Lab 5: Sending Sensor Data from Android to Pure Data

In this weeks lab we will extend our android app further. So far we have learned how to create a GUI for our app and how to send data from this GUI to our pd patch and from our patch to our GUI. Today we are going to learn how to send data from our phone’s sensors to the pure data patch. In this lab, we will be using accelerometer and location values.

Setup

Download the example from Github: https://github.com/Hydroxate/Lab5\_Template

Rename the containing folder to firstname\_surname\_lab05.

(This step is important as otherwise another student may accidentally delete your project.)

Open the provided template in android studio.

Part 1: Using Accelerometer Sensor Values

Add 6 TextView Widgets

Following on from last week’s lab, you should know how to drag textView widgets and update their text and IDs.

Step 1: Add the widget graphically

• Click on activity\_main.xml in the project pane, it is located under res/layout.

• Recreate the following interface layout in the design view with 6 textView Widgets.

Click on the TextView widgets that are now in place. Their properties should now be shown in the properties tab to the right hand side.

Now we need to set the properties for these new textViews.

• Go to the properties tab.

• Rename the TextView’s Text to the strings shown in the previous diagram.

• Rename the TextViews’s IDs to the following:

AccelXText

AccelXValue

AccelYText

AccelYValue

AccelZText

AccelZValue

Step 2: Create an instance in code

At this stage our widgets only exist as graphical objects, essentially a picture on the canvas. We have finished step 1 of our 2 step process. On to step two, “create an instance of the widget type in code with a unique name and link this to the graphical object we just made.”

• Double click the MainActivity.java file in the project pane. This is located under the java folder.

Lets declare our TextViews. Go to line 40 and type the below:

TextView accelX;

TextView accelY;

TextView accelZ;

Now let’s create the instance of the code at line 49 in the OnCreate() method. It will look like the below:

accelX = (TextView) findViewById(R.id.AccelXValue);

accelY = (TextView) findViewById(R.id.AccelYValue);

accelZ = (TextView) findViewById(R.id.AccelZValue);

We have now completed step 2 of our two step process. We have created 3 TextView instance with unique names and linked it to the 3 graphical TextView objects that we placed on the canvas in step 1.

Now we need to create our SensorEventListener!

Step 3: Add the sensorEventListener.

We want our app to receive data from our accelerometer in our phone. To do this we must add a sensorEventListener first: code that listens for changes in sensor values. This needs to be added to the class declaration which is at line 36.

• Go to line 37 and add the following after AppCompatActivity

implements SensorEventListener

It should look like the following:

There will be an error here as SensorEventListener is an abstract class. This means that all the methods from the class have to be implemented here.

• Click on SensorEventListener and press “alt + Enter”. You will see a the following prompt:

• Click on “Implement methods”. This will show the necessary methods:

• Click “OK”.

Android Studio will now add the onSensorChanged() method and onAccuracyChanged()Method. What do you think these mean?

Create the listeners’s functionality

Now that we have added the listener, we must create the functionality of the listener to update the TextViews and send the values to the pure data patch.

• Go to OnCreate() and add the following:

mSensorManager = (SensorManager) getSystemService(Context.*SENSOR\_SERVICE*);  
mAccelerometer = mSensorManager.getDefaultSensor(Sensor.*TYPE\_ACCELEROMETER*);  
  
mSensorManager.registerListener(this, mAccelerometer, SensorManager.*SENSOR\_DELAY\_NORMAL*);

Here we bind the sensor manager provided for you to the system sensor service, and the sensor provided for you to the phone’s accelerometer. We then register the listener to(the onSensorChanged method in this class, the accelerometer, and updates every DELAY\_NORMAL).

• Go to the method OnSensorChanged() and add the following:

Sensor mySensor = sensorEvent.sensor;  
  
if (mySensor.getType() == Sensor.*TYPE\_ACCELEROMETER*)  
{  
  
 float x = sensorEvent.values[0];  
 float y = sensorEvent.values[1];  
 float z = sensorEvent.values[2];  
  
 sendFloatPD("accelX", x);  
 accelX.setText(String.*valueOf*(x));  
 sendFloatPD("accelY", y);  
 accelY.setText(String.*valueOf*(y));  
 sendFloatPD("accelZ", z);  
 accelZ.setText(String.*valueOf*(z));  
  
}

Here we check to see if the sensor that triggered the listener is an accelerometer.. If this is true, then we get the X,Y,Z values and send them to pure data and update the text in the GUI accordingly.

• Uncomment the commented statements in OnResume() and OnPause() methods.

• Run the application if you have an android phone. If you are running in the emulator, check the section at the end of this sheet to feed test values to the sensors.

You should see the text update as you tilt the phone .

