CS551-HW10-Q6

Project Status –**Running**

Link -http://npu85.npu.edu/~henry/npu/classes/capstone/android/slide/exercise_android.html

[Q6] Integrate these two programs/processes so that an Android device can send accelerometer and Gyroscope/Orientation data to the server

Option 2: Determining falling down

- Update **Hellondroid** code to send data to Server

Added this code to `onSensorChanged()` function so that any change in sensor data can be reported to the server

```
public void onSensorChanged(SensorEvent event){

    int sensor = event.type;
    float[] values = event.values;

    // check sensor type
    if( sensor==Sensor.TYPE_ACCELEROMETER){

        // assign directions
        float x=event.values[0];
        float y=event.values[1];
        float z=event.values[2];

        xCoor.setText("X: "+x);
        yCoor.setText("Y: "+y);
        zCoor.setText("Z: "+z);
    }
    if( sensor ==  Sensor.TYPE_ORIENTATION ){

        // assign directions
        float x=event.values[0];
        float y=event.values[1];
        float z=event.values[2];

        xOrCoor.setText("X: "+x);
        yOrCoor.setText("Y: "+y);
        zOrCoor.setText("Z: "+z);
    }

    // Connect to Server ------- START
```

```
    // Connect to Server ------- START

    String[] value1 = xCoor.getText().toString().split(":");
    String[] value2 = yCoor.getText().toString().split(":");
    String[] value3 = zCoor.getText().toString().split(":");
    String[] value4 = xOrCoor.getText().toString().split(":");
    String[] value5 = yOrCoor.getText().toString().split(":");
    String[] value6 = zOrCoor.getText().toString().split(":");

    String sendData = value1[1] + "," + value2[1] +  "," + value3[1]
                    + "," + value4[1] + "," + value5[1] + "," + value6[1];

    Socket socket = null;
    DataOutputStream dataOutputStream = null;
    DataInputStream dataInputStream = null;

    try {

        socket = new Socket("10.0.2.2", 8888);
        Log.d("ServerSocket", "Connected: -" + sendData);
        dataOutputStream = new DataOutputStream(socket.getOutputStream());
        dataInputStream = new DataInputStream(socket.getInputStream());
        dataOutputStream.writeUTF(sendData);
        //textIn.setText(dataInputStream.readUTF());
    } catch (UnknownHostException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

- Update Server code to Include KNN algorithm code to determine Fall.

Added code to run() method

Note: Complete code is included at the end of the document.

```
public void run() {
    while (true) {
        try {

            DataOutputStream output;
            DataInputStream input;
            Socket connection = theServer.accept();

            input = new DataInputStream(connection.getInputStream());
            output = new DataOutputStream(connection.getOutputStream());
            System.out.println("Client Connected and Start get I/O!!");
            while (true) {
                // System.out.println("==> Input from Client: " +
                // input.readUTF());

                // Split the string & get each coordinate value
                // (x,y,z,x,y,z)
                String inputFromAndroid = input.readUTF();

                // System.out.println(inputFromAndroid);
                String[] inputArr = inputFromAndroid.split(",");

                // convert to integer values
                float[] coordinates = new float[inputArr.length];
                for (int i = 0; i < coordinates.length; i++) {
                    coordinates[i] = Float.parseFloat(inputArr[i]);
                }

