```java
package de.kai_morich.simple_bluetooth_terminal;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.AlertDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.text.Spannable;
import android.text.SpannableStringBuilder;
import android.text.style.ForegroundColorSpan;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import java.util.ArrayDeque;
import static java.lang.Math.*;



public class TerminalFragment extends Fragment implements ServiceConnection,
SerialListener {

    private enum Connected { False, Pending, True }

    private String deviceAddress;
    private SerialService service;

    //private TextView receiveText;

    private TextView speedText;
    private TextView gyroXText;
    private TextView gyroYText;
    private TextView gyroZText;
    private TextView anglGyrXText;
    private TextView anglGyrYText;
    private TextView anglGyrZText;
    private TextView accelXText;
    private TextView accelYText;
    private TextView accelZText;
    private TextView vbatText;
    private TextView vgenText;
    private TextView pitchText;
    private TextView yawText;
    private TextView rollText;


    private TextView sendText;
    private TextUtil.HexWatcher hexWatcher;

    private Connected connected = Connected.False;
    private boolean initialStart = true;
```

```
66      private boolean hexEnabled = false;
67      private boolean pendingNewline = false;
68      private String newline = TextUtil.newline_crlf;
69
70      // Personal variables
71      public double sValue = 0.0;
72      public double gxValue = 0.0;
73      public double gyValue = 0.0;
74      public double gzValue = 0.0;
75      public double gaxValue = 0.0;
76      public double gayValue = 0.0;
77      public double gazValue = 0.0;
78      public double axValue = 0.0;
79      public double ayValue = 0.0;
80      public double azValue = 0.0;
81      public double vbatValue = 0.0;
82      public double vgenValue = 0.0;
83
84      /*
85       * Lifecycle
86       */
87      @Override
88      public void onCreate(@Nullable Bundle savedInstanceState) {
89          super.onCreate(savedInstanceState);
90          setHasOptionsMenu(true);
91          setRetainInstance(true);
92          deviceAddress = getArguments().getString("device");
93      }
94
95      @Override
96      public void onDestroy() {
97          if (connected != Connected.False)
98              disconnect();
99          getActivity().stopService(new Intent(getActivity(), SerialService.class));
100         super.onDestroy();
101     }
102
103     @Override
104     public void onStart() {
105         super.onStart();
106         if(service != null)
107             service.attach(this);
108         else
109             getActivity().startService(new Intent(getActivity(), SerialService.class));
                // prevents service destroy on unbind from recreated activity caused by
                orientation change
110     }
111
112     @Override
113     public void onStop() {
114         if(service != null && !getActivity().isChangingConfigurations())
115             service.detach();
116         super.onStop();
117     }
118
119     @SuppressWarnings("deprecation") // onAttach(context) was added with API 23.
