayMicroseconds(delayTime);
   digitalWrite(SPEC_CLK, LOW);
   delayMicroseconds(delayTime);
}

//One more clock pulse before the actual read
digitalWrite(SPEC_CLK, HIGH);
delayMicroseconds(delayTime);
digitalWrite(SPEC_CLK, LOW);
delayMicroseconds(delayTime);

//Read from SPEC_VIDEO
for (int i = 0; i < SPEC_CHANNELS; i++)
{
   data[i] = analogRead(SPEC_VIDEO);
   filteredData[i] = 0;
   if (i < N) {
      filteredData[i] = data[i];
   }
```

```
      if (i >= N) {
         for (int k = 0; k < N; k++) {
            filteredData[i - 1] += data[i - k];
      }
         filteredData[i - 1] = filteredData[i - 1] / N;
      if (i >= SPEC_CHANNELS - 1) {
            filteredData[i] = data[i];
      }
      }
      digitalWrite(SPEC_CLK, HIGH);
      delayMicroseconds(delayTime);
      digitalWrite(SPEC_CLK, LOW);
      delayMicroseconds(delayTime);
   }

   //Set SPEC_ST to high
   digitalWrite(SPEC_ST, HIGH);

   //Sample for a small amount of time
   for (int i = 0; i < 7; i++)
   {
      digitalWrite(SPEC_CLK, HIGH);
      delayMicroseconds(delayTime);
      digitalWrite(SPEC_CLK, LOW);
      delayMicroseconds(delayTime);
   }

   digitalWrite(SPEC_CLK, HIGH);
   delayMicroseconds(delayTime);
}

/*
   The function below prints out data to the terminal or
   processing plot
*/

void printData()
{
   for (int i = 0; i < SPEC_CHANNELS; i++)
   {
      Serial.print(filteredData[i]);
      Serial.println(" ");
   }
   Serial.print("\n");
}

void loop()
{
   analogWrite(led_pin, value);
   readSpectrometer();
   printData();
   delay(1000);
}
```

(2) Arduino source code for wireless OpenFS (acquiring data through the Android application on a smartphone using the Bluetooth module)

```
#include <SoftwareSerial.h>
SoftwareSerial BT(2,3);
#include <avr/pgmspace.h>
#define SPEC_TRG              A0
```

```
#define SPEC_ST                A1
#define SPEC_CLK               A2
#define SPEC_VIDEO             A3

#define SPEC_CHANNELS 288 // New Spec Channel
uint16_t* data;

#define N 3    // MAF N

uint16_t tempData;
uint16_t tempData2;

#define INT_TIME 32    //Integration Time

int led_pin=10;
int value=255; //0~255

char buffer[100];
boolean blankStat, sensorStat, absorStat;

uint16_t firstData[SPEC_CHANNELS];
uint16_t sampleData[SPEC_CHANNELS];

int cnt;
float blankTemp;
float sampleTemp;

void setup()
{
   //Set desired pins to OUTPUT
   pinMode(SPEC_CLK, OUTPUT);
   pinMode(SPEC_ST, OUTPUT);
   pinMode(led_pin, OUTPUT);

   pinMode(8,INPUT_PULLUP);
   pinMode(9,INPUT_PULLUP);

   digitalWrite(SPEC_CLK, HIGH); // Set SPEC_CLK High
   digitalWrite(SPEC_ST, LOW); // Set SPEC_ST Low

   Serial.begin(9600); // Baud Rate set to 9600
   BT.begin(9600);
}

/*
   This function reads spectrometer data from SPEC_VIDEO
   See the Timing Chart in the Datasheet for more info
*/

void readSpectrometer(boolean readType){

uint8_t delayTime = 1; // delay time

   // Start clock cycle and set start pulse to signal start
   digitalWrite(SPEC_CLK, LOW);
   delayMicroseconds(delayTime);
   digitalWrite(SPEC_CLK, HIGH);
   delayMicroseconds(delayTime);
   digitalWrite(SPEC_CLK, LOW);
   digitalWrite(SPEC_ST, HIGH);
   delayMicroseconds(delayTime);
```

```
//Sample for a period of time
for (int i = 0; i < 10; i++) {
   digitalWrite(SPEC_CLK, HIGH);
   delayMicroseconds(delayTime);
   digitalWrite(SPEC_CLK, LOW);
   delayMicroseconds(delayTime);
}

//Clock cycles for integration time //1
for (int i = 0; i < INT_TIME; i++)
{
   digitalWrite(SPEC_CLK, HIGH);
   digitalWrite(SPEC_CLK, LOW);
}

//Clock cycles for integration time //2
for (int i = 0; i < INT_TIME; i++)
{
   digitalWrite(SPEC_CLK, HIGH);
   digitalWrite(SPEC_CLK, LOW);
}

//Clock cycles for integration time //3
for (int i = 0; i < INT_TIME; i++)
{
   digitalWrite(SPEC_CLK, HIGH);
   digitalWrite(SPEC_CLK, LOW);
}

