NAME: SAKSHI VINAYAK KITTURKAR
CSU ID: 2860273
HOMEWORK 16

Revise the sensor listing app so that the sensor values are displayed in a chart. You may use any charting library. The chart should display the most recent 30-second of the data collected for each sensor.

= In this report I have a sensor application where sensor values are displayed in a chart the chart displays the most 30 seconds of the data collected for each of the sensor.

In layout: MainActivity.java This Android code is for an app that displays a list of sensors available on the device and allows the user to select a sensor to view its capabilities.

In the onCreate method, the layout is set (activity_main.xml), and references to the UI elements are obtained. The SensorManager is initialized, and the list of all sensors on the device is retrieved. An instance of ListAdapter is created and set as the adapter for the ListView. Additionally, an OnItemClickListener is set for the ListView to handle sensor item clicks.

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

   mSensorManager = (SensorManager) this.getSystemService(Context.SENSOR_SERVICE);
   mSensorsList = mSensorManager.getSensorList(Sensor.TYPE_ALL);
   mSensorListView = (ListView) findViewById(R.id.session_list);
   mListAdapter = new ListAdapter();
   mSensorListView.setAdapter(mListAdapter);
   mSensorListView.setOnItemClickListener(this);
}
```

This method is invoked when a sensor item in the ListView is clicked. It creates an Intent to start the SensorCapabilityActivity and passes the sensor type as an extra.

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
   Intent intent = new Intent(getApplicationContext(), SensorCapabilityActivity.class);
   intent.putExtra(getResources().getResourceName(R.string.sensor_type),
         mSensorsList.get(position).getType());
```

```
    startActivity(intent);
}
```

This is an inner class representing the custom adapter for the ListView. It extends BaseAdapter and provides methods to populate the ListView with sensor information. The getView method is responsible for creating or reusing list item views, and it sets the sensor name for each item.

```
private class ListAdapter extends BaseAdapter {
    private TextView mSensorName;

    @Override
    public int getCount() {
        return mSensorsList.size();
    }

    @Override
    public Object getItem(int position) {
        return mSensorsList.get(position).getName();
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            convertView = getLayoutInflater().inflate(R.layout.list_rows, parent, false);
        }
        mSensorName = (TextView) convertView.findViewById(R.id.sensor_name);
        mSensorName.setText(mSensorsList.get(position).getName());
        return convertView;
    }
}
```

In layout: SensorCapabilityActivity This code is for an Android app's activity (SensorCapabilityActivity) that displays detailed information about a specific sensor on the device.

In the onCreate method, the layout is set (activity_sensor_capability.xml), and references to the UI elements are obtained. The intent is used to retrieve the sensor type from the previous activity, and the SensorManager is used to obtain the default sensor of that type. The various TextView widgets are then updated with information about the sensor, such as its name, maximum range, minimum delay, power consumption, resolution, vendor, and version.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_sensor_capability);
   Intent intent = getIntent();
   mSensorType = intent.getIntExtra(getResources().getResourceName(R.string.sensor_type), 0);
   mSensorManager = (SensorManager) this.getSystemService(Context.SENSOR_SERVICE);
   mSensor = mSensorManager.getDefaultSensor(mSensorType);
   mSensorNameTextView = (TextView) findViewById(R.id.sensor_name);
   mSensorMaximumRangeTextView = (TextView) findViewById(R.id.sensor_range);
   mSensorMinDelayTextView = (TextView) findViewById(R.id.sensor_mindelay);
   mSensorPowerTextView = (TextView) findViewById(R.id.sensor_power);
   mSensorResolutionTextView = (TextView) findViewById(R.id.sensor_resolution);
   mSensorVendorTextView = (TextView) findViewById(R.id.sensor_vendor);
   mSensorVersionTextView = (TextView) findViewById(R.id.sensor_version);
   mSensorNameTextView.setText(mSensor.getName());
   mSensorMaximumRangeTextView.setText(String.valueOf(mSensor.getMaximumRange()));
   mSensorMinDelayTextView.setText(String.valueOf(mSensor.getMinDelay()));
   mSensorPowerTextView.setText(String.valueOf(mSensor.getPower()));
   mSensorResolutionTextView.setText(String.valueOf(mSensor.getResolution()));
   mSensorVendorTextView.setText(String.valueOf(mSensor.getVendor()));
   mSensorVersionTextView.setText(String.valueOf(mSensor.getVersion()));
}
```

This method is triggered when a button with an onClick attribute referencing this method is clicked. It creates an intent to start the SensorValuesActivity and passes the sensor type as an extra.

