EESM5060 Embedded Systems

LAB #5

Applications with Bluetooth Low Energy Glance Sport Kit

ECE-EESM5060-LAB5
By Prof. Chi Ying Tsui

Version 1.0
SEPT 4, 2023

Electronic & Computer Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

OBJECTIVE

The objective of this lab is to develop application using commercial sports product, Glance. Glance is a BLE wearable used for motion analysis and movement feature extraction. The CWB Sports Kit series is a combination of hardware and apps library, developed based on our motion analysis and feature extraction core. The small sensor device collects motion data which can be further interpret into useful information for sports, health and rehabilitation studies. For the golf application, similar device can be used to capture swing data, reconstructs the path. Partners can then use this information to review club head speed, consistency, face angle in different reply speed and viewing angle.

Getting XYZ gravity values from our phone and it can interpret as walking steps. With BLE connectivity and by constructing bonding pairs, temperature profile can be used to send temperature values to our phone. In this lab, we use glance wearable sports kit which provide a comprehensive libraries and APIs which allow more precise algorithm and application development.

This sports kit software contains a list of pre-built commands:

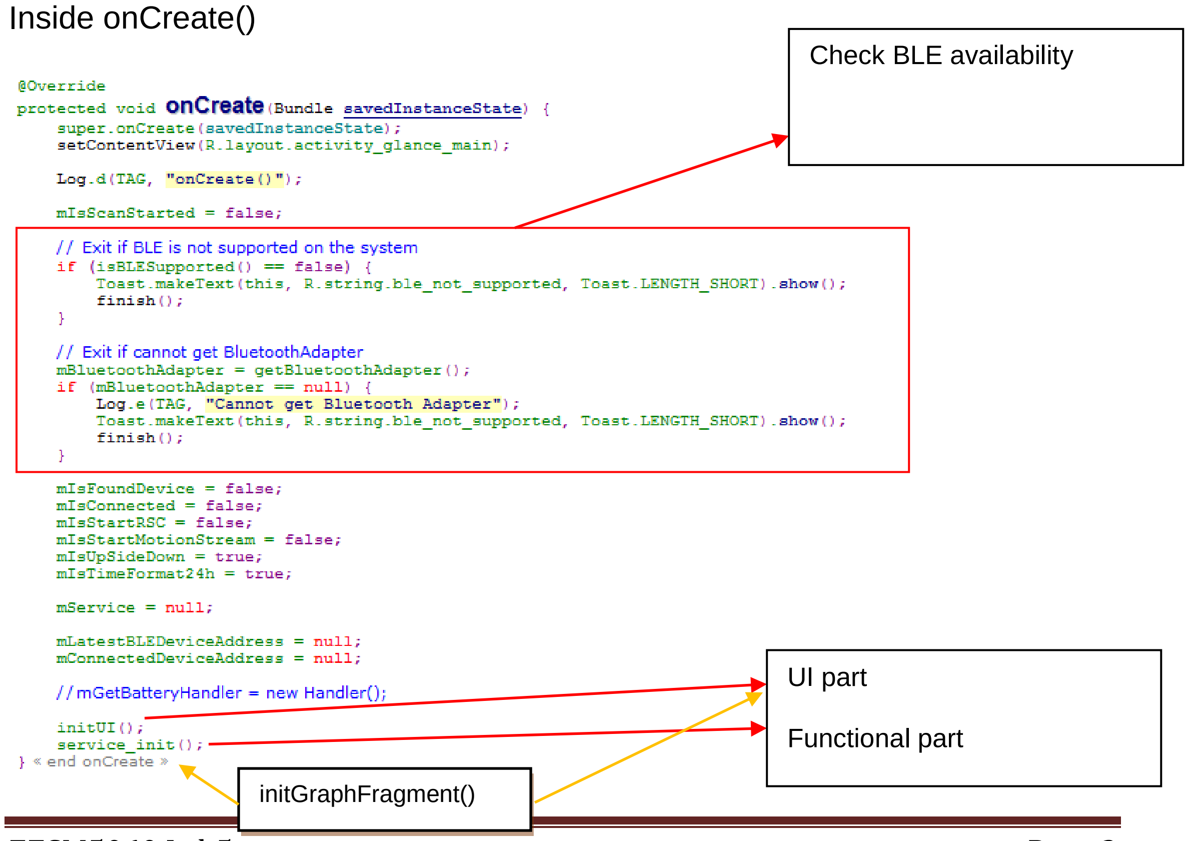
SET_USER_DATA_PROFILE; GET_USER_DATA_PROFILE
SET_USER_DATA_GOAL; GET_ DATA_GOAL
GET_FW_VERSION
SET_DATETIME;GET_DATETIME
SET_AUTOWALK_INTERVAL;GET_AUTOWALK_INTERVAL
GET_STEP_COUNT

It also allow real-time sampling of streaming of RSC and Motion Stream data.In this lab, the glance sensor will collect motion information, with BLE to talk with phone Apps.

