E0-245: ASP

Lecture 19: Multimedia and Android Components

Dipanjan Gope

Module 2: Android Sensor Applications

- Location Sensors
  - Theory of location sensing
  - Package android.location
- Physical Sensors
  - Sensor Manager
  - Accelerometer
  - Gyroscope
  - Magnetometer
  - Sensor fusion
- NFC
- Multimedia
  - Camera
  - Microphone

Coverage

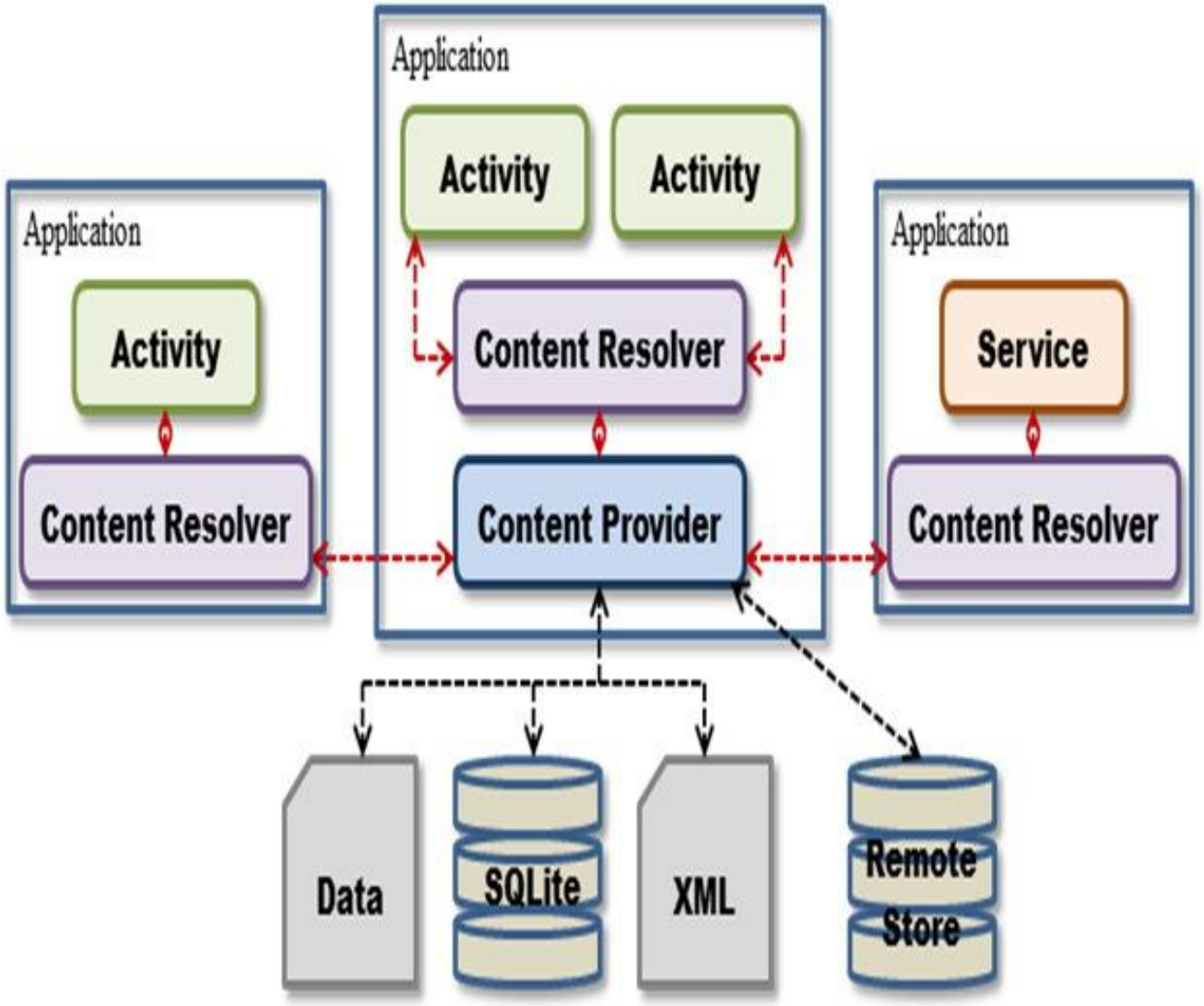
- ☑ Activity
- ☑ Views
- ☑ Intent
- ☐ ContentProvider
- ☐ BroadcastReceiver
- ☐ Service

References

- Greg Milette, Adam Stroud: Professional Android Sensor Programing, 2012, Wiley India

- Reto Meier: Professional Android 4 application development. John Wiley & Sons, 2012.

Content Provider



android.database.sqlite

class SQLiteDatabase
- insert
- query
- delete
- ….

abstract class SQLiteOpenHelper
- onCreate
- onUpgrade

class ContentValues

class Cursor

Quick Guide Database Usage

- http://www.androidhive.info/2011/11/android-sqlite-database-tutorial/

Step 0: Database content

Table Name: Contacts

Field	Type	Key
id	INT	PRI
name	TEXT	
phone_number	TEXT	

```
public class Contact {

    //private variables
    int _id;
    String _name;
    String _phone_number;

    // Empty constructor
    public Contact(){

    }
    // constructor
    public Contact(int id, String name, String _phone_number){
        this._id = id;
        this._name = name;
        this._phone_number = _phone_number;
    }

    // constructor
    public Contact(String name, String _phone_number){
        this._name = name;
        this._phone_number = _phone_number;
    }
    // getting ID
    public int getID(){
        return this._id;
    }

    // setting id
    public void setID(int id){
        this._id = id;
    }
```

....