```
public void onClickSensorValues(View v) {
   Intent intent = new Intent(getApplicationContext(), SensorValuesActivity.class);
   intent.putExtra(getResources().getResourceName(R.string.sensor_type), mSensorType);
   startActivity(intent);
}
```

In layout: SensorValuesActivity This code is for an Android app's activity (SensorValuesActivity) that displays real-time sensor data on a graph for a specific sensor. The graph uses the GraphView library to visualize the sensor readings over time.

In the onCreate method, the layout is set (activity_sensor_values.xml), and references to the UI elements and the graph are obtained. A PointsGraphSeries is created for plotting data points, and various settings for the graph are initialized. The intent is used to retrieve the sensor type from the previous activity, and the SensorManager is used to obtain the default sensor of that type.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_sensor_values);
    graph = (GraphView) findViewById(R.id.graph);
    mSeries1 = new PointsGraphSeries<DataPoint>();
    graph.addSeries(mSeries1);

    // Graph settings
    // ...

    Intent intent = getIntent();
    mSensorType = intent.getIntExtra(getResources().getResourceName(R.string.sensor_type), 0);
    mSensorManager = (SensorManager) this.getSystemService(Context.SENSOR_SERVICE);
    mSensor = mSensorManager.getDefaultSensor(mSensorType);
}
```

These methods are part of the activity lifecycle. onResume registers the activity as a listener for sensor events, and onPause unregisters the listener to conserve resources when the activity is not in the foreground.

```
@Override
protected void onResume() {
    super.onResume();
    mSensorManager.registerListener(this, mSensor, SensorManager.SENSOR_DELAY_NORMAL);
}

@Override
protected void onPause() {
    super.onPause();
```

```
    mSensorManager.unregisterListener(this);
}
```

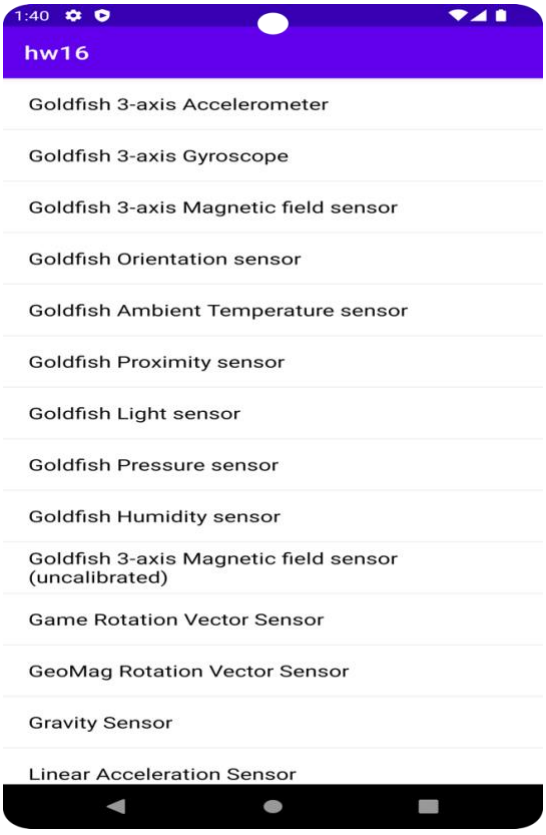
This method is called when the accuracy of the sensor changes. However, it is currently empty and does not perform any specific action.

```
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
}
```