//Clock cycles for integration time //4
for (int i = 0; i < INT_TIME; i++)
{
   digitalWrite(SPEC_CLK, HIGH);
   digitalWrite(SPEC_CLK, LOW);
}

//Set SPEC_ST to low
digitalWrite(SPEC_ST, LOW);

//Sample for a period of time
for (int i = 0; i < 87; i++)
{
   digitalWrite(SPEC_CLK, HIGH);
   delayMicroseconds(delayTime);
   digitalWrite(SPEC_CLK, LOW);
   delayMicroseconds(delayTime);
}

//One more clock pulse before the actual read
digitalWrite(SPEC_CLK, HIGH);
delayMicroseconds(delayTime);
digitalWrite(SPEC_CLK, LOW);
delayMicroseconds(delayTime);

//Read from SPEC_VIDEO
if(readType) data = firstData;
else data = sampleData;
for (int i = 0; i < SPEC_CHANNELS; i++)
{
   data[i] = analogRead(SPEC_VIDEO);
   //filteredData[i] = 0;
   if (i < N) {
```

```
            //filteredData[i] = data[i];
            if(i == N-1)
               tempData2 = data[i-1];
         }
         if (i >= N) {
            /*for (int k = 0; k < N; k++) {
               filteredData[i - 1] += data[i - k];
            }*/
            //filteredData[i - 1] = filteredData[i - 1] / N;
            tempData = data[i-1];
            if(i < SPEC_CHANNELS - 1) {
               data[i-1] = (tempData2 + data[i-1] + data[i])/N;
            }
            tempData2 = tempData;
            /*if (i >= SPEC_CHANNELS - 1) {
                  filteredData[i] = data[i];
            }*/
         }
         digitalWrite(SPEC_CLK, HIGH);
         delayMicroseconds(delayTime);
         digitalWrite(SPEC_CLK, LOW);
         delayMicroseconds(delayTime);
      }

      //Set SPEC_ST to high
      digitalWrite(SPEC_ST, HIGH);

      //Sample for a small amount of time
      for (int i = 0; i < 7; i++)
      {
         digitalWrite(SPEC_CLK, HIGH);
         delayMicroseconds(delayTime);
         digitalWrite(SPEC_CLK, LOW);
         delayMicroseconds(delayTime);
      }

      digitalWrite(SPEC_CLK, HIGH);
      delayMicroseconds(delayTime);
   }

   /*
      The function below prints out data to the terminal or
      processing plot
   */

   void printBlank()
   {
      if(blankStat == true)
      {
            readSpectrometer(true);

            cnt ++;
           if(cnt == 3)
           {

                  for (int i = 0; i < SPEC_CHANNELS; i++)
                  {

                        firstData[i] = data[i];
                        sprintf(buffer,"a%u", firstData[i]);

                        BT.println(buffer);
```

```
            }
            cnt = 0;
            blankStat = false;
        }
    }

     if(sensorStat == true)
    {
          readSpectrometer(false);


         cnt ++;
        if(cnt == 3)
        {

            for (int i = 0; i < SPEC_CHANNELS; i++)
            {

                sampleData[i] = data[i];
                //sprintf(buffer,"b%d",sampleTemp);
                sprintf(buffer,"b%u", sampleData[i]);

                BT.println(buffer);
            }
            cnt = 0;
            sensorStat = false;
        }
    }
}

/*
   The function below subtracts blank data from sample data
*/

int16_t subTemp;
void subCalculation()
{
      if(absorStat == true)
      {

           for (int i = 0; i < SPEC_CHANNELS; i++)
           {
                subTemp = sampleData[i] - firstData[i];


                 //Serial.println(subData[i],6);

                 sprintf(buffer,"c%d",subTemp);

                 BT.println(buffer);
           }
             absorStat = false;
      }
}


bool dataFlag;
String rxData, strParsed;
int arrNum;
int subIndex;

void loop()
```

```
{
      analogWrite(led_pin, value);
      printBlank();
      delay(1000);
      subCalculation();
      //delay(1000);

        // Serial.println(digitalRead(8));
      if(!(digitalRead(9)) && blankStat == false)
            blankStat = true ;

      if(!(digitalRead(8)) && sensorStat == false)
            sensorStat = true ;

        if(BT.available())
      //if(Serial.available())
      {
                  //char dataGet = BT.read();
                    String strGet    =BT.readStringUntil('\n');

                    if(strGet.length()==1)
                    {
                              if(strGet == "a")
                              {
                                        //Serial.println("aaaaaa");
                                          if(blankStat == false)
                                              blankStat = true;
                              }
                              if(strGet == "b")
                              {
                                      //Serial.println("bbbbbbb");
                                          if(sensorStat == false)
                                              sensorStat = true;
                              }
                              if(strGet == "c")
                              {
                                      //Serial.println("ccccccc");
                                          if(absorStat == false)
                                              absorStat = true;
                              }
                    }
      }
}
```

(3) Java source code for the Android application