                System.out.format(
                        "Accelerometer    X: %f \n %15s Y: %f \n %15s Z: %f \n"
                                + "Orientation      X: %f \n %15s Y: %f \n %15s Z: %f \nFall Down (YES/NO):",
                        coordinates[0], "", coordinates[1], "", coordinates[2], coordinates[3], "", coordinates[4],
                        "", coordinates[5]);
```

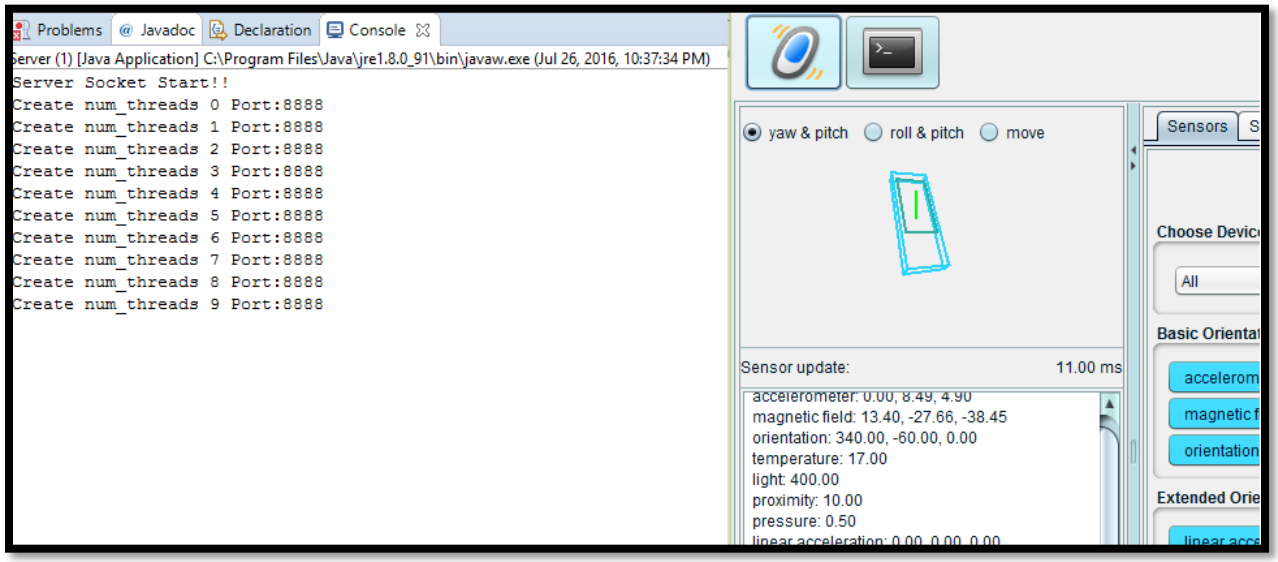
- Add some training data to the file. I added it as per some readings from simulator.

(Acceleration & Orientation X,Y,Z values are comma delimited ."-" depicts NO FALL & "+" depicts FALL)

test.java | Server.java | Pair.java | training_data