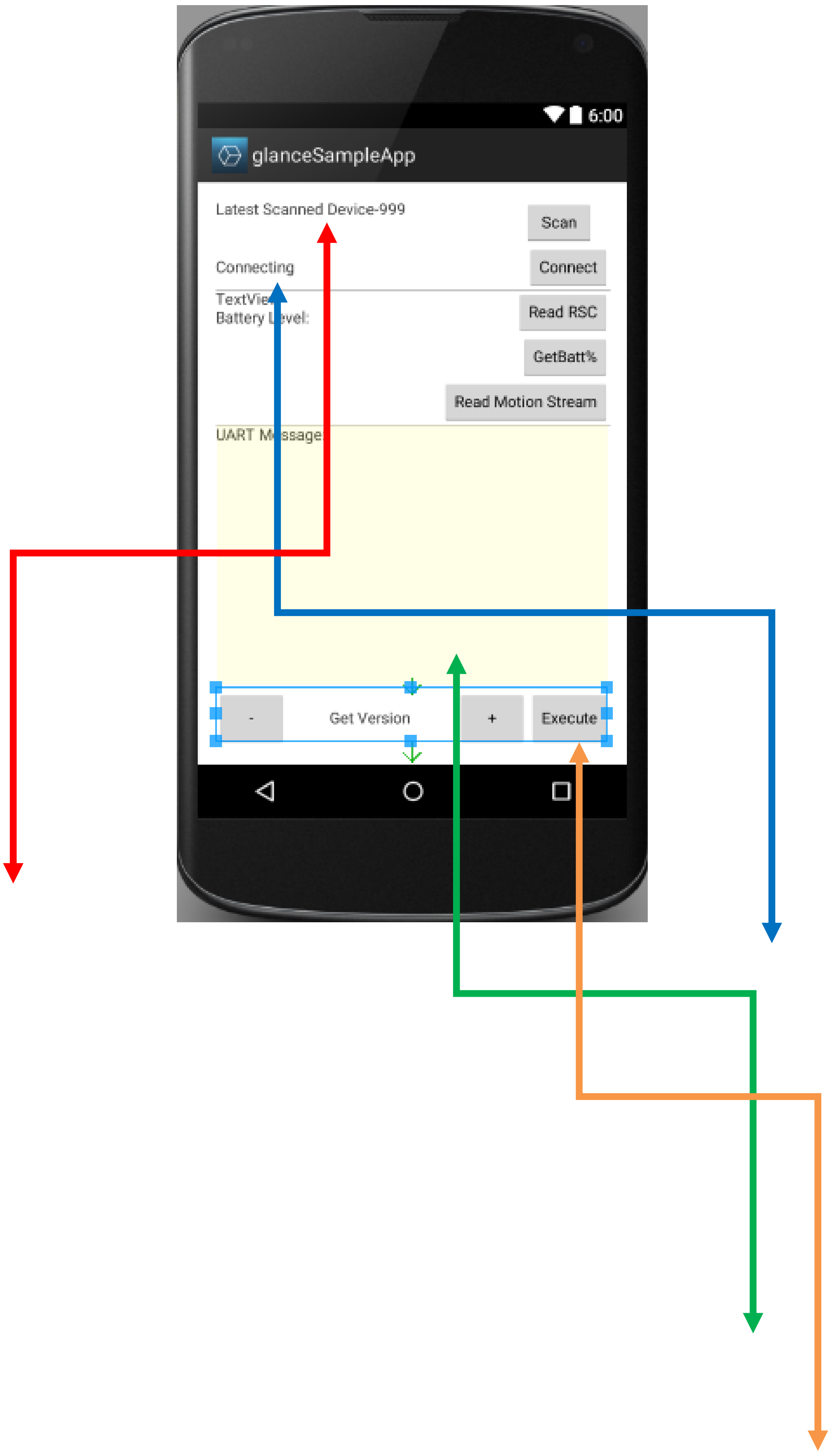
Display: OLED
Memory: 7-day detailed daily and sport motions tracked minute by minute
Power: 8 - 12 days
Wrist Size: 17cm – 22cm

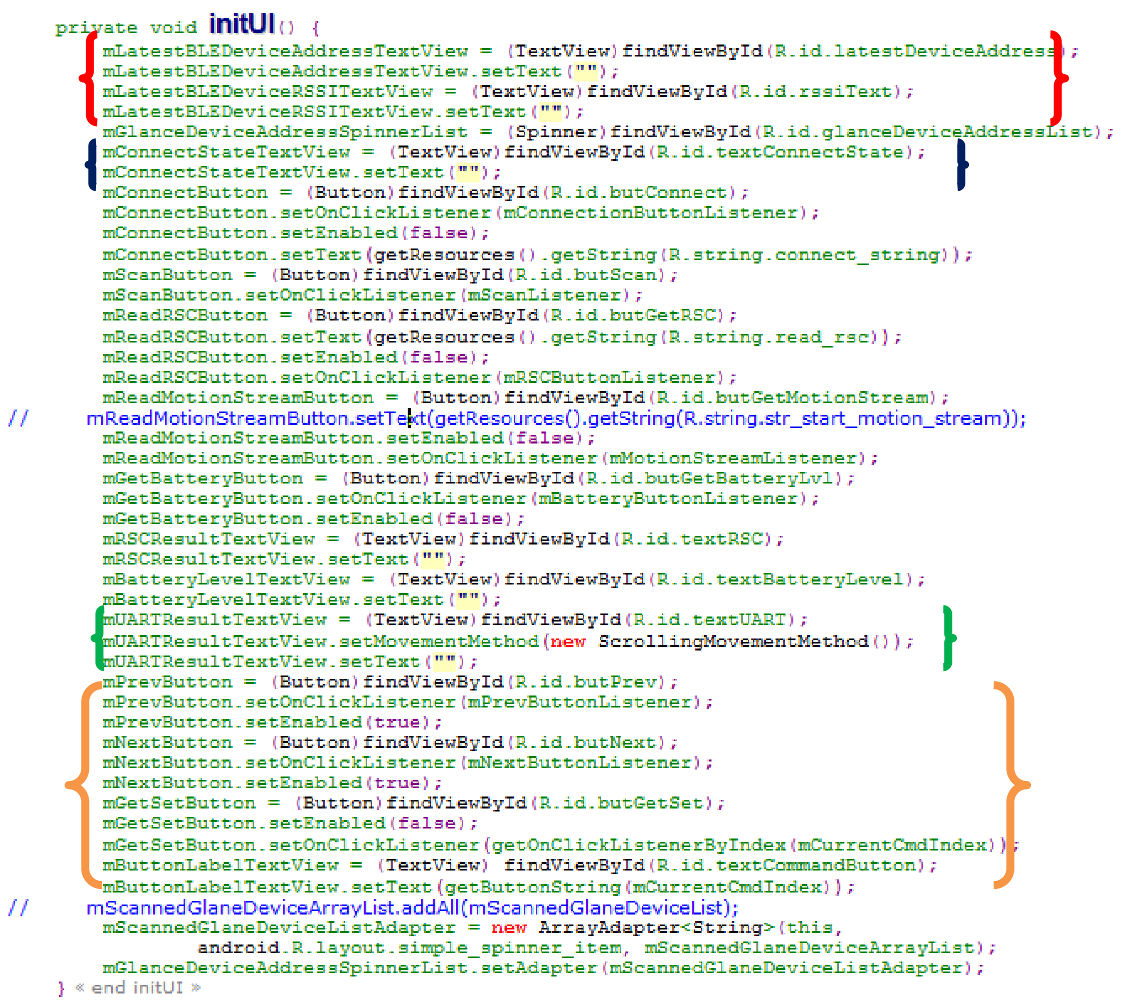
Glance Wearable

Understanding the system archictecture:
Inside onCreate()