```
package com.example.androidproject.a0415_graph_2;

import android.app.Activity;
import android.app.AlertDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.util.Log;
```

```
import android.view.KeyEvent;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;

import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.components.AxisBase;
import com.github.mikephil.charting.components.Description;
import com.github.mikephil.charting.components.Legend;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.components.YAxis;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.formatter.ValueFormatter;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;
import com.jraska.falcon.Falcon;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Set;
import java.util.UUID;
import java.util.regex.Pattern;



public class MainActivity extends Activity
{
	private static final String FILE_NAME = "example.txt";

	public static final int NOTI_ID = 999;
	static final int REQUEST_ENABLE_BT = 10;
	int mPariedDeviceCount = 0;

	//objects for Bluetooth control
	Set<BluetoothDevice> mDevices;
	BluetoothAdapter mBluetoothAdapter;

	BluetoothDevice mRemoteDevie;
	// BluetoothSocket for communication with arduino
	BluetoothSocket mSocket = null;
	OutputStream mOutputStream = null;
	InputStream mInputStream = null;
	String mStrDelimiter = "\n";
	char mCharDelimiter =    '\n';

	Thread mWorkerThread = null;
	byte[] readBuffer;
```

```
    int readBufferPosition;

    Button Blank;
    Button Sample;
    Button Subtract;

    ImageButton screenShotButt;
    ImageButton imageBtn;

    TextView absSample;
    TextView e_fret_text;

    EditText waveLength ;
    InputMethodManager imm;

    int[] waveLengthValues = new int[288];
    int startIndex, endIndex = 0;
    String[] waveValues = new String[288];


    int cnt;
    int result1;
    int result2;
    int result3;

    String dataWrite;

    int classfy;

    int[] blankData = new int[288];
    int[] sampleData = new int[288];
    int[] absData = new int[288];

    int absMax = 0;
    int index1;
    int index2;
    int index3;

    private LineChart mChart1;
    private LineChart mChart2;
    private LineChart mChart3;

    private static final int X_COUNT_MAX = 288;

    private Thread thread;
    private boolean plotData = true;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

        //wave length array of sensor
        for (int A3=0; A3 < 288; A3++) {
            double curWaveLength = 3.068464691*100+2.707179585*1*A3-1.45040638*1/1000*A3*A3-
4.596809405*1/1000000*A3*A3*A3-
3.105844784*1/1000000000*A3*A3*A3*A3+2.269126371*1/100000/1000000*A3*A3*A3*A3*A3;

            waveLengthValues[A3] = (int)Math.round(curWaveLength);
            if(waveLengthValues[A3] >= 528 && startIndex == 0) startIndex = A3;
```

```
            else if(waveLengthValues[A3] >= 770 && endIndex == 0) endIndex = A3;
            waveValues[A3] = Long.toString(Math.round(curWaveLength));
        }

        imm = (InputMethodManager)getSystemService(INPUT_METHOD_SERVICE);     // lower
keyboard

        Blank = findViewById(R.id.blank);
        Sample = findViewById(R.id.sample);
        Subtract = findViewById(R.id.subtract);

        waveLength = (EditText) findViewById(R.id.waveLength);
        absSample = (TextView) findViewById(R.id.absSample);
        e_fret_text = (TextView) findViewById(R.id.e_fret);

        screenShotButt = findViewById(R.id.screenShotBtn);
        imageBtn = findViewById(R.id.imageBtn);

        Blank.setOnClickListener(new View.OnClickListener(){
            //get blank data from Arduino by sending command
            @Override
            public void onClick(View view) {
                mChart1.clearValues();
                mChart3.clearValues();
                sendData("a\n");
            }
        });

        Sample.setOnClickListener(new View.OnClickListener(){
            //get sample data from Arduino by sending command
            @Override
            public void onClick(View view) {
                mChart2.clearValues();
                mChart3.clearValues();
                sendData("b\n");
            }
        });

        Subtract.setOnClickListener(new View.OnClickListener(){
            //get subtracted data from Arduino by sending command
            @Override
            public void onClick(View view) {
                mChart3.clearValues();
                sendData("c\n");
            }
        });

        waveLength.setOnEditorActionListener(new TextView.OnEditorActionListener()
        {
            //display written wavelength's amplitude
            @Override
            public boolean onEditorAction(TextView textView, int actionId, KeyEvent keyEvent) {
                Log.i("chart", "wave");
                if (Pattern.matches("^[0-9]+$", waveLength.getText())) {
                    if (Pattern.matches("^[3-9][0-9][0-9]+$", waveLength.getText())) {

                        int waveLengthIndex = Integer.parseInt(waveLength.getText().toString());

                        int closestWaveLength = 0;
                        int waveIndex = 0;