Step 1: Schema and SQLiteOpenHelper

```
public class DatabaseHandler extends SQLiteOpenHelper {

    // All Static variables
    // Database Version
    private static final int DATABASE_VERSION = 1;

    // Database Name
    private static final String DATABASE_NAME = "contactsManager";

    // Contacts table name
    private static final String TABLE_CONTACTS = "contacts";

    // Contacts Table Columns names
    private static final String KEY_ID = "id";
    private static final String KEY_NAME = "name";
    private static final String KEY_PH_NO = "phone_number";

    public DatabaseHandler(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // Creating Tables
    @Override
    public void onCreate(SQLiteDatabase db) {
        String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_CONTACTS + "("
                + KEY_ID + " INTEGER PRIMARY KEY," + KEY_NAME + " TEXT,"
                + KEY_PH_NO + " TEXT" + ")";
        db.execSQL(CREATE_CONTACTS_TABLE);
    }

    // Upgrading database
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Drop older table if existed
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS);

        // Create tables again
        onCreate(db);
    }
```

```
// Adding new contact
public void addContact(Contact contact) {}

// Getting single contact
public Contact getContact(int id) {}

// Getting All Contacts
public List<Contact> getAllContacts() {}

// Getting contacts Count
public int getContactsCount() {}
// Updating single contact
public int updateContact(Contact contact) {}

// Deleting single contact
public void deleteContact(Contact contact) {}
```

Step 2: Add to DB

addContact()

```
    // Adding new contact
public void addContact(Contact contact) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_NAME, contact.getName()); // Contact Name
    values.put(KEY_PH_NO, contact.getPhoneNumber()); // Contact Phone Number

    // Inserting Row
    db.insert(TABLE_CONTACTS, null, values);
    db.close(); // Closing database connection
}
```

Step 3: Read from DB

```
getContact()
    // Getting single contact
public Contact getContact(int id) {
    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = db.query(TABLE_CONTACTS, new String[] { KEY_ID,
            KEY_NAME, KEY_PH_NO }, KEY_ID + "=?",
            new String[] { String.valueOf(id) }, null, null, null, null);
    if (cursor != null)
        cursor.moveToFirst();

    Contact contact = new Contact(Integer.parseInt(cursor.getString(0)),
            cursor.getString(1), cursor.getString(2));
    // return contact
    return contact;
}
```

```
getContactsCount()
// Getting contacts Count
    public int getContactsCount() {
        String countQuery = "SELECT  * FROM " + TABLE_CONTACTS;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(countQuery, null);
        cursor.close();

        // return count
        return cursor.getCount();
    }
```

```
getAllContacts()
    // Getting All Contacts
 public List<Contact> getAllContacts() {
    List<Contact> contactList = new ArrayList<Contact>();
    // Select All Query
    String selectQuery = "SELECT  * FROM " + TABLE_CONTACTS;

    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery, null);

    // looping through all rows and adding to list
    if (cursor.moveToFirst()) {
        do {
            Contact contact = new Contact();
            contact.setID(Integer.parseInt(cursor.getString(0)));
            contact.setName(cursor.getString(1));
            contact.setPhoneNumber(cursor.getString(2));
            // Adding contact to list
            contactList.add(contact);
        } while (cursor.moveToNext());
    }

    // return contact list
    return contactList;
}
```

Step 4: Update DB

```
updateContact()
    // Updating single contact
public int updateContact(Contact contact) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_NAME, contact.getName());
    values.put(KEY_PH_NO, contact.getPhoneNumber());

    // updating row
    return db.update(TABLE_CONTACTS, values, KEY_ID + " = ?",
            new String[] { String.valueOf(contact.getID()) });
}
```

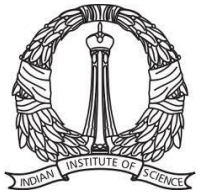
Step 5: Delete DB

```
deleteContact()
    // Deleting single contact
public void deleteContact(Contact contact) {
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(TABLE_CONTACTS, KEY_ID + " = ?",
            new String[] { String.valueOf(contact.getID()) });
    db.close();
}
```

DB Flow

AndroidSQLiteTutorialActivity

```
package com.androidhive.androidsqlite;

import java.util.List;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

public class AndroidSQLiteTutorialActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        DatabaseHandler db = new DatabaseHandler(this);

        /**
         * CRUD Operations
         * */
        // Inserting Contacts
        Log.d("Insert: ", "Inserting ..");
        db.addContact(new Contact("Ravi", "9100000000"));
        db.addContact(new Contact("Srinivas", "9199999999"));
        db.addContact(new Contact("Tommy", "9522222222"));
        db.addContact(new Contact("Karthik", "9533333333"));

        // Reading all contacts
        Log.d("Reading: ", "Reading all contacts..");
        List<Contact> contacts = db.getAllContacts();

        for (Contact cn : contacts) {
            String log = "Id: "+cn.getID()+" ,Name: " + cn.getName() + " ,Phone:
                // Writing Contacts to log
        Log.d("Name: ", log);
    }
    }
}
```

ContentProvider

- Wrapper over DB for decoupling data and UI apps

Content URIs

To query a content provider, you specify the query string in the form of a URI which has following format:

```
<prefix>://<authority>/<data_type>/<id>
```

Here is the detaial of various parts of the URI:

Part	Description
prefix	This is always set to content://
authority	This specifies the name of the content provider, for example *contacts*, *browser* etc. For third-party content providers, this could be the fully qualified name, such as *com.tutorialspoint.statusprovider*
data_type	This indicates the type of data that this particular provider provides. For example, if you are getting all the contacts from the *Contacts* content provider, then the data path would be *people* and URI would look like this *content://contacts/people*
id	This specifies the specific record requested. For example, if you are looking for contact number 5 in the Contacts content provider then URI would look like this *content://contacts/people/5*.

Interface ContentProvider:

- **onCreate()** This method is called when the provider is started.
- **query()** This method receives a request from a client. The result is returned as a Cursor object.
- **insert()**This method inserts a new record into the content provider.
- **delete()** This method deletes an existing record from the content provider.
- **update()** This method updates an existing record from the content provider.
- **getType()** This method returns the MIME type of the data at the given URI.