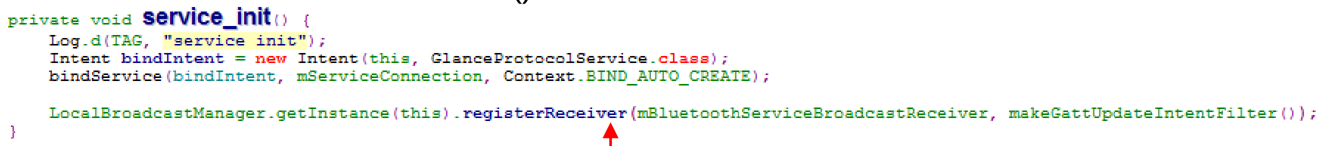
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_glance_main);

    Log.d(TAG, "onCreate()");

    mIsScanStarted = false;

    // Exit if BLE is not supported on the system
    if (isBLESupported() == false) {
        Toast.makeText(this, R.string.ble_not_supported, Toast.LENGTH_SHORT).show();
        finish();
    }

    // Exit if cannot get BluetoothAdapter
    mBluetoothAdapter = getBluetoothAdapter();
    if (mBluetoothAdapter == null) {
        Log.e(TAG, "Cannot get Bluetooth Adapter");
        Toast.makeText(this, R.string.ble_not_supported, Toast.LENGTH_SHORT).show();
        finish();
    }

    mIsFoundDevice = false;
    mIsConnected = false;
    mIsStartRSC = false;
    mIsStartMotionStream = false;
    mIsUpSideDown = true;
    mIsTimeFormat24h = true;

    mService = null;

    mLatestBLEDeviceAddress = null;
    mConnectedDeviceAddress = null;

    //mGetBatteryHandler = new Handler();

    initUI();
    service_init();
} « end onCreate »
```


6:00
glanceSampleApp
Latest Scanned Device-999
Scan
Connecting
Connect
TextView
Battery Level:
Read RSC
GetBatt%
Read Motion Stream
UART Message
-
Get Version
+
Execute



```
    private void initUI() {
        mLatestBLEDeviceAddressTextView = (TextView)findViewById(R.id.latestDeviceAddress);
        mLatestBLEDeviceAddressTextView.setText("");
        mLatestBLEDeviceRSSITextView = (TextView)findViewById(R.id.rssiText);
        mLatestBLEDeviceRSSITextView.setText("");
        mGlanceDeviceAddressSpinnerList = (Spinner)findViewById(R.id.glanceDeviceAddressList);
        mConnectStateTextView = (TextView)findViewById(R.id.textConnectState);
        mConnectStateTextView.setText("");
        mConnectButton = (Button)findViewById(R.id.butConnect);
        mConnectButton.setOnClickListener(mConnectionButtonListener);
        mConnectButton.setEnabled(false);
        mConnectButton.setText(getResources().getString(R.string.connect_string));
        mScanButton = (Button)findViewById(R.id.butScan);
        mScanButton.setOnClickListener(mScanListener);
        mReadRSCButton = (Button)findViewById(R.id.butGetRSC);
        mReadRSCButton.setText(getResources().getString(R.string.read_rsc));
        mReadRSCButton.setEnabled(false);
        mReadRSCButton.setOnClickListener(mRSCButtonListener);
        mReadMotionStreamButton = (Button)findViewById(R.id.butGetMotionStream);
//      mReadMotionStreamButton.setText(getResources().getString(R.string.str_start_motion_stream));
        mReadMotionStreamButton.setEnabled(false);
        mReadMotionStreamButton.setOnClickListener(mMotionStreamListener);
        mGetBatteryButton = (Button)findViewById(R.id.butGetBatteryLvl);
        mGetBatteryButton.setOnClickListener(mBatteryButtonListener);
        mGetBatteryButton.setEnabled(false);
        mRSCResultTextView = (TextView)findViewById(R.id.textRSC);
        mRSCResultTextView.setText("");
        mBatteryLevelTextView = (TextView)findViewById(R.id.textBatteryLevel);
        mBatteryLevelTextView.setText("");
        mUARTResultTextView = (TextView)findViewById(R.id.textUART);
        mUARTResultTextView.setMovementMethod(new ScrollingMovementMethod());
        mUARTResultTextView.setText("");
        mPrevButton = (Button)findViewById(R.id.butPrev);
        mPrevButton.setOnClickListener(mPrevButtonListener);
        mPrevButton.setEnabled(true);
        mNextButton = (Button)findViewById(R.id.butNext);
        mNextButton.setOnClickListener(mNextButtonListener);
        mNextButton.setEnabled(true);
        mGetSetButton = (Button)findViewById(R.id.butGetSet);
        mGetSetButton.setEnabled(false);
        mGetSetButton.setOnClickListener(getOnClickListenerByIndex(mCurrentCmdIndex));
        mButtonLabelTextView = (TextView) findViewById(R.id.textCommandButton);
        mButtonLabelTextView.setText(getButtonString(mCurrentCmdIndex));
//      mScannedGlaneDeviceArrayList.addAll(mScannedGlaneDeviceList);
        mScannedGlaneDeviceListAdapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_spinner_item, mScannedGlaneDeviceArrayList);
        mGlanceDeviceAddressSpinnerList.setAdapter(mScannedGlaneDeviceListAdapter);
    } « end initUI »
```

InitUI()



```
private void service_init() {
    Log.d(TAG, "service init");
    Intent bindIntent = new Intent(this, GlanceProtocolService.class);
    bindService(bindIntent, mServiceConnection, Context.BIND_AUTO_CREATE);