                        if(waveLengthIndex <= waveLengthValues[0]) {
```

```
                        closestWaveLength = waveLengthValues[0];
                    }
                    else if(waveLengthIndex >= waveLengthValues[287]) {
                        closestWaveLength = waveLengthValues[287];
                        waveIndex = 287;
                    }
                    else {
                        for(int i=0; i<287;i++) {
                            if(waveLengthIndex >= waveLengthValues[i] && waveLengthIndex <
waveLengthValues[i+1] ) {
                                int d1 = waveLengthIndex - waveLengthValues[i];
                                int d2 = waveLengthValues[i+1] - waveLengthIndex;

                                if(d1 < d2) {
                                    closestWaveLength = waveLengthValues[i];
                                    waveIndex = i;
                                }
                                else {
                                    closestWaveLength = waveLengthValues[i + 1];
                                    waveIndex = i+1;
                                }
                            }
                        }
                    }
                    String closestWaveString = Integer.toString(closestWaveLength);
                    String absString = Integer.toString(absData[waveIndex]);
                    absSample.setText(absString);
                    imm.hideSoftInputFromWindow(waveLength.getWindowToken(), 0);
                    Toast.makeText(MainActivity.this, closestWaveString,
Toast.LENGTH_LONG).show();
                }
                else {
                    Toast toast = Toast.makeText(MainActivity.this, "300 -900사이 숫자로
입력하세요", Toast.LENGTH_SHORT);
                    toast.show();
                }
            }
            else {
                Toast toast = Toast.makeText(MainActivity.this, "3자리 숫자로 입력하세요",
Toast.LENGTH_SHORT);
                toast.show();
            }
            return false;
        }
    });

    screenShotButt.setOnClickListener(new View.OnClickListener(){
        //screenshot current screen and save to gallery
        @Override
        public void onClick(View view) {
            screenShot(findViewById(R.id.dataView));
        }
    });

    imageBtn.setOnClickListener(new View.OnClickListener()
    {
        //save data to external storage
        public void onClick(View v)
        {

            dataExtract();
```

```
        }
});

// check if application has access to external storage
isExternalStorageWriteable();

mChart1   = (LineChart)findViewById(R.id.chart1);
mChart2   = (LineChart)findViewById(R.id.chart2);
mChart3   = (LineChart)findViewById(R.id.chart3);

// enable description text
mChart1.getDescription().setEnabled(true);
mChart2.getDescription().setEnabled(true);
mChart3.getDescription().setEnabled(true);

// enable touch gestures
mChart1.setTouchEnabled(true);
mChart2.setTouchEnabled(true);
mChart3.setTouchEnabled(true);

// enable scaling and dragging
mChart1.setDragEnabled(true);
mChart1.setScaleEnabled(true);
mChart1.setDrawGridBackground(false);

mChart2.setDragEnabled(true);
mChart2.setScaleEnabled(true);
mChart2.setDrawGridBackground(false);

mChart3.setDragEnabled(true);
mChart3.setScaleEnabled(true);
mChart3.setDrawGridBackground(false);

// if disabled, scaling can be done on x- and y-axis separately
mChart1.setPinchZoom(true);
mChart2.setPinchZoom(true);
mChart3.setPinchZoom(true);

// set an alternative background color
mChart1.setBackgroundColor(Color.WHITE);
mChart2.setBackgroundColor(Color.WHITE);
mChart3.setBackgroundColor(Color.WHITE);

LineData data1 = new LineData();
data1.setValueTextColor(Color.WHITE);

LineData data2 = new LineData();
data2.setValueTextColor(Color.WHITE);

LineData data3 = new LineData();
data3.setValueTextColor(Color.WHITE);

// add empty data
mChart1.setData(data1);
mChart2.setData(data2);
mChart3.setData(data3);

// get the legend (only possible after setting data)
Legend l1 = mChart1.getLegend();
Legend l2 = mChart2.getLegend();
Legend l3 = mChart3.getLegend();
```

```
l1.setEnabled(false);
l2.setEnabled(false);
l3.setEnabled(false);

// set y axis settings
YAxis leftAxis1 = mChart1.getAxisLeft();
leftAxis1.setTextColor(Color.BLACK);
leftAxis1.setDrawGridLines(false);
leftAxis1.setAxisMaximum(1000f);
leftAxis1.setAxisMinimum(0f);

YAxis leftAxis2 = mChart2.getAxisLeft();
leftAxis2.setTextColor(Color.BLACK);
leftAxis2.setDrawGridLines(false);
leftAxis2.setAxisMaximum(1000f);
leftAxis2.setAxisMinimum(0f);

YAxis leftAxis3 = mChart3.getAxisLeft();
leftAxis3.setTextColor(Color.BLACK);
leftAxis3.setDrawGridLines(false);
leftAxis3.setAxisMaximum(600f);
leftAxis3.setAxisMinimum(0f);

YAxis rightAxis1 = mChart1.getAxisRight();
rightAxis1.setEnabled(false);

YAxis rightAxis2 = mChart2.getAxisRight();
rightAxis2.setEnabled(false);

YAxis rightAxis3 = mChart3.getAxisRight();
rightAxis3.setEnabled(false);