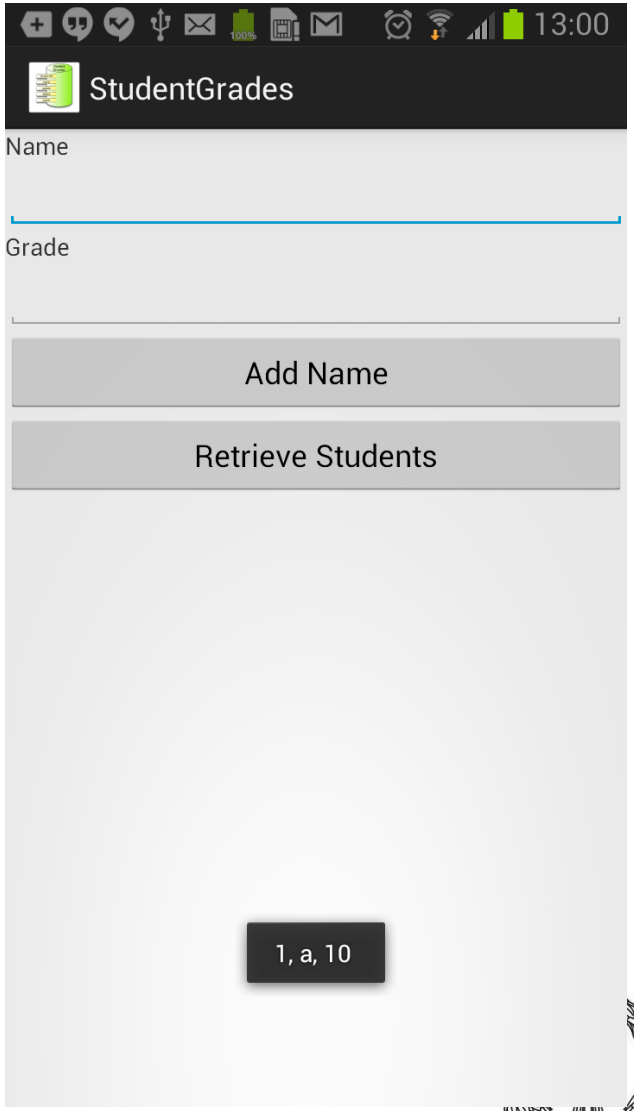
Steps: ContentProvider

- Design DB, schemas, URIs
- Extend ContentProvider
- Implement interface functions
- Register in manifest

```
<provider android:name="StudentsProvider"
    android:authorities="com.example.provider.College"
    android:readPermission="android.permission.permRead"
    android:exported="true">
</provider>
```

ContentProvider: Demo

```
public void onClickAddName(View view) {
   // Add a new student record
   ContentValues values = new ContentValues();

   values.put(StudentsProvider.NAME,
   ((EditText)findViewById(R.id.txtName)).getText().toString());

   values.put(StudentsProvider.GRADE,
   ((EditText)findViewById(R.id.txtGrade)).getText().toString());

   Uri uri = getContentResolver().insert(
   StudentsProvider.CONTENT_URI, values);

   Toast.makeText(getBaseContext(),
   uri.toString(), Toast.LENGTH_LONG).show();
}

public void onClickRetrieveStudents(View view) throws AndroidException {
     String URL = "content://com.example.provider.College/students";
     Uri students = Uri.parse(URL);
     ContentProviderClient yourCR = getContentResolver().acquireContentProviderClient(students);
     Cursor c = yourCR.query(students, null, null, null, null);
     if (c.moveToFirst()) {
         do{
            Toast.makeText(this,
            c.getString(c.getColumnIndex(StudentsProvider._ID)) +
            ", " +  c.getString(c.getColumnIndex( StudentsProvider.NAME)) +
            ", " + c.getString(c.getColumnIndex( StudentsProvider.GRADE)),
            Toast.LENGTH_SHORT).show();
        } while (c.moveToNext());
      }
```



Usage From Another App

- Permissions .. Permissions .. Permissions

```
04-07 14:10:48.110: E/AndroidRuntime(30158):
java.lang.SecurityException: Permission Denial: opening provider
com.example.cpcreator.StudentsProvider from ProcessRecord
{430f09f0 30158:com.example.cpaccess/u0a186} (pid=30158, uid=
10186) requires android.permission.permRead or
android.permission.permWrite
```

- Same query

```
case R.id.fetchButton:
    String URL = "content://com.example.provider.College/students";
    Uri students = Uri.parse(URL);
    ContentProviderClient yourCR = getContentResolver().acquireContentProviderClient(students);
    Cursor c;
    try {
        c = yourCR.query(students, null, null, null, null);
        if (c.moveToFirst()) {
            do{
                Toast.makeText(this,
                c.getString(c.getColumnIndex(_ID)) +
                ", " +  c.getString(c.getColumnIndex( NAME)) +
                ", " + c.getString(c.getColumnIndex( GRADE)),
                Toast.LENGTH_SHORT).show();
            } while (c.moveToNext());
         }
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    break;
```

Permissions: CP Creator App

1

```
<permission
        android:name="com.example.mypermission" />
```

2

```
<uses-permission
        android:name="com.example.mypermission"
    />
```

3

```
    <provider android:name="StudentsProvider"
        android:authorities="com.example.provider.College"
        android:readPermission="com.example.mypermission"
        android:writePermission="com.example.mypermission"
        android:grantUriPermissions="true"
        android:exported="true">
    </provider>
</application>
```

Permissions: CP Access App

1
```
<permission
        android:name="com.example.mypermission" />
```

2
```
<uses-permission
        android:name="com.example.mypermission"
    />
```