    LocalBroadcastManager.getInstance(this).registerReceiver(mBluetoothServiceBroadcastReceiver, makeGattUpdateIntentFilter());
}
```

service_init()

"Service init" will start bluetooth service connection, setup bluetooth service receiver for handling service event. Intent filter are added to tell what kind of services are needed in protocol

Program procedure (To plan what should need to do to reach the target)

Task 1 – Familiar with the project environment, build the app and connect to glance
Task 2 – Getting data from glance, understand how it work
Task 3 – Get gravity value from sensor and plot a real-time stream graph

Start to do:

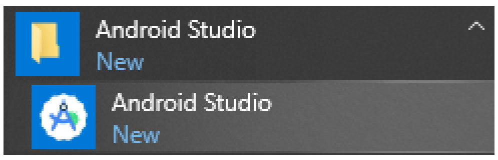
Task 1	Familiar with the project, build and try the app

External resources

	1) Phone of Android 8.0 or above and support BLE is recommended. You can set target build to 8.0 to run in other android phone

Procedure of the task:

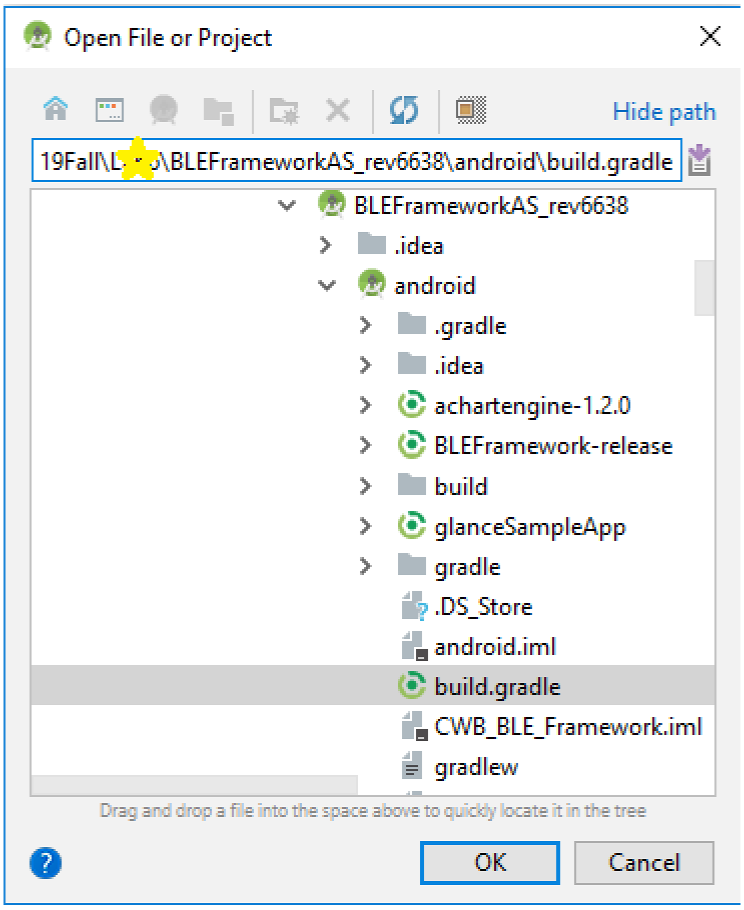
01) Open Android Studio in the startup menu
Start -> Android Studio->Android Studio



02) The Glance Sample Project was put under folder \lab5

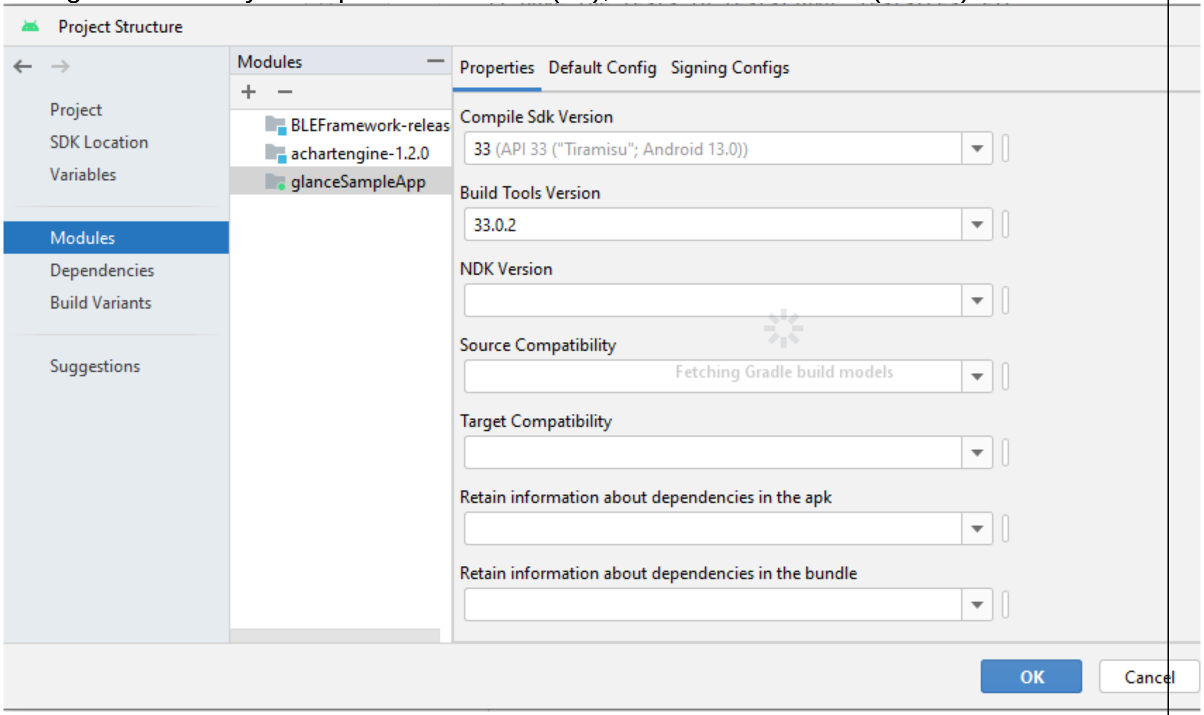
In the "Welcome to Android Studio" Interface, choose "Open an existing Android Studio Project" option.

Select "build.gradle" in the path
lab5\BLEFrameworkAS_rev6638\android\build.gradle

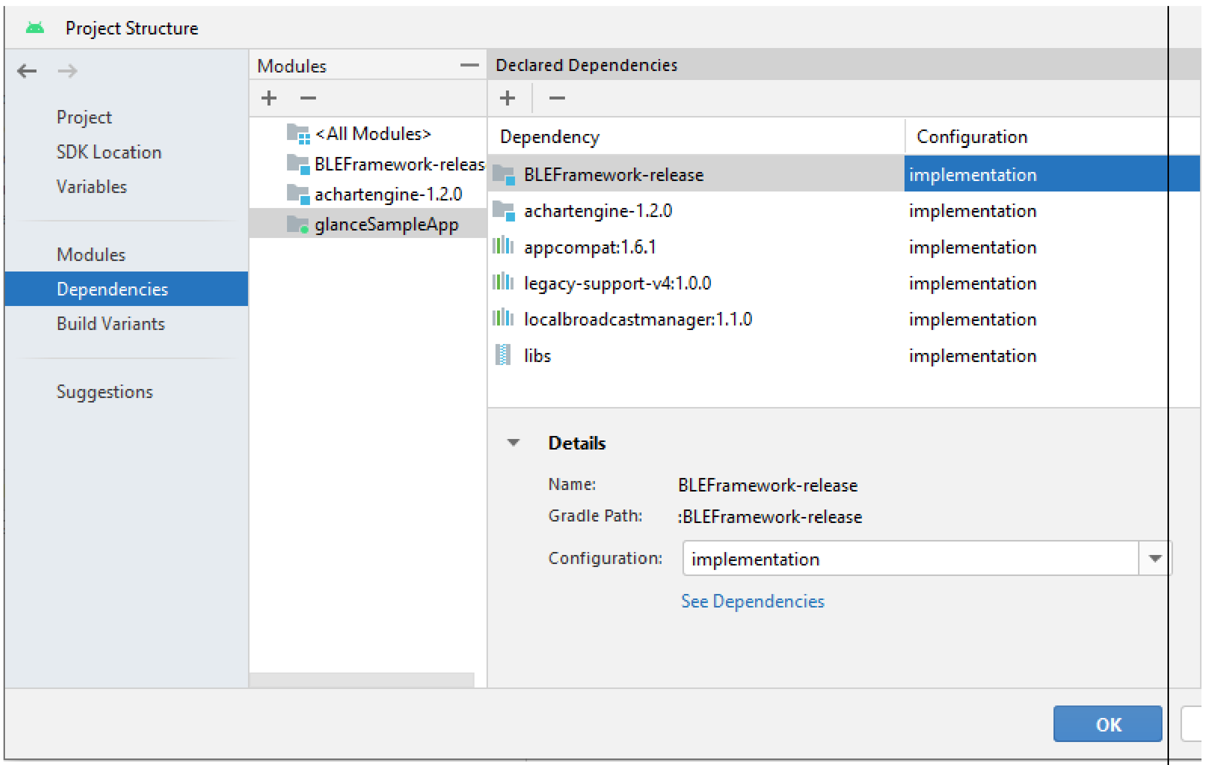


then click OK

03) Make sure the module is configured as below. Check the build dependencies is configured correctly. Compile Sdk Version(33), Build Tools Version(33.0.x)



Compile Sdk Version: 33 (API 33 ("Tiramisu"; Android 13.0))