// set x axis settings
XAxis xl1 = mChart1.getXAxis();
xl1.setDrawGridLines(false);
xl1.setPosition(XAxis.XAxisPosition.BOTTOM);
//xl1.setGranularityEnabled(false);
//xl1.setLabelCount(288);

xl1.setTextColor(Color.BLACK);
xl1.setDrawGridLines(false);
xl1.setAvoidFirstLastClipping(true);
xl1.disableGridDashedLine();
xl1.setEnabled(true);

XAxis xl2 = mChart2.getXAxis();
xl2.setDrawGridLines(false);
xl2.setPosition(XAxis.XAxisPosition.BOTTOM);
//xl2.setGranularityEnabled(false);
//xl2.setLabelCount(288);

xl2.setTextColor(Color.BLACK);
xl2.setDrawGridLines(false);
xl2.setAvoidFirstLastClipping(true);
xl2.setEnabled(true);

XAxis xl3 = mChart3.getXAxis();
xl3.setDrawGridLines(false);
xl3.setPosition(XAxis.XAxisPosition.BOTTOM);
//xl3.setGranularityEnabled(false);
//xl3.setLabelCount(288);
```

```
        xl3.setTextColor(Color.BLACK);
        xl3.setDrawGridLines(false);
        xl3.setAvoidFirstLastClipping(true);
        xl3.setEnabled(true);

        xl1.setValueFormatter(new MyXAxisValueFormatter(waveValues));                    //
setValueFormatter
        xl2.setValueFormatter(new MyXAxisValueFormatter(waveValues));                    //
setValueFormatter
        xl3.setValueFormatter(new MyXAxisValueFormatter(waveValues));                    //
setValueFormatter

        Description description1 = new Description();
        description1.setText("");

        Description description2 = new Description();
        description2.setText("");

        Description description3 = new Description();
        description3.setText("");

        mChart1.setDescription(description1);
        mChart1.getAxisLeft().setDrawGridLines(false);
        mChart1.getXAxis().setDrawGridLines(false);
        mChart1.setDrawBorders(true);

        mChart2.setDescription(description2);
        mChart2.getAxisLeft().setDrawGridLines(false);
        mChart2.getXAxis().setDrawGridLines(false);
        mChart2.setDrawBorders(true);

        mChart3.setDescription(description3);
        mChart3.getAxisLeft().setDrawGridLines(false);
        mChart3.getXAxis().setDrawGridLines(false);
        mChart3.setDrawBorders(true);

        feedMultiple();

        // enable Bluetooth
        checkBluetooth();
    }

    void screenShot(View view) {

        Date now = new Date();
        android.text.format.DateFormat.format("yyyy-MM-dd_hh:mm:ss", now);

        Bitmap bitmap = Falcon.takeScreenshotBitmap(MainActivity.this);
        try {
            String mPath = getExternalFilesDir(null) + "/" + now + ".jpg";


            File imageFile = new File(mPath);

            FileOutputStream outputStream = new FileOutputStream(imageFile);
            int quality = 100;
            bitmap.compress(Bitmap.CompressFormat.JPEG, quality, outputStream);
            outputStream.flush();
            outputStream.close();
            Toast.makeText(MainActivity.this,now+" - file saved",Toast.LENGTH_SHORT).show();
        } catch (Throwable e) {
            // Several errors may come out with file handling or DOM
```

```
            e.printStackTrace();
        }
    }

    private void isExternalStorageWriteable(){
        if(Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState()))
        {
            Log.i("State", "Yes, it is writable!");

        }
        else
         {
             Log.i("State", "NO, it is NOT writable!");
            //Toast.makeText(this, "NO EXTERNAL MEMORY", Toast.LENGTH_LONG).show();

         }
    }

    private void dataExtract()
    {
        String dataStr ="WaveLength;"+"Blank;" + "Sample;" + "Absorbance;"+"\r\n";
        int WaveLeng;

        try{

            File saveFile = new File(getExternalFilesDir(null) + "/data"); // save
            // make folder
            if(!saveFile.exists()){
                boolean success = saveFile.mkdir();
                if(!success) Toast.makeText(getApplicationContext(), "폴더를 생성할 수 없습니다.",
                        Toast.LENGTH_LONG).show();
            }
            //get current time and date for file name
            Calendar curTime = Calendar.getInstance();

            int year = curTime.get(Calendar.YEAR);
            int month = curTime.get(Calendar.MONTH) + 1;
            int day = curTime.get(Calendar.DAY_OF_MONTH);
            int hour = curTime.get(Calendar.HOUR_OF_DAY);
            int minute = curTime.get(Calendar.MINUTE);
            int second = curTime.get(Calendar.SECOND);
            String timeText = String.format("%04d.%02d.%02d_%02d:%02d:%02d_IoMT_data", year,
month, day, hour, minute, second);