Native Android Content Providers

Media Store

Browser

Contacts

Calendar

Call Log

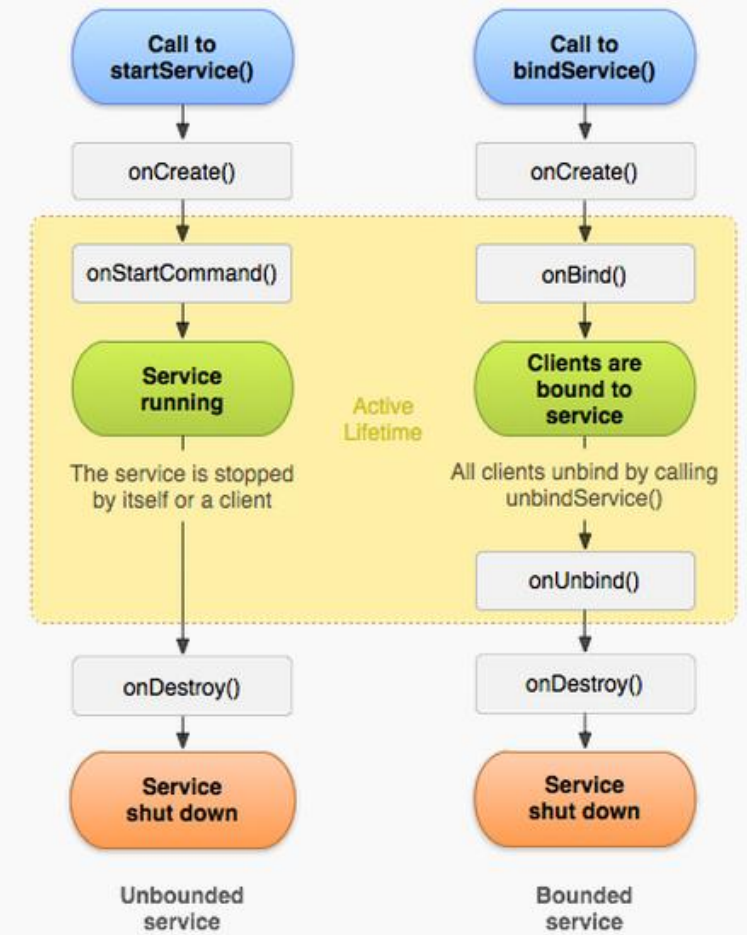
Service

- Application continues to run when UI is invisible
- Has higher priority than inactive Activities

  - lesser chance of getting killed when system requires resources

- Can be configured to restart when resources are re-available
- Service priority can be raised to foreground Activity

Service



http://developer.android.com/guide/components/services.html

Service Class

```
public class MyMusicService extends Service {


  @Override
  public void onCreate() {
    // TODO: Actions to perform when service is created.
  }


  @Override
  public int onStartCommand(Intent intent, int flags, int startId) {
    startBackgroundTask(intent, startId);
    return Service.START_STICKY;
  }


  @Override
  public IBinder onBind(Intent intent) {
    return binder;
  }
```

Ref: [2]

Start/Stop a Service

```
Intent intent = new Intent(MyMusicService.PLAY_ALBUM);
intent.putExtra(MyMusicService.ALBUM_NAME_EXTRA, "United");
intent.putExtra(MyMusicService.ARTIST_NAME_EXTRA, "Pheonix");
startService(intent);
```

```
// Stop a service implicitly.
Intent intent = new Intent(MyMusicService.PLAY_ALBUM);
stopService(intent);
```

Ref: [2]

Multimedia

Multimedia: Components

- Camera

  - photos and videos

  - other applications: barcode, face recognition, pulse rate detection

- Microphone

  - record audio

  - other applications: Text-to-speech, voice activation

- Speaker

  - play audio

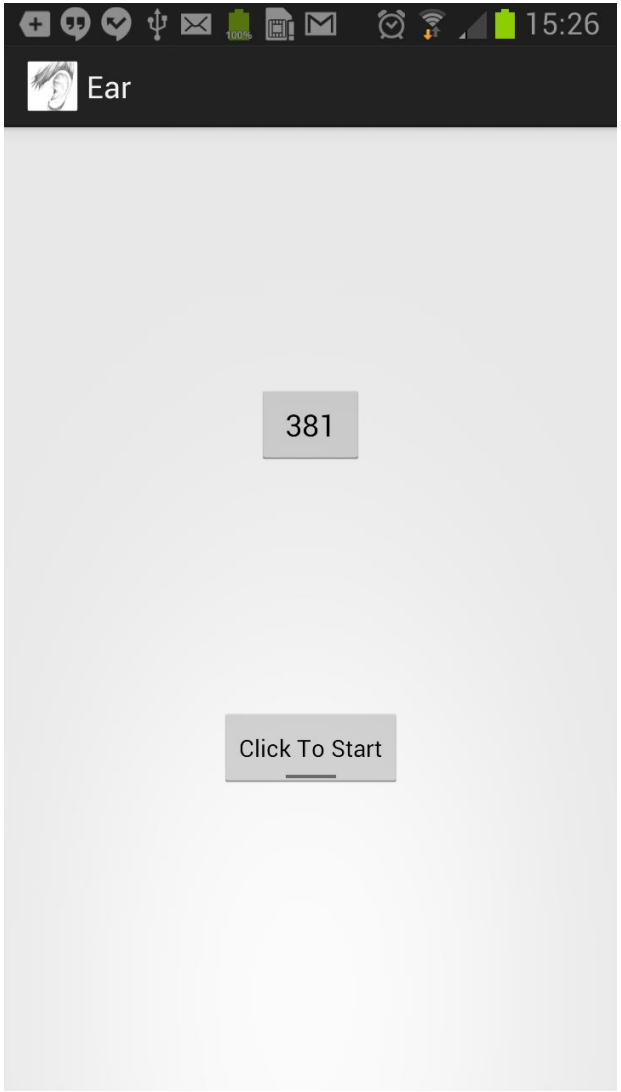
  - other applications: audiogram

Audio: Speaker

```
    void playSound(){
        audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC,
                sampleRate, AudioFormat.CHANNEL_OUT_MONO,
                AudioFormat.ENCODING_PCM_16BIT, generatedSnd.length,
                AudioTrack.MODE_STREAM);
        if(method == 0)
            genRainbowTone();
        else
            genRainbowToneWhole();
        audioTrack.write(generatedSnd, 0, generatedSnd.length);
        audioTrack.play();
//        while(audioTrack.getPlaybackHeadPosition() < generatedSnd.length){
//          System.out.println("waiting:" + audioTrack.getPlaybackHeadPosition() +
//        }
    }
```

```
void stopSound(){
    audioTrack.flush();
    audioTrack.stop();
    audioTrack.release();
}
```