Build Tools Version: 33.0.2
NDK Version
Source Compatibility
Target Compatibility
Retain information about dependencies in the apk
Retain information about dependencies in the bundle



Project Structure

Project
SDK Location
Variables
Modules
Dependencies
Build Variants
Suggestions

Modules
<All Modules>
BLEFramework-releas
achartengine-1.2.0
glanceSampleApp

Declared Dependencies

Dependency	Configuration
BLEFramework-release	implementation
achartengine-1.2.0	implementation
appcompat:1.6.1	implementation
legacy-support-v4:1.0.0	implementation
localbroadcastmanager:1.1.0	implementation
libs	implementation

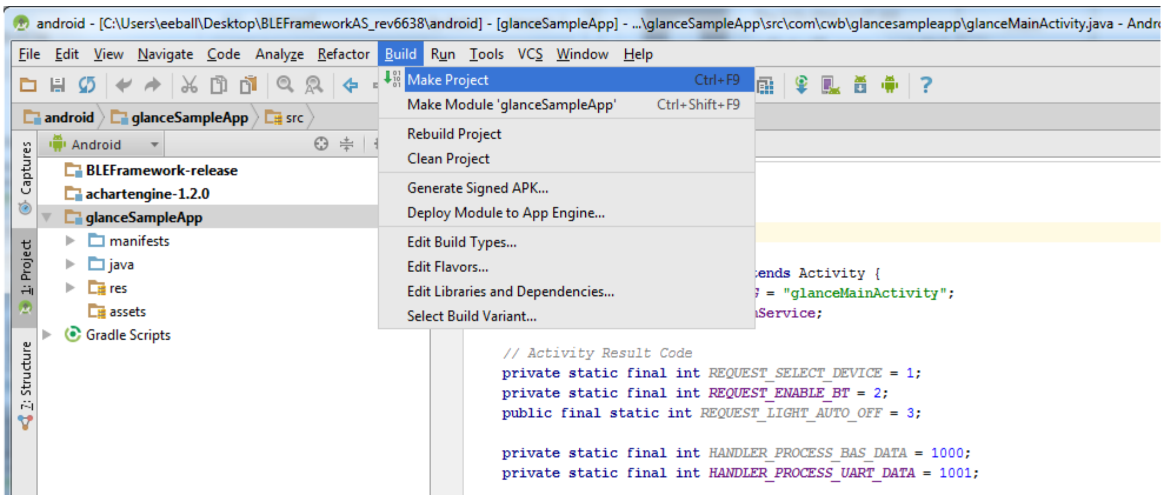
Details

Name: BLEFramework-release
Gradle Path: :BLEFramework-release
Configuration: implementation

See Dependencies

OK

04) Click Build->Make Project, Build->Build Bundle(s)/APK(s)->Build APK(s) to build the App
05) Install the App in android phone and try.



android - [C:\Users\eeball\Desktop\BLEFrameworkAS_rev6638\android] - [glanceSampleApp] - ...\glanceSampleApp\src\com\cwb\glancesampleapp\glanceMainActivity.java - Andr

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

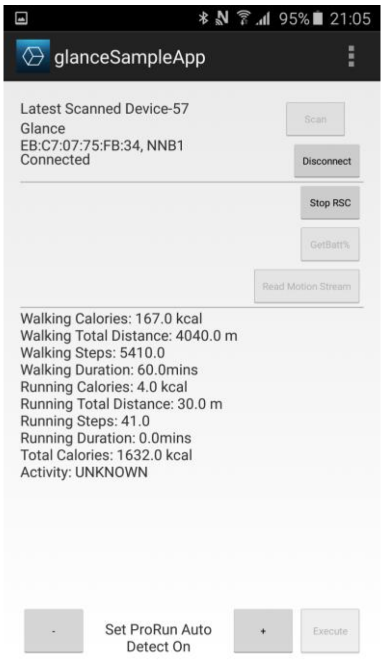
android > glanceSampleApp > src

Android
BLEFramework-release
achartengine-1.2.0
glanceSampleApp
manifests
java
res
assets
Gradle Scripts

Make Project	Ctrl+F9
Make Module 'glanceSampleApp'	Ctrl+Shift+F9
Rebuild Project	
Clean Project	
Generate Signed APK...	
Deploy Module to App Engine...	
Edit Build Types...	
Edit Flavors...	
Edit Libraries and Dependencies...	
Select Build Variant...	