Your task is:

• Edit your patch to use the values coming from your accelerometer.

• Use the [r accelX], [r accelY], [r accelZ] objects to use the data.

Part 2: Using GPS Latitude and Longitude Values

Add 4 TextView Widgets

Step 1: Add the widgets graphically

• Click on activity\_main.xml in the project pane, it is located under res/layout.

• Add to your screen the following textViews for GPS support.

Click on the TextView widgets that are now in place. Their properties should now be shown in the properties tab to the right hand side.

Now we need to set the properties for these new textViews.

• Go to the properties tab.

• Rename the TextView’s Text to the strings shown in the previous diagram.

• Rename the TextViews’s IDs to the following:

LatText

LatValue

LongText

LongValue

Step 2: Create an instance in code

At this stage our widgets only exist as graphical objects, essentially a picture on the canvas. We have finished step 1 of our 2 step process. On to step two, “create an instance of the widget type in code with a unique name and link this to the graphical object we just made.”

• Double click the MainActivity.java file in the project pane. This is located under the java folder.

Lets declare our TextViews. Go to line 45 and type the below:

TextView lati;

TextView longi;

Now let’s create the instance of the code at line 48 in the OnCreate() method. It will look like the below:

lati = (TextView) findViewById(R.id.LatValue);

longi = (TextView) findViewById(R.id.LongValue);

We have now completed step 2 of our two step process. We have created 2 TextView instance with unique names and linked it to the 2 graphical TextView objects that we placed on the canvas in step 1.

Now we need to create our LocationListener!

Step 3: Add the LocationListener.

We want our app to receive data from our GPS in our phone. To do this we must add a LocationListener first: code that listens for changes in location values. This needs to be added in the class declaration which is at line 36.

• Go to line 37 and add the following after SensorEventListener

, LocationListener

It should look like the following:

There will be an error here as LocationListener is an abstract class. This means that all the methods from the class have to be implemented here.

• Click on LocationListener and press “alt + Enter”.

You will see a the following prompt:

• Click on “Implement methods”.

This will show the necessary methods:

• Click “OK”.

Android Studio will now add the onLocationChanged() method, onProviderEnabled() method, onProviderDisabled(), and onStatusChanged() method. What do you think these mean?

Create the listeners’s functionality

Now that we have added the listener, we must create the functionality of the listener to update the TextViews and send the values to the pure data patch.

• Go to the end of OnCreate() and add the following:

if (ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED* && ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_COARSE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED*)  
{  
 ActivityCompat.*requestPermissions*(this, new String[]{Manifest.permission.*ACCESS\_COARSE\_LOCATION*, Manifest.permission.*ACCESS\_FINE\_LOCATION*}, 1);  
 return;  
}  
  
LocationManager mlocManager = (LocationManager) getSystemService(Context.*LOCATION\_SERVICE*);  
  
mlocManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 0, 1,this);

Since the GPS location values are private data. We must first ask the user when the app is installed to access their location. The if statement checks the current permissions given, if the app doesnt have access to the location, it will ask the user for permission.

The line

LocationManager mlocManager = (LocationManager) getSystemService(Context.*LOCATION\_SERVICE*);

Creates a manager tied to the location service of the phone. We need this manager to specify the time and distance to request updates for the location of the phone.

mlocManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 0, 1,this);

This line requests updates from (GPS\_Provider, polls every 0ms, updates once 1m covered, the listener that we have is located in *this* class.).

• Go to OnLocationChanged().

• Add the following code:

latitude = location.getLatitude();  
longitude = location.getLongitude();  
Log.*i*("MyLocation", Double.*toString*(latitude) + " " + Double.*toString*(longitude));  
sendFloatPD("latitude",Float.*parseFloat*(Double.*toString*(latitude)));  
lati.setText(String.*valueOf*(latitude));  
sendFloatPD("longitude",Float.*parseFloat*(Double.*toString*(longitude)));

longi.setText(String.*valueOf*(longitude));

• Run the application if you have an android phone. If you are running in the emulator, check the section at the end of this sheet to feed test values to the sensors.

Your task is:

• Edit your patch to use the values coming from your locations latitude and longitude values.

• Use the [r latitude], [r longitude] objects to use the data.

Sending Sensor Values in Android Emulator

When the emulator is running, look at the side control panel.

• Click “---“.

• The default screen for Location will come up. It is shown below:

• The fields shown for longitude and latitude are present in the top right. Editing these values and clicking “Send” will update the location of the emulator.

• Click the side-tab labeled “Virtual Sensors” . This is where the accelerometer values can be changed.

• By changing the Yaw, Pitch, and Roll values, you can update the accelerometer values in real time.

Going Onwards

The homework for this lab will be sent out shortly.