            BufferedWriter buf = new BufferedWriter(new FileWriter(saveFile+"/"+timeText+".txt", true));
            buf.append(dataStr); // write file
            for (int i = 0; i < X_COUNT_MAX; i++) {
                WaveLeng = waveLengthValues[i];
                dataStr =   String.valueOf(WaveLeng)+';'+ blankData[i] + ';' + sampleData[i] + ';' +
absData[i] + ';'+"\r\n";
                buf.append(dataStr);
            }
            buf.newLine();
            buf.close();
            Toast.makeText(MainActivity.this,timeText+" - file saved",Toast.LENGTH_SHORT).show();

        }catch (FileNotFoundException e) {
            e.printStackTrace();
            Toast.makeText(MainActivity.this,"data save failed: "+e.toString(),
Toast.LENGTH_LONG).show();
        } catch (IOException e) {
            e.printStackTrace();
```

```
                Toast.makeText(MainActivity.this,"data save failed: "+e.toString(),
Toast.LENGTH_LONG).show();
            }
        }

        private void addEntry1(int event) {
            LineData data1 = mChart1.getData();
            // add data to data array
            blankData[index1] = event;
            index1++;

            Log.w("myApp", "index1 : " + index1);
            if(index1 == X_COUNT_MAX)
            {
                index1 = 0;
                Log.w("myApp", "index1 X_COUNT_MAX : " + index1);
            }

            if(index1 >= startIndex && index1 <= endIndex) {

                if (data1 != null) {
                    ILineDataSet set1 = data1.getDataSetByIndex(0);

                    if (set1 == null) {
                        set1 = createSet1();
                        // add new dataset to chart
                        data1.addDataSet(set1);
                    }

                    data1.addEntry(new Entry(set1.getEntryCount(), event), 0);
                    data1.notifyDataChanged();
                    // let the chart know it's data has changed
                    mChart1.notifyDataSetChanged();
                    // limit the number of visible entries

                    mChart1.setVisibleXRangeMaximum(endIndex - startIndex+1 );

                    // move to the latest entry
                    mChart1.moveViewToX(data1.getEntryCount());
                    mChart1.invalidate();
                }
            }
        }

        private void addEntry2(int event) {
            LineData data2 = mChart2.getData();

            sampleData[index2] = event;
            index2++;
            Log.w("myApp", "index2 : " + index2);

            if(index2 == X_COUNT_MAX) {
                index2 = 0;
            }

            if(index2 >= startIndex && index2 <= endIndex) {

                if (data2 != null) {
                    ILineDataSet set2 = data2.getDataSetByIndex(0);
                    // set.addEntry(...); // can be called as well

                    if (set2 == null) {
```

```
                set2 = createSet2();
                data2.addDataSet(set2);
            }

            data2.addEntry(new Entry(set2.getEntryCount(), event), 0);
            data2.notifyDataChanged();
            // let the chart know it's data has changed
            mChart2.notifyDataSetChanged();
            // limit the number of visible entries

            mChart2.setVisibleXRangeMaximum(endIndex - startIndex+1);
            //mChart1.setVisibleXRangeMaximum(X_COUNT_MAX);
            //mChart1.setVisibleYRange(30, AxisDependency.LEFT);

            // move to the latest entry
            mChart2.moveViewToX(data2.getEntryCount());
            mChart2.invalidate();

        }
    }
}
private void addEntry3(int event) {
    // private void addEntry3(float event) {
     LineData data3 = mChart3.getData();

     absData[index3] = event;
     index3++;

     Log.w("myApp", "index3 : " + index3);

     if(index3 == X_COUNT_MAX) {
         index3 = 0;
         int i_d = absData[103];    //565.189 nm
         int i_a = absData[152];    //668.868 nm

         double efret_val = ((double) i_a) /(((double) i_a) + ((double) i_d));

         absMax = 0;
         String efret_val_string = String.format("%.4f", efret_val);

         e_fret_text.setText(efret_val_string);
         Log.w("myApp", "X_COUNT_MAX : " + index3);
     }

     if(index3 >= startIndex && index3 <= endIndex) {

         if (data3 != null) {
             ILineDataSet set3 = data3.getDataSetByIndex(0);
             // set.addEntry(...); // can be called as well

             if (set3 == null) {
                 set3 = createSet3();
                 data3.addDataSet(set3);
             }
             //float dst = 400;
             // data3.addEntry(new Entry(dst, (float) event), 0);
             data3.addEntry(new Entry(set3.getEntryCount(), event), 0);
             data3.notifyDataChanged();
             // let the chart know it's data has changed
             mChart3.notifyDataSetChanged();
             // limit the number of visible entries
```

```
                mChart3.setVisibleXRangeMaximum(endIndex - startIndex+1);

                if(absMax < event) absMax = event;
                //mChart3.setVisibleYRangeMaximum(absMax + 50, YAxis.AxisDependency.LEFT);
                mChart3.getAxisLeft().setAxisMaximum(absMax+50);

                // move to the latest entry
                mChart3.moveViewToX(data3.getEntryCount());
                mChart3.invalidate();