```
1 0,-9,2,346,75,0,-
2 0,9,1,349,-83,0,-
3 0,0,9,287,-4,0,-
4 0,9,0,321,-86,0,-
5 0,1,9,293,-7,0,-
6 0,6,7,9,-40,0,-
7 -0.02,-9,-0,286,86,-180,+
8 -0.02,1,-9,117,-7,-180,+
9 -0.02,2,9,73,-11,0,+
10 -0.02,-9,-1,209,84,-180,+
11 -0.02,0,-9,89,1,-180,+
12 -0.02,9,0,321,-86,0,+
```

- Start the Server
- Start the Sensor Simulator

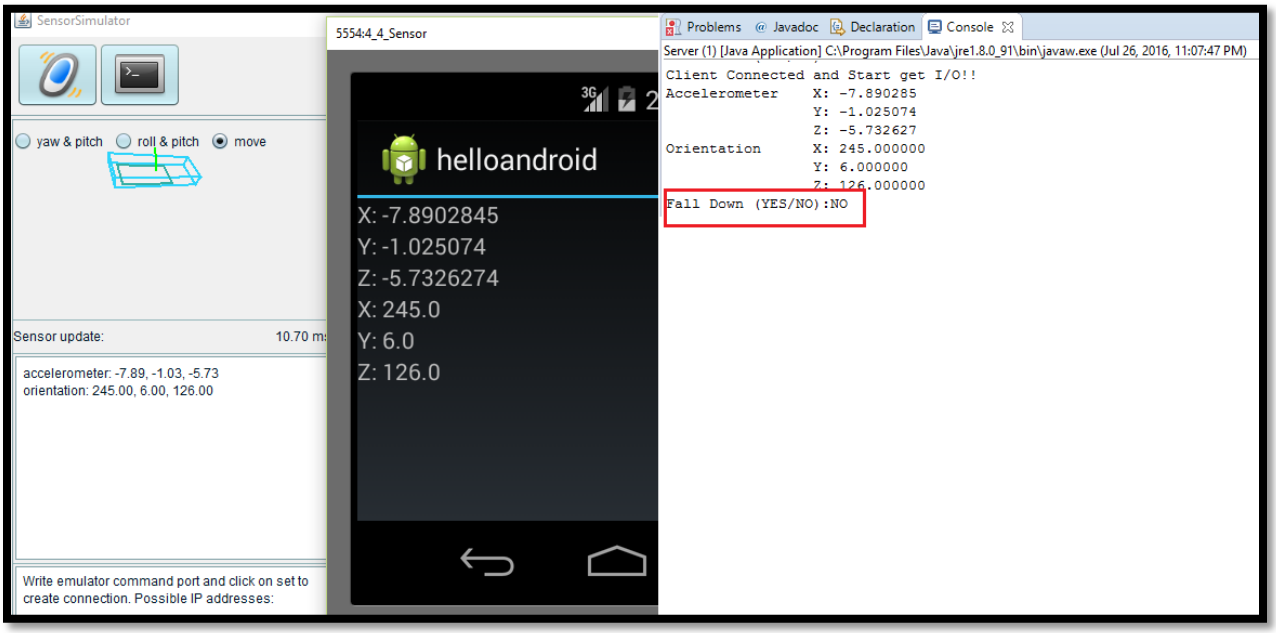


- Start the emulator & Run the HelloAndroid App

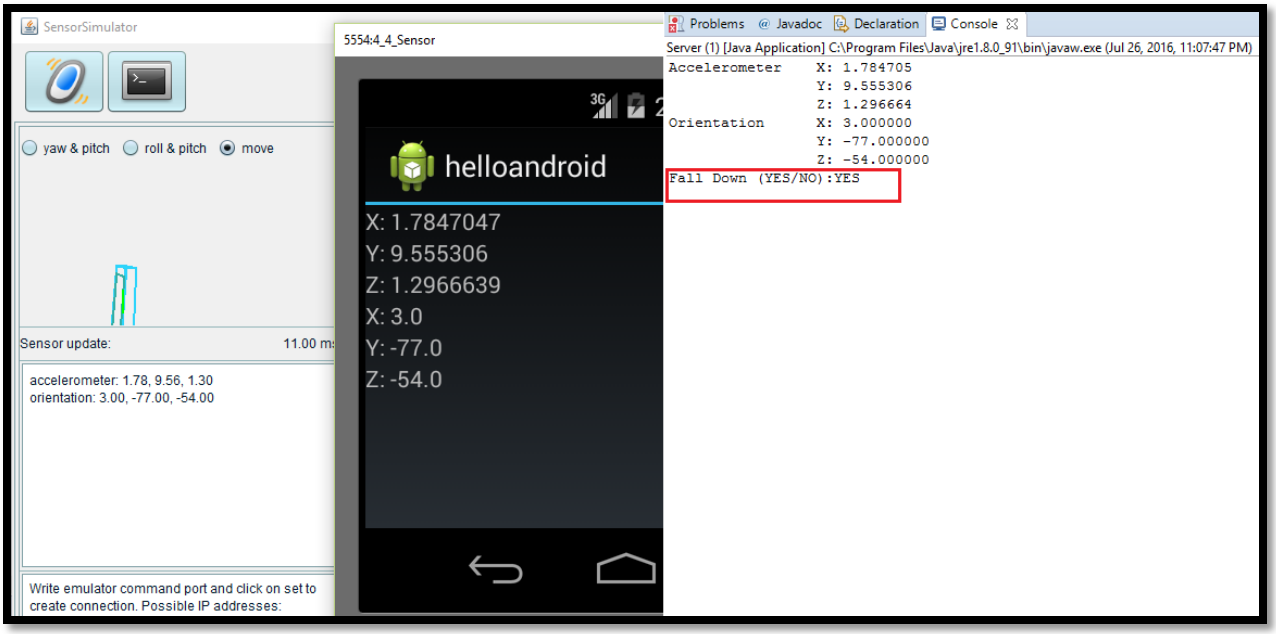
The sensor simulator data will reflect on Emulator first. The same data will be send to Server which will calculate & show the RESULT

Case 1 FALL scenario

Case 2 NO FALL scenario



Case 3 Again FALL



- **Code for Server**

```
import java.net.*;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collection;

import java.util.Collections;

import java.util.List;

import java.util.Scanner;

import java.io.*;


public class Server extends Thread {

public final static int defaultPort = 8888;
```

```
ServerSocket theServer;
static int num_threads = 1;



public static void main(String[] args) {
int port = defaultPort;
try {
port = Integer.parseInt(args[0]);
} catch (Exception e) {
}
if (port <= 0 || port >= 65536)
port = defaultPort;
try {
ServerSocket ss = new ServerSocket(port);
System.out.println("Server Socket Start!!");
for (int i = 0; i < num_threads; i++) {
System.out.println("Create num_threads " + i + " Port:" + port);
Server pes = new Server(ss);
pes.start();
}
} catch (IOException e) {
System.err.println(e);
}
}
```