        onAttach(activity) works for all API versions
120     @Override
121     public void onAttach(@NonNull Activity activity) {
122         super.onAttach(activity);
123         getActivity().bindService(new Intent(getActivity(), SerialService.class), this,
            Context.BIND_AUTO_CREATE);
124     }
125
126     @Override
127     public void onDetach() {
```

```
128         try { getActivity().unbindService(this); } catch(Exception ignored) {}
129         super.onDetach();
130     }
131
132     @Override
133     public void onResume() {
134         super.onResume();
135         if(initialStart && service != null) {
136             initialStart = false;
137             getActivity().runOnUiThread(this::connect);
138         }
139     }
140
141     @Override
142     public void onServiceConnected(ComponentName name, IBinder binder) {
143         service = ((SerialService.SerialBinder) binder).getService();
144         service.attach(this);
145         if(initialStart && isResumed()) {
146             initialStart = false;
147             getActivity().runOnUiThread(this::connect);
148         }
149     }
150
151     @Override
152     public void onServiceDisconnected(ComponentName name) {
153         service = null;
154     }
155
156     /*
157      * UI
158      */
159     @Override
160     public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
161         View view = inflater.inflate(R.layout.fragment_terminal, container, false);
162         // Assignment text to textView
163         speedText = view.findViewById(R.id.speed_viewer);
164         gyroXText = view.findViewById(R.id.gyroX_viewer);
165         gyroYText = view.findViewById(R.id.gyroY_viewer);
166         gyroZText = view.findViewById(R.id.gyroZ_viewer);
167         anglGyrXText = view.findViewById(R.id.angGyrX_viewer);
168         anglGyrYText = view.findViewById(R.id.angGyrY_viewer);
169         anglGyrZText = view.findViewById(R.id.angGyrZ_viewer);
170         accelXText = view.findViewById(R.id.accelX_viewer);
171         accelYText = view.findViewById(R.id.accelY_viewer);
172         accelZText = view.findViewById(R.id.accelZ_viewer);
173         vbatText = view.findViewById(R.id.vbat_viewer);
174         vgenText = view.findViewById(R.id.vgen_viewer);
175         pitchText = view.findViewById(R.id.pitch_viewer);
176         yawText = view.findViewById(R.id.yaw_viewer);
177         rollText = view.findViewById(R.id.roll_viewer);
178
179         //receiveText = view.findViewById(R.id.receive_text);
            // TextView performance decreases with number of spans
180         //receiveText.setTextColor(getResources().getColor(R.color.colorRecieveText));
            // set as default color to reduce number of spans
181         //receiveText.setMovementMethod(ScrollingMovementMethod.getInstance());
182
183         sendText = view.findViewById(R.id.send_text);
184         hexWatcher = new TextUtil.HexWatcher(sendText);
185         hexWatcher.enable(hexEnabled);
186         sendText.addTextChangedListener(hexWatcher);
187         sendText.setHint(hexEnabled ? "HEX mode" : "");
188
189         View sendBtn = view.findViewById(R.id.send_btn);
190         sendBtn.setOnClickListener(v -> send(sendText.getText().toString()));
```

```
191         return view;
192     }
193
194     @Override
195     public void onCreateOptionsMenu(@NonNull Menu menu, MenuInflater inflater) {
196         inflater.inflate(R.menu.menu_terminal, menu);
197         menu.findItem(R.id.hex).setChecked(hexEnabled);
198     }
199
200     @Override
201     public boolean onOptionsItemSelected(MenuItem item) {
202         int id = item.getItemId();
203         if (id == R.id.clear) {
204             //receiveText.setText("");
205             return true;
206         } else if (id == R.id.newline) {
207             String[] newlineNames = getResources().getStringArray(R.array.newline_names);
208             String[] newlineValues = getResources().getStringArray(R.array.newline_values
                );
209             int pos = java.util.Arrays.asList(newlineValues).indexOf(newline);
210             AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
211             builder.setTitle("Newline");
212             builder.setSingleChoiceItems(newlineNames, pos, (dialog, item1) -> {
213                 newline = newlineValues[item1];
214                 dialog.dismiss();
215             });
216             builder.create().show();
217             return true;
218         } else if (id == R.id.hex) {
219             hexEnabled = !hexEnabled;
220             sendText.setText("");
221             hexWatcher.enable(hexEnabled);
222             sendText.setHint(hexEnabled ? "HEX mode" : "");
223             item.setChecked(hexEnabled);
224             return true;
225         } else {
226             return super.onOptionsItemSelected(item);
227         }
228     }
229
230     /*
231      * Serial + UI
232      */
233     private void connect() {
234         try {
235             BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
236             BluetoothDevice device = bluetoothAdapter.getRemoteDevice(deviceAddress);
237             status("connecting...");
238             connected = Connected.Pending;
239             SerialSocket socket = new SerialSocket(getActivity().getApplicationContext(),
                 device);
240             service.connect(socket);
241         } catch (Exception e) {
242             onSerialConnectError(e);
243         }
244     }
245
246     private void disconnect() {
247         connected = Connected.False;
248         service.disconnect();
249     }
250
251     private void send(String str) {
252         if(connected != Connected.True) {
253             Toast.makeText(getActivity(), "not connected", Toast.LENGTH_SHORT).show();
254             return;
```

```
255             }
256             try {
257                 String msg;
258                 byte[] data;
259                 if(hexEnabled) {
260                     StringBuilder sb = new StringBuilder();
261                     TextUtil.toHexString(sb, TextUtil.fromHexString(str));
262                     TextUtil.toHexString(sb, newline.getBytes());
263                     msg = sb.toString();
264                     data = TextUtil.fromHexString(msg);
265                 } else {
266                     msg = str;
267                     data = (str + newline).getBytes();
268                 }
269                 SpannableStringBuilder spn = new SpannableStringBuilder(msg + '\n');
270                 spn.setSpan(new ForegroundColorSpan(getResources().getColor(R.color.