            }
        }

}
// set chart style
private LineDataSet createSet1() {

    LineDataSet set1 = new LineDataSet(null, "Blank");

    set1.setAxisDependency(YAxis.AxisDependency.LEFT);
    set1.setLineWidth(2f);
    set1.setColor(Color.rgb(31,119,180));
    set1.setHighlightEnabled(false);
    set1.setDrawValues(false);
    set1.setDrawCircles(false);
    set1.setMode(LineDataSet.Mode.CUBIC_BEZIER);
    set1.setCubicIntensity(0.1f);

    return set1;
}
private LineDataSet createSet2() {

    LineDataSet set2 = new LineDataSet(null, "Sample");

    set2.setAxisDependency(YAxis.AxisDependency.LEFT);
    set2.setLineWidth(2f);
    set2.setColor(Color.rgb(31,119,180));
    set2.setHighlightEnabled(false);
    set2.setDrawValues(false);
    set2.setDrawCircles(false);
    set2.setMode(LineDataSet.Mode.CUBIC_BEZIER);
    set2.setCubicIntensity(0.1f);

    return set2;
}
private LineDataSet createSet3() {

    LineDataSet set3 = new LineDataSet(null, "Subtract");

    set3.setAxisDependency(YAxis.AxisDependency.LEFT);
    set3.setLineWidth(2f);
    set3.setColor(Color.rgb(31,119,180));
    set3.setHighlightEnabled(false);
    set3.setDrawValues(false);
    set3.setDrawCircles(false);
    set3.setMode(LineDataSet.Mode.CUBIC_BEZIER);
    set3.setCubicIntensity(0.1f);
    return set3;
}

private void feedMultiple() {
    if (thread != null){
```

```
            thread.interrupt();
        }
        thread = new Thread(new Runnable() {

            @Override
            public void run() {
                while (true){
                    plotData = true;
                    try {
                        Thread.sleep(10);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        });
        thread.start();
    }

    public class MyXAxisValueFormatter extends ValueFormatter {

        private String[] mValues;

        public MyXAxisValueFormatter(String[] values) {
            this.mValues = values;
        }

        @Override
        public String getAxisLabel(float value, AxisBase axis)
        {
            // "value" represents the position of the label on the axis (x or y)
            if ((int)value >= 0 && (int)value <= endIndex - startIndex - 1) {
                return mValues[(int)value + startIndex];
            } else {
                return "";
            }
        }

        /** this is only needed if numbers are returned, else return 0 */
        // @Override
        //    public int getDecimalDigits() { return 0; }
    }


    // find paired Bluetooth devices of given name
    BluetoothDevice getDeviceFromBondedList(String name) {
        // BluetoothDevice : get list of paired devices
        BluetoothDevice selectedDevice = null;

        for(BluetoothDevice deivce : mDevices) {
            // getName() : return Bluetooth Adapter name
            if(name.equals(deivce.getName())) {
                selectedDevice = deivce;
                break;
            }
        }
        return selectedDevice;
    }


    // send string data to Arduino
    void sendData(String msg) {
```

```
        msg += mStrDelimiter;   // add \n
        try {
            // getBytes() : String to byte

            mOutputStream.write(msg.getBytes());    // send string
        }catch(Exception e) {   // error while sending string
            Toast.makeText(getApplicationContext(), "데이터 전송중 오류가 발생",
                    Toast.LENGTH_LONG).show();
        }
    }

    // connectToSelectedDevice() : connect to remote device
    void connectToSelectedDevice(String selectedDeviceName) {

        mRemoteDevie = getDeviceFromBondedList(selectedDeviceName);
        UUID uuid = java.util.UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");

        try {
            // create socket, connect via RFCOMM channel
            // createRfcommSocketToServiceRecord(uuid) : create communication socket with uuid

            mSocket = mRemoteDevie.createRfcommSocketToServiceRecord(uuid);
            mSocket.connect(); // connect between two devices through socket

            // BluetoothSocket object has two streams
            // 1. OutputStrem: send data
            // 2. InputStream: get data
            mOutputStream = mSocket.getOutputStream();
            mInputStream = mSocket.getInputStream();

            // ready for sending data
            beginListenForData();

        }catch(Exception e) { // Bluetooth connection error
            Toast.makeText(getApplicationContext(),
                    "블루투스 연결 중 오류가 발생했습니다.", Toast.LENGTH_LONG).show();
        }
    }


    // check if data received from Arduino in thread
    void beginListenForData() {
        final Handler handler = new Handler();

        readBufferPosition = 0;
        readBuffer = new byte[1024];


        // get data from Arduino
        mWorkerThread = new Thread(new Runnable()
        {
            @Override
            public void run() {


                while(!Thread.currentThread().isInterrupted()) {
                    try {
                        int byteAvailable = mInputStream.available();     // received data exists

                        if(byteAvailable > 0) {                                 // data received
                            //          Log.e("data" , byteAvailable+"");
```

```
                                byte[] packetBytes = new byte[byteAvailable];
                                mInputStream.read(packetBytes);
                                for(int i=0; i<byteAvailable; i++) {
                                      byte b = packetBytes[i];
                                      if(b == mCharDelimiter) {
                                            byte[] encodedBytes = new byte[readBufferPosition];