```
ends Activity {
; = "glanceMainActivity";
Service;

// Activity Result Code
private static final int REQUEST_SELECT_DEVICE = 1;
private static final int REQUEST_ENABLE_BT = 2;
public final static int REQUEST_LIGHT_AUTO_OFF = 3;

private static final int HANDLER_PROCESS_BAS_DATA = 1000;
private static final int HANDLER_PROCESS_UART_DATA = 1001;
```

06) Make sure you can open the App like this **Self-Checkpoint 1**



Latest Scanned Device-57
Glance
EB:C7:07:75:FB:34, NNB1
Connected

Walking Calories: 167.0 kcal
Walking Total Distance: 4040.0 m
Walking Steps: 5410.0
Walking Duration: 60.0mins
Running Calories: 4.0 kcal
Running Total Distance: 30.0 m
Running Steps: 41.0
Running Duration: 0.0mins
Total Calories: 1632.0 kcal
Activity: UNKNOWN

Set ProRun Auto Detect On

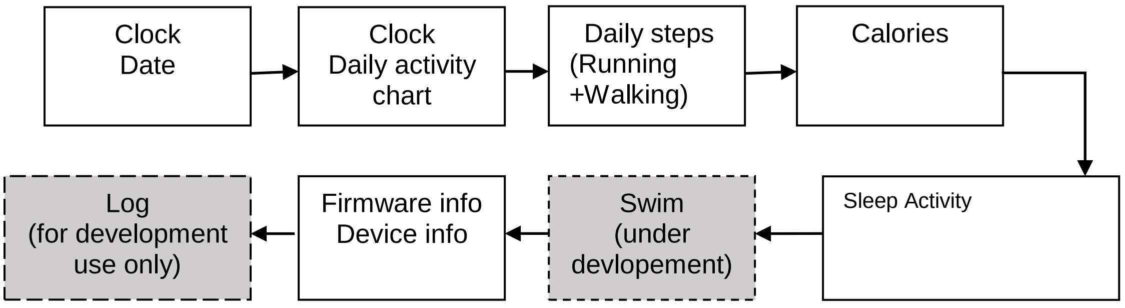
07) Press ONCE to power on the glance device,
Find the Mac address(in the label behind) and connect to the App

1) Open the App->Scan->Find out and click the Mac address(last 4 digits) in the list-> Click Connect
2) The status should be “Connecting”, press the glance device once
3) If success, the status should change to “Connected”
4) If fail, kill the App and repeat 1)-3) again

Task 2	Getting data from glance and understand how it work

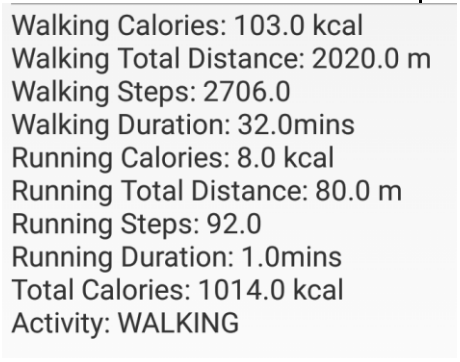
A firmware was already loaded inside the glance device. By pressing it(Just press, do not hold), it toggle in different modes with UI shown in OLED.

UI flow:



with this expression methodology, many other modes can be added in this chain.

Once power up, the device will start recording steps, calories and sleep parameters immediately.
Play with the app, click “Read RSC” button and see how the parameters changes:



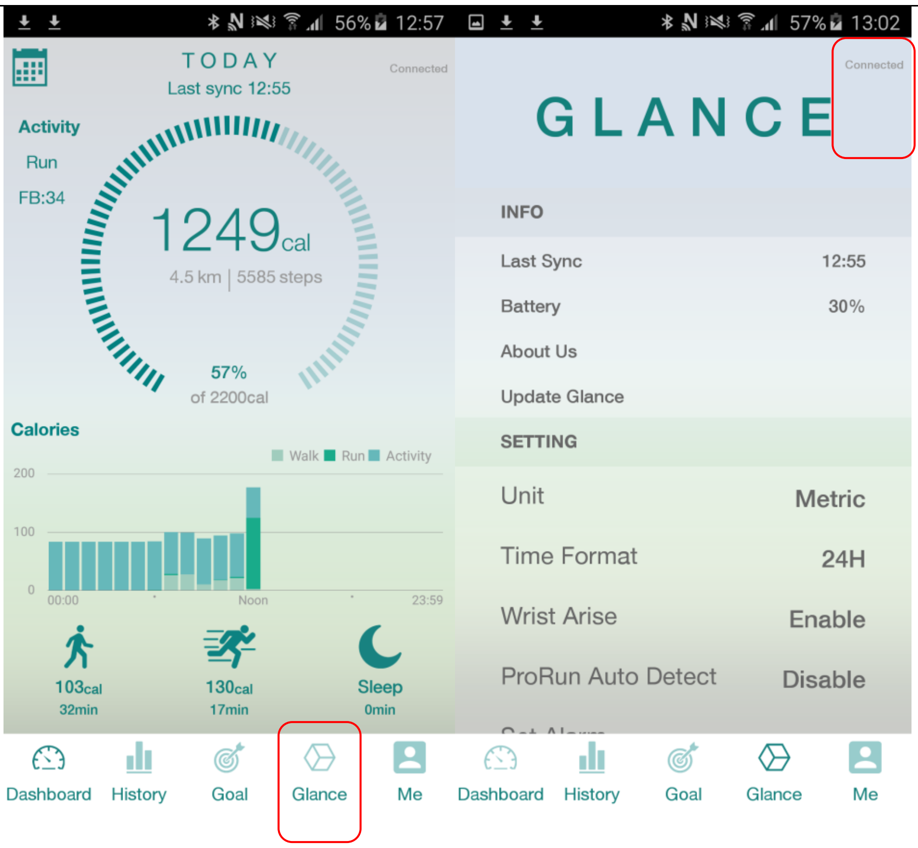
```
Walking Calories: 103.0 kcal
Walking Total Distance: 2020.0 m
Walking Steps: 2706.0
Walking Duration: 32.0mins
Running Calories: 8.0 kcal
Running Total Distance: 80.0 m
Running Steps: 92.0
Running Duration: 1.0mins
Total Calories: 1014.0 kcal
Activity: WALKING
```

Exercise1:(This part is just for reference, it is NOT an requirement for this lab)
1)Spend some time and see how the above parameters changes
2)Parameters are live update.
3)Locate the code in the project associating with this feature. Modify the code to disable this feature.

Exercise2: (This part is just for reference, it is NOT an requirement for this lab AND AutoWalk mode was removed in firmware)
Try to send command to the device:
Try the “ Get AutoWalk and Set AutoWalk” command. After that, “Start RSC” to see how parameter change. Can you tell what is AutoWalk mode?

While the device is **SET TO AutoWalk mode**, install another App Glance
Source:
From apkpure (https://apkpure.com/glance-wearable-for-runner/com.cwb.glance)
OR “Glance_1.16-2.apk” in zip file

After installation and pairing with the device, you should be able to get something like this. The app provide another mean to interpret the data(Make sure you have to disconnect from previous App before connecting to this App)



INFO

Last Sync	12:55
Battery	30%
About Us	
Update Glance	

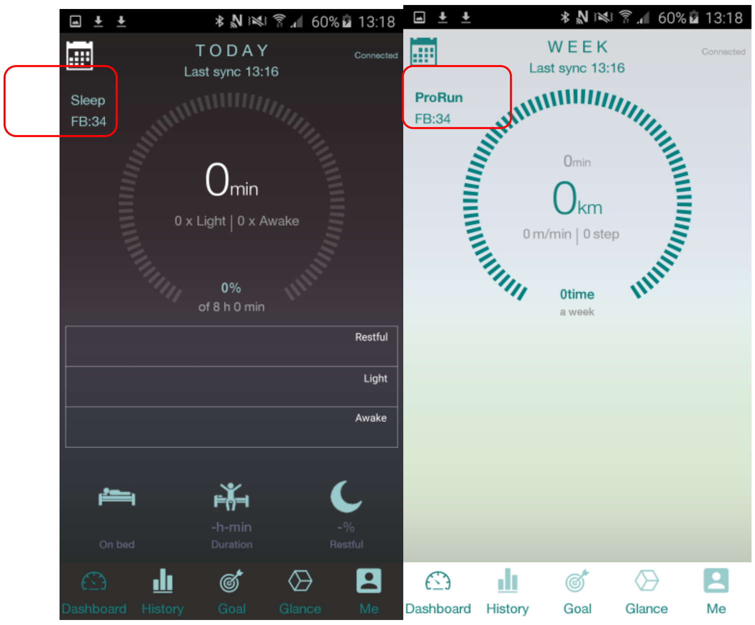
SETTING

Unit	Metric
Time Format	24H
Wrist Arise	Enable
ProRun Auto Detect	Disable

To pair the device:

Click Glance, Press the device, If connected, your device will virbate once.

Can you see the step count is still incrementing as we set the device in AutoWalk Mode before



There are 3 operation modes in Dashboard: Activity-Sleep-ProRun. You can futher investigate how it works.

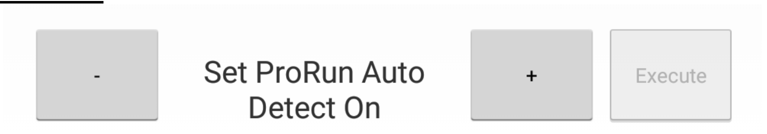
(SetAutoWalk function may not work and this is okay as some of glance devices firmware had been upgraded.)

SetAutoWalk back to 0 after this step

SetAutoWalk back to 0 after this step

SetAutoWalk back to 0 after this step

Knowledge Check Point:



What did you learnt up to now? This is a real work that engineers make a product!
Now go back to the **glanceSampleApp**
There are a number of commands in the command list. If we want to trim down the command list menu to the list as below, could you try how to accomplish this?