                    colorSendText)), 0, spn.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
271                 //receiveText.append(spn);
272                 service.write(data);
273             } catch (Exception e) {
274                 onSerialIoError(e);
275             }
276         }
277
278         @SuppressLint("SetTextI18n")
279         private void receive(ArrayDeque<byte[]> datas) {
280             SpannableStringBuilder spn = new SpannableStringBuilder();
281             for (byte[] data : datas) {
282                 if (hexEnabled) {
283                     spn.append(TextUtil.toHexString(data)).append('\n');
284                 } else {
285                     String msg = new String(data);
286                     if (newline.equals(TextUtil.newline_crlf) && msg.length() > 0) {
287                         // don't show CR as ^M if directly before LF
288                         msg = msg.replace(TextUtil.newline_crlf, TextUtil.newline_lf);
289                         // special handling if CR and LF come in separate fragments
290                         if (pendingNewline && msg.charAt(0) == '\n') {
291                             if (spn.length() >= 2) {
292                                 spn.delete(spn.length() - 2, spn.length());
293                             } else {
294                                 //Editable edt = receiveText.getEditableText();
295                                 //if (edt != null && edt.length() >= 2)
296                                 //    edt.delete(edt.length() - 2, edt.length());
297                             }
298                         }
299                         pendingNewline = msg.charAt(msg.length() - 1) == '\r';
300                     }
301                     spn.append(TextUtil.toCaretString(msg, newline.length() != 0));
302                 }
303             }
304
            //--------------------------------------------------------------------------
            ---------// PERSONAL
305
306             String[] pairs = spn.toString().split("\\s+"); // Séparer la chaîne en paires
                clé-valeur
307
308             for (String pair : pairs) {
309                 String[] parts = pair.split("="); // Séparer chaque paire clé-valeur
310
311                 if (parts.length == 2) {
312                     String key = parts[0];
313                     String valueStr = parts[1];
314
315                     try {
316                             double value = Double.parseDouble(valueStr);
```

```
                    switch (key) {
                        case "S":
                            sValue = value;
                            break;
                        case "GX":
                            gxValue = value;
                            break;
                        case "GY":
                            gyValue = value;
                            break;
                        case "GZ":
                            gzValue = value;
                            break;
                        case "GAX":
                            gaxValue = value;
                            break;
                        case "GAY":
                            gayValue = value;
                            break;
                        case "GAZ":
                            gazValue = value;
                            break;
                        case "AX":
                            axValue = value;
                            break;
                        case "AY":
                            ayValue = value;
                            break;
                        case "AZ":
                            azValue = value;
                            break;
                        case "VB":
                            vbatValue = value;
                            break;
                        case "VG":
                            vgenValue = value;
                            break;

                    }
                } catch (NumberFormatException e) {
                    // Gérer l'exception si la valeur n'est pas un nombre décimal valide
                }
            }
        }