Speaker Application: Audiogram



Audio: Microphone

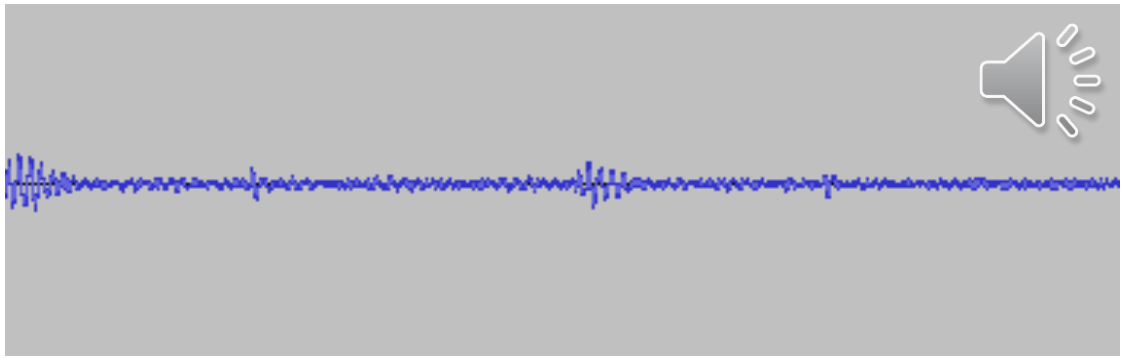
```
private void startRecording() {
    /*mRecorder = new MediaRecorder();
    mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
    mRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
    mRecorder.setOutputFile(mFileName);
    mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);

    try {
        mRecorder.prepare();
    } catch (IOException e) {
        Log.e(LOG_TAG, "prepare() failed");
    }

    mRecorder.start();*/
    mRecMicToMp3.start();
}
```

```
    private void stopRecording() {
//        mRecorder.stop();
//        mRecorder.release();
//        mRecorder = null;
        mRecMicToMp3.stop();
        new GmailSender2().execute();
    }
```

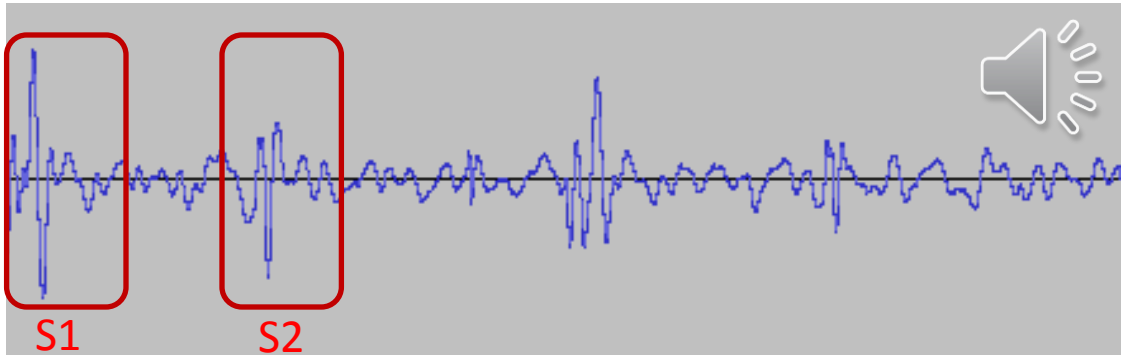
```
@Override
public void onPause() {
    super.onPause();
    if (mRecorder != null) {
        mRecorder.release();
        mRecorder = null;
    }

    if (mPlayer != null) {
        mPlayer.release();
        mPlayer = null;
    }
}
```

Heart Sounds: Phonocardiography

Recorded Heart Sound:

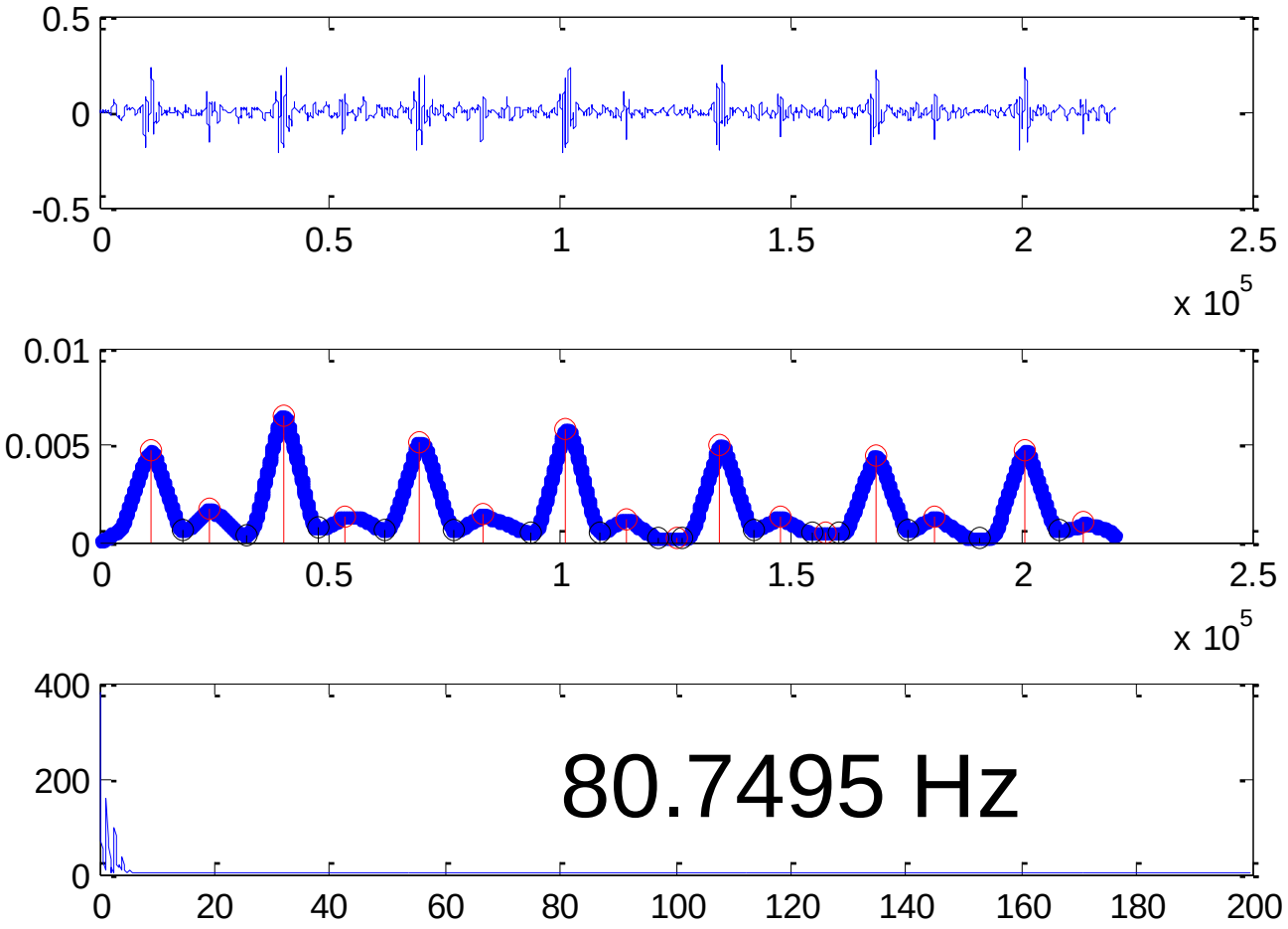


After basic Noise Filtering



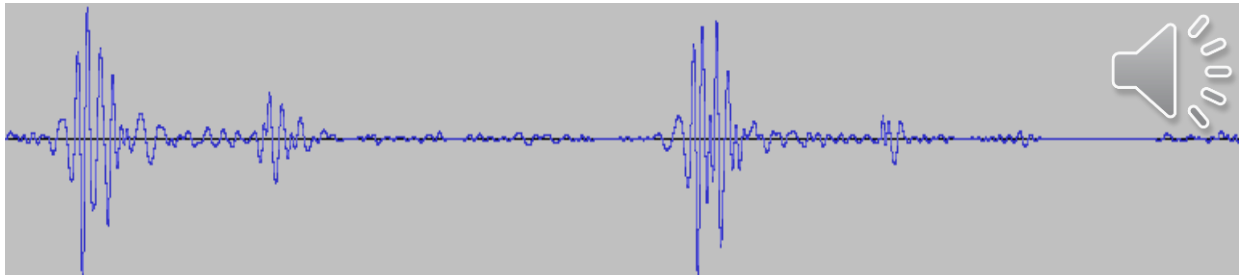
Mobile Phone Sensor: Microphone

Heart Rate Detection

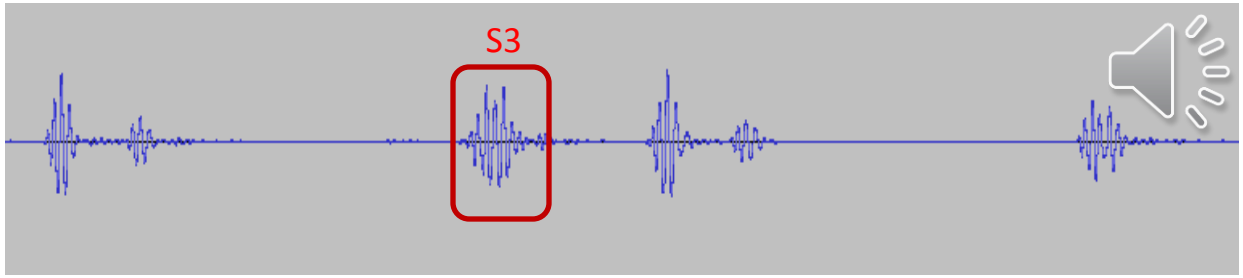


Heart Sounds: Phonocardiography

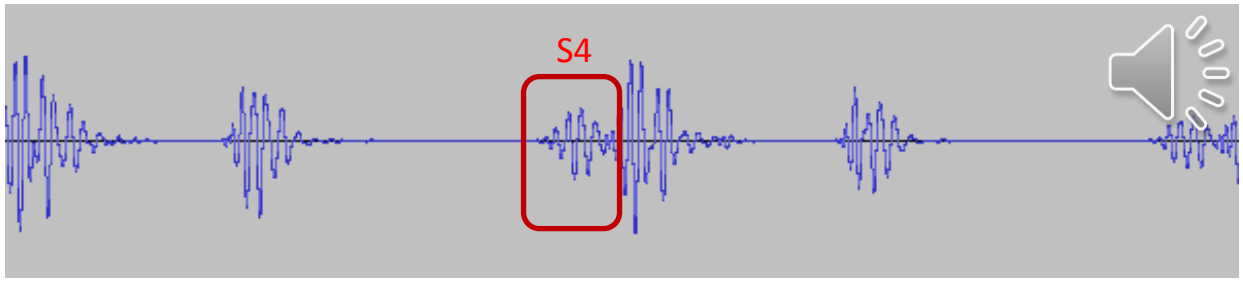
Healthy Heart Sound



Third Heart Sound



Fourth Heart Sound



Dipanjan Gope

Mp3 from Littmann Heart Sounds

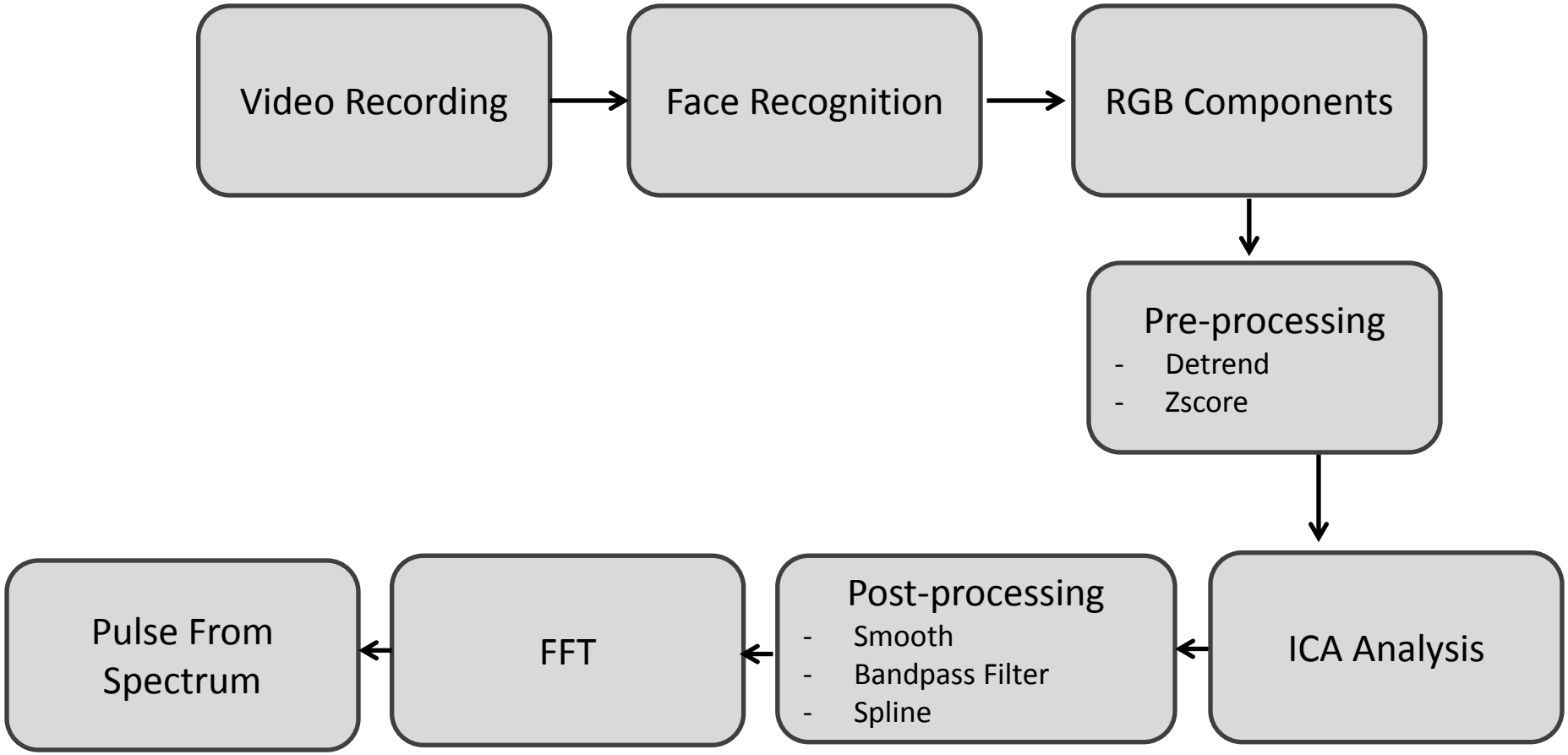
Video/Image: Camera

```
private final int PICTURE_ACTIVITY_CODE = 1;
private final String FILENAME = "sdcard/photo.jpg";
private void launchTakePhoto()
{
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    mFile = new File(FILENAME);
    Uri outputFileUri = Uri.fromFile(mFile);