Target:

Set/Get AutoWalk<->Set Secondary Display(Ascii String)<->Set Secondary Display(Bitmap)<->Get Firmware Version

Hints: Go back into initUI and check with getButtonString,getOnClickListenerByIndex,COMMAND_LIST

Task 3	Gyroscope and gravity sensor data graph plotting**(This part is required)**

Knowledge learn in this task:

1) Identify how to get gyroscope and gravity data from glance for futher processing

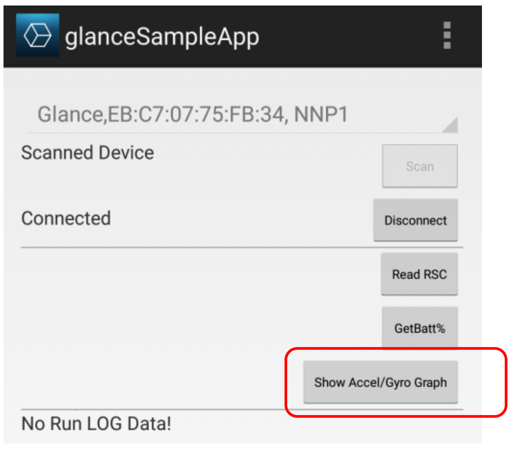
2) Identify how to stream data from glance in fast sampling interval

3) Identify how to use plot API

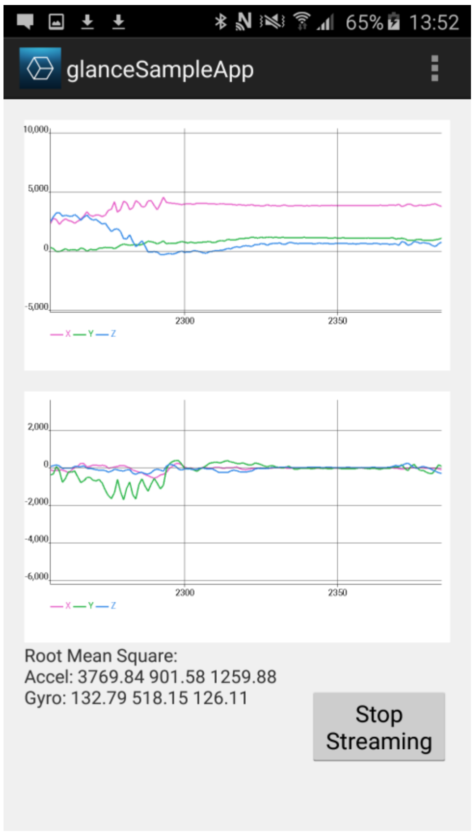
Code studying: Identify the mechanism how accelorometer graph plot were done. In task 3, show also the gyro data graph.

Requirments:

1) Change the text:



2) Put the 2 graphs in parallel, show also the gyro data, with Start/Stop Streaming button in below:



Root Mean Square:
Accel: 3769.84 901.58 1259.88
Gyro: 132.79 518.15 126.11

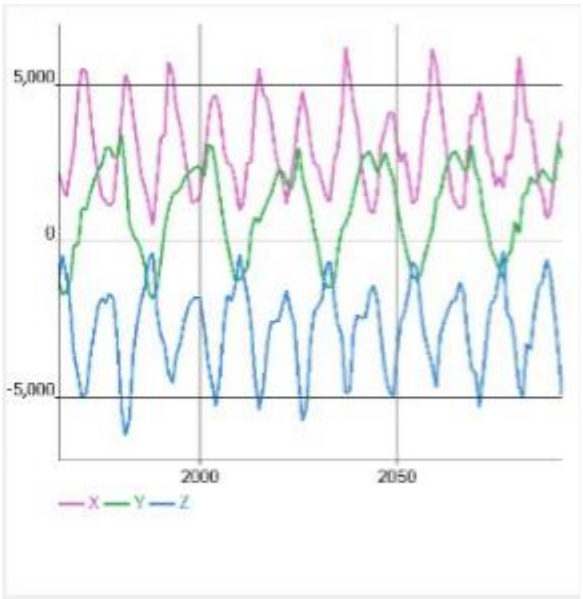
Hints: Edit including, GraphFragment.java and fragment_layout_graph.xml file
Reference how variable, mGraphViewAccl doing

For your information, if you want to develop an APP using Glance Protocol in future, you have to add the service in manifest.xml

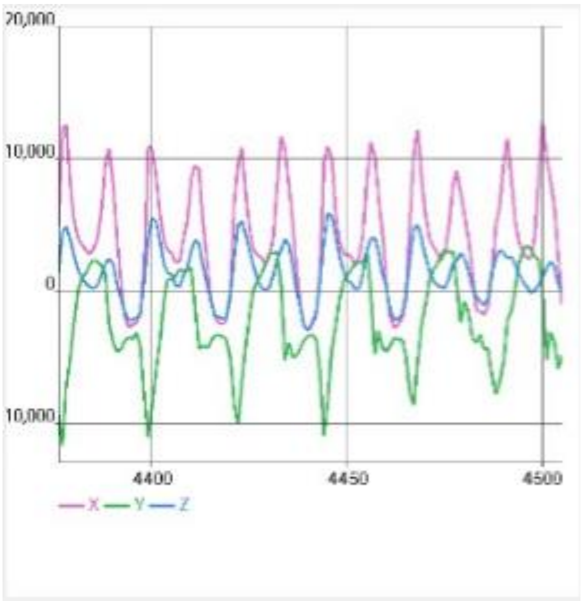
Please notice you have to add:
<service android:enabled="true"
android:name="com.cwb.bleframework.GlanceProtocolService" />
when using the glance protocolservice

Test the apps if you can see two graphs appeared as above and is it reasonable. **Self-Checkpoint 2**(This part is required)

Before you leave, there are two graphs of X-,Y-,Z- values, that one of them describes "Walking" and the other one describes "Running". Could you distinguish among them and reasons? Using what you learnt in the class?



Root Mean Square:
Accel: 3281.16 1767.34 3139.09



Root Mean Square:
Accel: 6115.54 4303.48 2459.43

Hand-in method:

In the class demonstration to Person-In-Charge