```
public Server(ServerSocket ss) {

theServer = ss;

}



public void run() {

while (true) {

try {



DataOutputStream output;

DataInputStream input;

Socket connection = theServer.accept();



input = new DataInputStream(connection.getInputStream());

output = new DataOutputStream(connection.getOutputStream());

System.out.println("Client Connected and Start get I/O!!");

while (true) {

// System.out.println("==> Input from Client: " +

// input.readUTF());



// Split the string & get each coordinate value

// (x,y,z,x,y,z)

String inputFromAndroid = input.readUTF();
```

```
// System.out.println(inputFromAndroid);
String[] inputArr = inputFromAndroid.split(",");


// convert to integer values
float[] coordinates = new float[inputArr.length];
for (int i = 0; i < coordinates.length; i++) {
coordinates[i] = Float.parseFloat(inputArr[i]);
}
System.out.format(
"Accelerometer    X: %f \n %15s Y: %f \n %15s Z: %f \n"
+ "Orientation      X: %f \n %15s Y: %f \n %15s Z: %f \nFall Down (YES/NO):",
coordinates[0], "", coordinates[1], "", coordinates[2], coordinates[3], "", coordinates[4],
"", coordinates[5]);


// Logic to read file, calculate distance & decide FALL or
// Not FALL
String fileName = "src/training_data";
String line = null;
List<Pair<Integer, String>> charList = new ArrayList<Pair<Integer, String>>();
List<Integer> distances = new ArrayList<Integer>();
int records = 0;
int fall = 0;
int notfall = 0;
```

```
try {


// Read training data
FileReader fileReader = new FileReader(fileName);
// Always wrap FileReader in BufferedReader.
BufferedReader bufferedReader = new BufferedReader(fileReader);


// Read one record at a time & calculate distance
while ((line = bufferedReader.readLine()) != null) {


String[] lineVal = line.split(",");
// System.out.println( Arrays.toString(lineVal));
int distance = (int) Math.pow((Float.parseFloat(lineVal[0]) - coordinates[0]), 2)
+ (int) Math.pow((Float.parseFloat(lineVal[1]) - coordinates[1]), 2)
+ (int) Math.pow((Float.parseFloat(lineVal[2]) - coordinates[2]), 2)
+ (int) Math.pow((Float.parseFloat(lineVal[3]) - coordinates[3]), 2)
+ (int) Math.pow((Float.parseFloat(lineVal[4]) - coordinates[4]), 2)
+ (int) Math.pow((Float.parseFloat(lineVal[5]) - coordinates[5]), 2);


charList.add(new Pair<Integer, String>(distance, lineVal[6])); // calculatesDistance,+/-
distances.add(distance); // calculatedDistances
```

```
//System.out.println( distance );
records++;
}

// Calculate threshold( here it is 2 always coz 8
// records , square root of 8 = approx 2 )
int threshold = (int) Math.pow(records, 0.5);

// sort distances
Collections.sort(distances);

// Check & decide the minimun distances
for (Pair<Integer, String> pair : charList) {

// System.out.println(pair.key + " -> " +
// pair.value);

for (int j = 0; j < threshold; j++) {
if (pair.key.compareTo(distances.get(j)) == 0) {
if (pair.value.contains("+")) {
fall++;
} else {
```

```
notfall++;
}
}
}


}
// System.out.println("fall = " + fall + "Notfall =" +
// notfall);
if (fall != 0 || notfall != 0) {
if (fall >= notfall) {
System.out.println("YES");
// output.writeUTF("FALL");
} else {
System.out.println("NO");
// output.writeUTF("NO FALL DETECTED");
}
} else {
output.writeUTF("NOT SURE");
}


// Always close files.
bufferedReader.close();
// File operations end
} catch (IOException e) {
```

```
//e.printStackTrace();

}



// System.out.println("Output to Client ==> \"Connection

// successful\"");

output.writeUTF("Connection successful");

output.flush();

} // end while

} // end try

catch (IOException e) {



//e.printStackTrace();

}



}

}



}



public class Pair<T, U> {

   public final T key;

   public final U value;
```

```
    public Pair(T key, U value) {
        this.key = key;
        this.value = value;
    }
}
```