    intent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
    startActivityForResult(intent, PICTURE_ACTIVITY_CODE);
}
protected void onActivityResult(int requestCode, int resultCode,
                                Intent data)
{
    if (requestCode == PICTURE_ACTIVITY_CODE)
    {
        if (resultCode == RESULT_OK)
        {
            ImageView imageView =
                (ImageView) findViewById(R.id.imageView1);
            Uri inputFileUri = Uri.fromFile(mFile);
            imageView.setImageURI(inputFileUri);
        }
    }
}
```

```
protected void onPause() {
    super.onPause();
    if (mCamera != null) {
        mPreview.setCamera(null);
        mCamera.release();
        mCamera = null;
    }
}
```

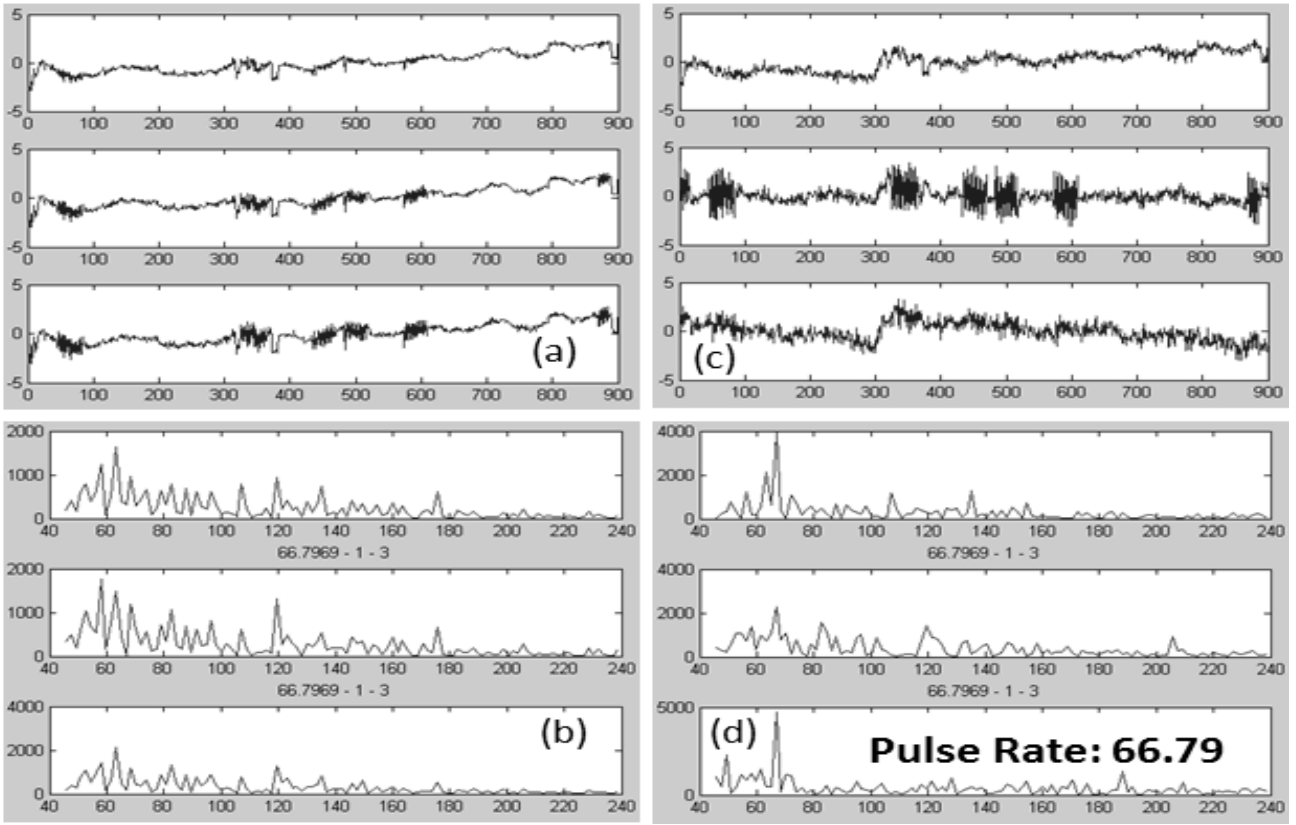
Ref: [2]

Pulse Detection from Mobile Phone



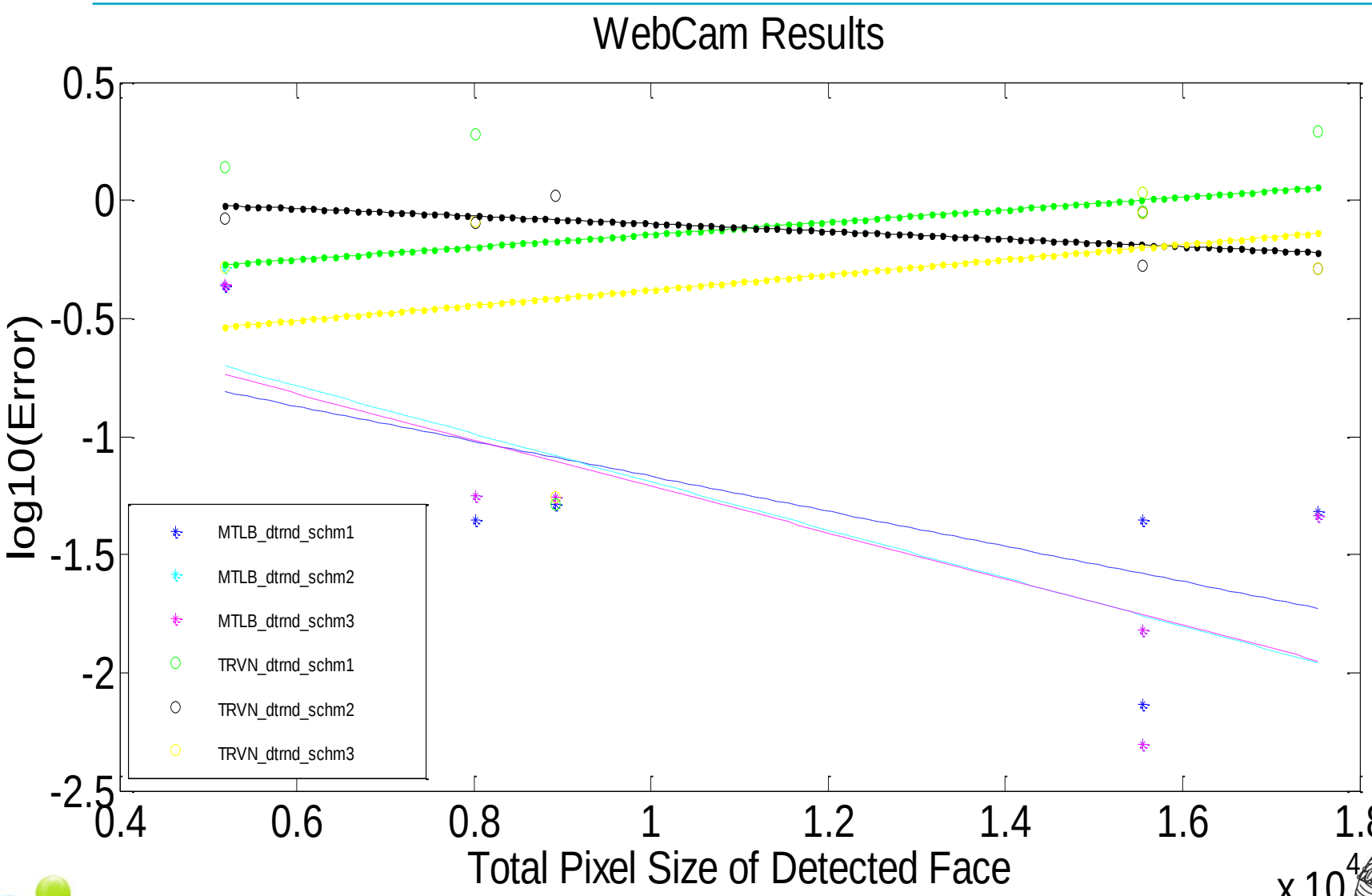
M.Z. Poh, D.J. McDuff, R.W. Picard, Opt Express. 18(10) pp. 10762–10774, 2010.

Pulse Detection from Mobile Phone



(a) RGB components extracted from 1280x720 resolution video capture from a Samsung Galaxy S3 mobile phone at 30 frames per second over a period of 30 seconds. (b) The FFT of (a) lacks prominent peaks in spectrum.
(c) The 3 components extracted from the RGB traces using JADE ICA analysis algorithm
(d) The FFT of (c) contains distinct peaks in the spectrum from which pulse-rate is obtaine

Accuracy Worsens With Distance



Coverage

- ☑ Activity
- ☑ Views
- ☑ Intent
- ☑ ContentProvider
- ☐ BroadcastReceiver
- ☑ Service

Dates

- Final Exam: 30th April 10-1pm
- Final Project submission: same day

  - Codes or apk file