        speedText.setText(   "SPEED    : " + String.valueOf(sValue) + " km/h");
        gyroXText.setText(   "GYRO X   : " + String.valueOf(gxValue) + " dps");
        gyroYText.setText(   "GYRO Y   : " + String.valueOf(gyValue) + " dps");
        gyroZText.setText(   "GYRO Z   : " + String.valueOf(gzValue) + " dps");
        anglGyrXText.setText("ANGL GYR X   : " + String.valueOf(gaxValue) + " °");
        anglGyrYText.setText("ANGL GYR Y   : " + String.valueOf(gayValue) + " °");
        anglGyrZText.setText("ANGL GYR Z   : " + String.valueOf(gazValue) + " °");
        accelXText.setText(  "ACCEL X  : " + String.valueOf(axValue) + " g");
        accelYText.setText(  "ACCEL Y  : " + String.valueOf(ayValue) + " g");
        accelZText.setText(  "ACCEL Z  : " + String.valueOf(azValue) + " g");
        vbatText.setText(    "VBAT     : " + String.valueOf(vbatValue) + " V");
        vgenText.setText(    "VGEN     : " + String.valueOf(vgenValue) + " V");

        // Calculation of angles
        //pitchValue = Math.atan(axValue/(Math.sqrt(Math.pow(ayValue, 2) +
        Math.pow(azValue, 2))));

        //double angle = atan2(ayValue, axValue); // Calcul de l'angle en radians
```

```
382             //double angle = atan(axValue/ayValue);
383             // Conversion de l'angle en degrés et en float
384             //float pitchValue = (float) toDegrees(angle);
385 
386             // Calcul des angles
387             double roll = Math.atan2(ayValue, azValue);
388             double pitch = Math.atan2(-axValue, Math.sqrt(ayValue * ayValue + azValue *
                azValue));
389             double yaw = Math.atan2(Math.sin(roll) * axValue - Math.cos(roll) * ayValue, Math
                .cos(roll) * azValue);
390 
391             // Conversion des angles en degrés
392             //roll = Math.toDegrees(roll);
393             pitch = Math.toDegrees(pitch);
394             yaw = Math.toDegrees(yaw);
395 
396             pitchText.setText(   "PITCH    : " + String.format("%.2f", pitch) + " °");
397             yawText.setText(   "YAW        : " + String.format("%.2f", yaw) + " °");
398             //rollText.setText(   "ROLL     : " + String.format("%.2f", roll) + " °");
399 
400 
401             /*
402             receiveText.append("Speed = " + sValue + " km/h" + " GyroX = " + gxValue + "
                GyroY = " + gyValue +
403                     " GyroZ = " + gzValue + " AccelX = " + axValue  + " AccelY = " + ayValue
                    +
404                     " AccelZ = " + azValue  + " Vbat = " + vbatValue  + " Vgen = " +
                    vgenValue + '\n');
405             */
406 
407 
                //-------------------------------------------------------------------------------
                ---------//
408 
409             //receiveText.append(spn);
410         }
411 
412 
413         private void status(String str) {
414             SpannableStringBuilder spn = new SpannableStringBuilder(str + '\n');
415             spn.setSpan(new ForegroundColorSpan(getResources().getColor(R.color.
                colorStatusText)), 0, spn.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
416             //receiveText.append(spn);
417         }
418 
419         /*
420          * SerialListener
421          */
422         @Override
423         public void onSerialConnect() {
424             status("connected");
425             connected = Connected.True;
426         }
427 
428         @Override
429         public void onSerialConnectError(Exception e) {
430             status("connection failed: " + e.getMessage());
431             disconnect();
432         }
433 
434         @Override
435         public void onSerialRead(byte[] data) {
436             ArrayDeque<byte[]> datas = new ArrayDeque<>();
437             datas.add(data);
438             receive(datas);
439         }
```

```
440
441         public void onSerialRead(ArrayDeque<byte[]> datas) {
442             receive(datas);
443         }
444
445         @Override
446         public void onSerialIoError(Exception e) {
447             status("connection lost: " + e.getMessage());
448             disconnect();
449         }
450
451     }
452
```