                                            System.arraycopy(readBuffer, 0, encodedBytes, 0,
encodedBytes.length);

                                            final String data = new String(encodedBytes, "US-ASCII");

                                            readBufferPosition = 0;
                                            handler.post(new Runnable(){
                                                  // received data processing
                                                  @Override
                                                  public void run() {
                                                        //blank data header
                                                        if( data.toCharArray()[0] =='a')
                                                        {
                                                              classfy = 1;
                                                              //String subone = data.substring(1,data.length());
                                                              result1 = Integer.parseInt(data.replaceAll("[^0-9]",
""));

                                                            Log.w("myApp","result1 : "+result1);
                                                              //mChart1.invalidate();
                                                              //mChart1.clear();
                                                              // add data to array and chart
                                                              addEntry1(result1);
                                                        }
                                                        //sample data header
                                                        if( data.toCharArray()[0] =='b')
                                                        {
                                                              classfy = 2;
                                                              //String subone = data.substring(1,data.length());
                                                              result2 = Integer.parseInt(data.replaceAll("[^0-9]",
""));

                                                            Log.w("myApp","result2 : "+result2);
                                                              //mChart1.invalidate();
                                                              //mChart1.clear();

                                                                addEntry2(result2);
                                                        }
                                                        // subtract data header
                                                        if( data.toCharArray()[0] =='c')
                                                        {
                                                              classfy = 3;
                                                              //String subone = data.substring(1,data.length());

                                                              result3 = Integer.parseInt(data.replaceAll("[^0-9]",
""));
                                                              Log.w("myApp","result3 : "+result3);

                                                              addEntry3(result3);
                                                        }

                                                  }
                                            });
                                      }
                                      else {
```

```
                                    readBuffer[readBufferPosition++] = b;
                                }
                            }
                        }

                    } catch (Exception e) {       // error
                        //Toast.makeText(getApplicationContext(), "데이터 수신 중 오류가 발생
했습니다.", Toast.LENGTH_LONG).show();
                        // finish();                 // Exit App
                    }
                }
            }

        });
        mWorkerThread.start();
    }

    // select Arduino device at start page
    void selectDevice() {

        mDevices = mBluetoothAdapter.getBondedDevices();
        mPariedDeviceCount = mDevices.size();

        if(mPariedDeviceCount == 0 ) { // no paired device
            Toast.makeText(getApplicationContext(), "페어링된 장치가 없습니다.",
Toast.LENGTH_LONG).show();
            //finish(); // App 종료.
        }
        // paired device exists
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("블루투스 장치 선택");

        // show list of paired devices
        List<String> listItems = new ArrayList<String>();
        for(BluetoothDevice device : mDevices) {
            // device.getName() : return device name
            listItems.add(device.getName());
        }
        listItems.add("취소");    // cancel

        final CharSequence[] items = listItems.toArray(new CharSequence[listItems.size()]);
        listItems.toArray(new CharSequence[listItems.size()]);

        builder.setItems(items, new DialogInterface.OnClickListener() {

            @Override
            public void onClick(DialogInterface dialog, int item) {
                // TODO Auto-generated method stub
                if(item == mPariedDeviceCount) { // select cancel
                    Toast.makeText(getApplicationContext(), "연결할 장치를 선택하지 않았습니다.",
Toast.LENGTH_LONG).show();
                    //finish();
                }
                else { // if certain device is selected, try to connect
                    connectToSelectedDevice(items[item].toString());
                }
            }

        });
```

```
        builder.setCancelable(false);    // disable go-back button
        AlertDialog alert = builder.create();
        alert.show();
    }

    void checkBluetooth() {

        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        if(mBluetoothAdapter == null ) {    // cannot use Bluetooth
            Toast.makeText(getApplicationContext(), "기기가 블루투스를 지원하지 않습니다.",
Toast.LENGTH_LONG).show();
        }
        else { // can use Bluetooth

            if(!mBluetoothAdapter.isEnabled()) { // Bluetooth state is disabled
                Toast.makeText(getApplicationContext(), "현재 블루투스가 비활성 상태입니다.",
Toast.LENGTH_LONG).show();
                Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);

                startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
            }
            else // Bluetooth state is enabled
                selectDevice();
        }
    }


    @Override
    protected void onDestroy() {
        try{
            mWorkerThread.interrupt();
            mInputStream.close();
            mSocket.close();
        }catch(Exception e){}
        super.onDestroy();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {

        switch(requestCode) {
            case REQUEST_ENABLE_BT:
                if(resultCode == RESULT_OK) { // check Bluetooth state
                    selectDevice();
                }
                else if(resultCode == RESULT_CANCELED) { // Bluetooth disabled
                    Toast.makeText(getApplicationContext(), "블루투수를 사용할 수 없어 프로그램을
종료합니다",
                            Toast.LENGTH_LONG).show();
                }
                break;
        }
        super.onActivityResult(requestCode, resultCode, data);
    }
}
```