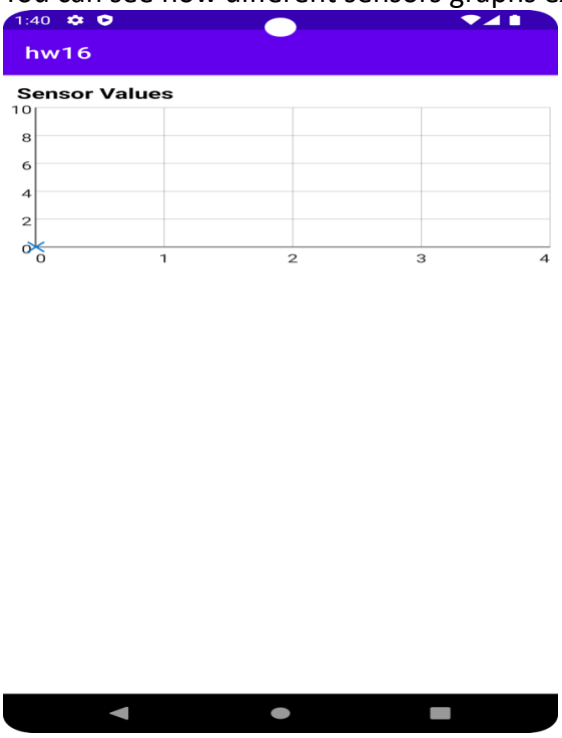
This method is called when the sensor readings change. It retrieves the sensor data (e.g., accelerometer values) and updates the graph in real-time. The sensor data is plotted against time, and the graph is configured to show a specific range of values. The PointsGraphSeries is updated with the new data point, and the graph is redrawn.

```
@Override
public void onSensorChanged(SensorEvent event) {
    // ...
}
```

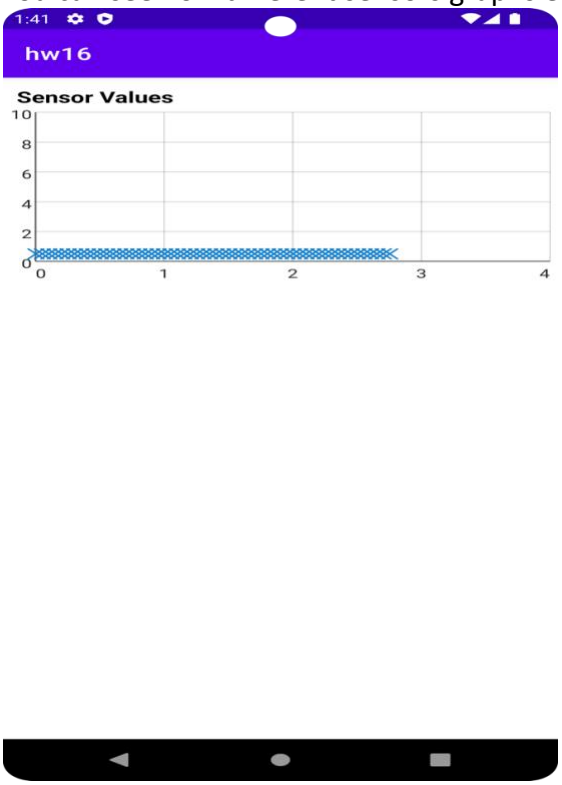
First screen with all the sensors:



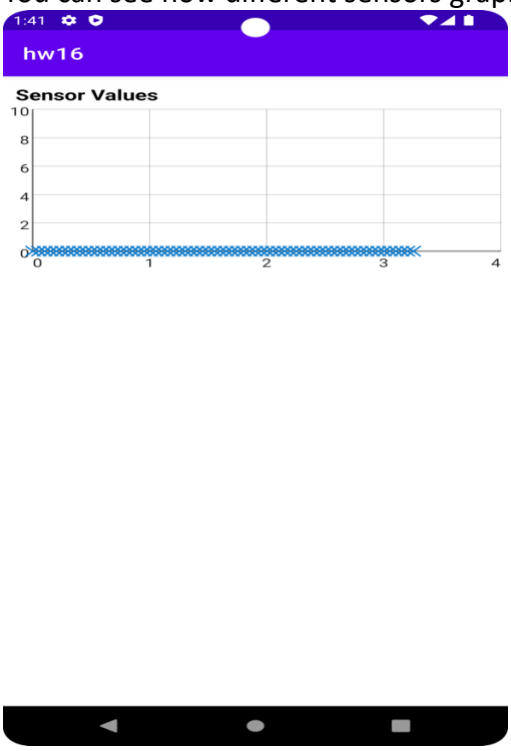
You can see now different sensors graphs example 1:



You can see now different sensors graphs example 2:



You can see now different sensors graphs example 3:



You can see now different sensors graphs example 4: