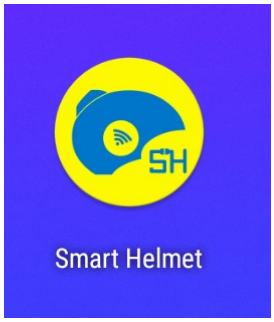
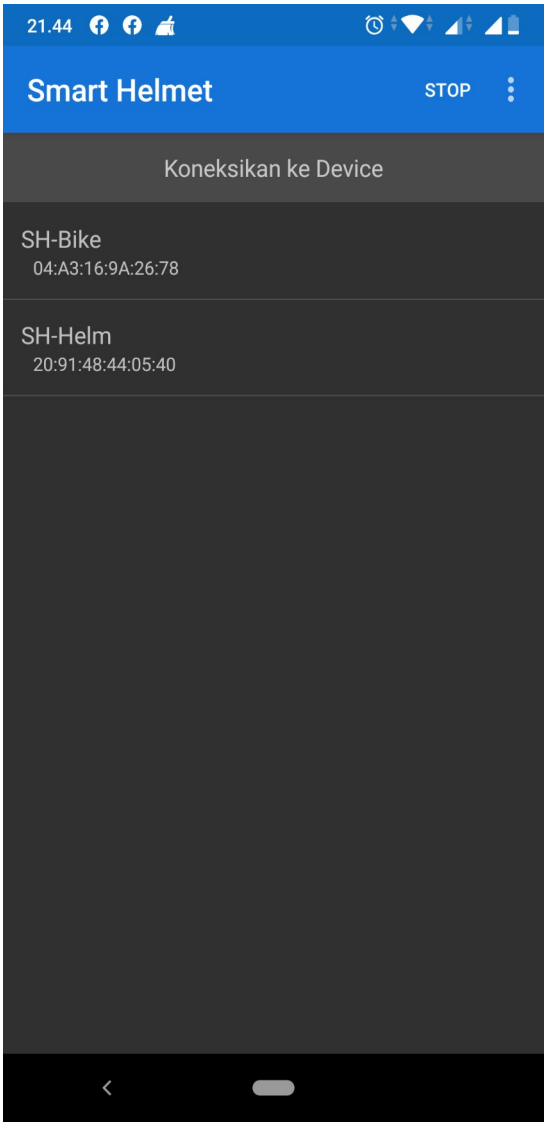
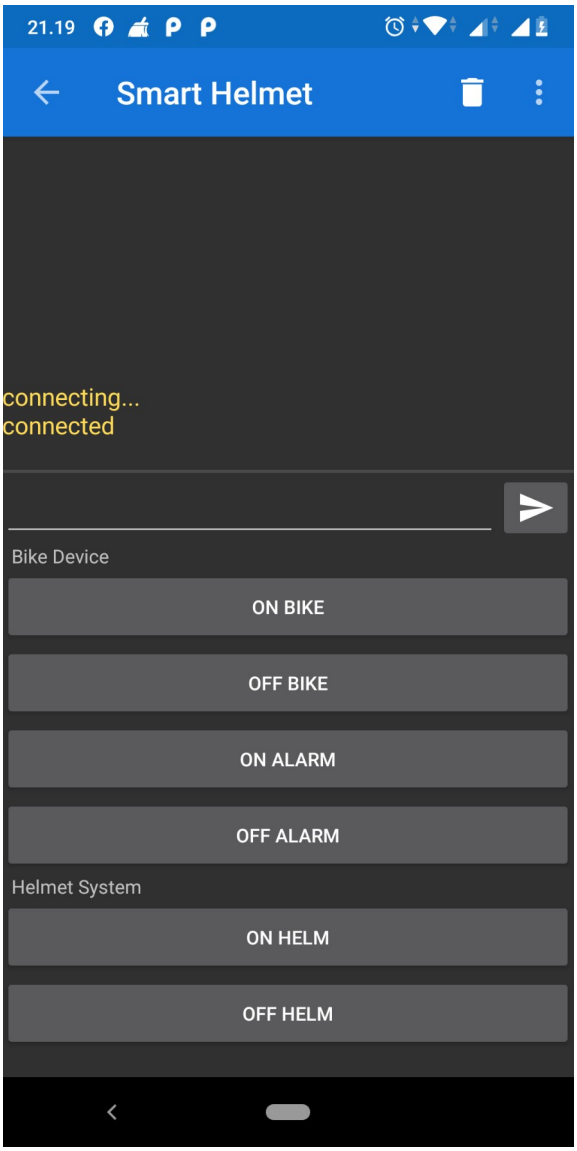
A. CARA PENGGUNAAN

Aplikasi Smart Helmet merupakan perangkat lunak berbasis android yang digunakan untuk mengendalikan perangkat keras yang dipasang atau letakkan pada helm dan sepeda motor dengan sistem antarmuka ***bluetooth.***

	Gambar Ikon aplikasi Smart Helmet. Klik untuk memulai menggunakan.
	Halaman utama aplikasii. Aktifkan bluetooth pada smartphone, kemudian klik **Scan** untuk melakukan pencarian perangkat keras smart helmet.



Tampilan hasil **Scan.**
Pilihlah perangkat yang ingin dikendalikan.
SH-Bike: perangkat pada sepeda motor
SH-Helm: perangkat pada Helm motor



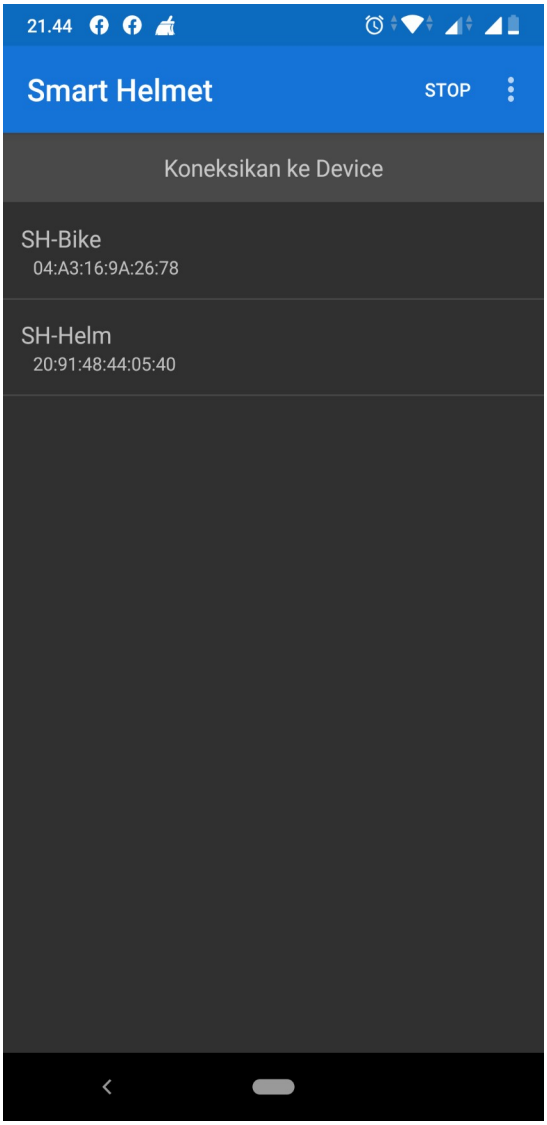
Tampilan kendali untuk perangkat pada SH-Bike dan SH-Helm.
ON BIKE: Menyalakan fungsi perangkat keras pada sepeda motor sehingga sistem keamanan posisi on
OFF BIKE: Mematikan fungsi perangkat keras pada sepeda motor sehingga sistem keamanan posisi off
ON ALARM: Menyalakan fungsi alarm pada sepeda motor
OFF ALARM: Mematikan fungsi alarm pada sepeda motor
ON HELM: Menyalakan fungsi perangkat helm sehingga sistem keamanan helm posisi on
OFF HELM: Mematikan fungsi perangkat helm, sehingga sistem keamanan helm posisi off

B. KODE SUMBER

Listing program Menu Scan Device

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/ble_scan"
        android:title="SCAN"
        app:showAsAction="always" />
    <item
        android:id="@+id/ble_scan_stop"
        android:title="STOP"
        app:showAsAction="always"
        android:visible="false" />
    <item
        android:id="@+id/bt_settings"
        android:title="Bluetooth settings" />
</menu>
```

Tampilan Scan Device



Listing Data Menu Scan

```
package bens.SmartHelmet.SH;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.location.LocationManager;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.ListFragment;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.Collections;

/**
 * show list of BLE devices
 */
public class DevicesFragment extends ListFragment {

    private enum ScanState { NONE, LE_SCAN, DISCOVERY, DISCOVERY_FINISHED }
    private ScanState scanState = ScanState.NONE;
    private static final long LE_SCAN_PERIOD = 10000; // similar to
bluetoothAdapter.startDiscovery
    private final Handler leScanStopHandler = new Handler();
    private final BluetoothAdapter.LeScanCallback leScanCallback;
    private final BroadcastReceiver discoveryBroadcastReceiver;
    private final IntentFilter discoveryIntentFilter;

    private Menu menu;
    private BluetoothAdapter bluetoothAdapter;
    private final ArrayList<BluetoothDevice> listItems = new ArrayList<>();
    private ArrayAdapter<BluetoothDevice> listAdapter;

    public DevicesFragment() {
        leScanCallback = (device, rssi, scanRecord) -> {
            if(device != null && getActivity() != null) {
```

```
                getActivity().runOnUiThread(() -> { updateScan(device); });
            }
        };
        discoveryBroadcastReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                if(BluetoothDevice.ACTION_FOUND.equals(intent.getAction())) {
                    BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                    if(device.getType() != BluetoothDevice.DEVICE_TYPE_CLASSIC &&
getActivity() != null) {
                        getActivity().runOnUiThread(() -> updateScan(device));
                    }
                }

if(BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(intent.getAction())) {
                    scanState = ScanState.DISCOVERY_FINISHED; // don't cancel again
                    stopScan();
                }
            }
        };
        discoveryIntentFilter = new IntentFilter();
        discoveryIntentFilter.addAction(BluetoothDevice.ACTION_FOUND);
        discoveryIntentFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setHasOptionsMenu(true);

if(getActivity().getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOTH)
)
            bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        listAdapter = new ArrayAdapter<BluetoothDevice>(getActivity(), 0, listItems) {
            @NonNull
            @Override
            public View getView(int position, View view, @NonNull ViewGroup parent) {
                BluetoothDevice device = listItems.get(position);
                if (view == null)
                    view =
getActivity().getLayoutInflater().inflate(R.layout.device_list_item, parent, false);
                TextView text1 = view.findViewById(R.id.text1);
                TextView text2 = view.findViewById(R.id.text2);
                if(device.getName() == null || device.getName().isEmpty())
                    text1.setText("<unnamed>");
                else
                    text1.setText(device.getName());
                text2.setText(device.getAddress());
                return view;
            }
        };
    }

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        setListAdapter(null);
        View header =
getActivity().getLayoutInflater().inflate(R.layout.device_list_header, null, false);
        getListView().addHeaderView(header, null, false);
```

```
        setEmptyText("initializing...");
        ((TextView) getListView().getEmptyView()).setTextSize(18);
        setListAdapter(listAdapter);
    }

    @Override
    public void onCreateOptionsMenu(@NonNull Menu menu, MenuInflater inflater) {
        inflater.inflate(R.menu.menu_devices, menu);
        this.menu = menu;
        if (bluetoothAdapter == null) {
            menu.findItem(R.id.bt_settings).setEnabled(false);
            menu.findItem(R.id.ble_scan).setEnabled(false);
        } else if(!bluetoothAdapter.isEnabled()) {
            menu.findItem(R.id.ble_scan).setEnabled(false);
        }
    }

    @Override
    public void onResume() {
        super.onResume();
        getActivity().registerReceiver(discoveryBroadcastReceiver,
discoveryIntentFilter);
        if(bluetoothAdapter == null) {
            setEmptyText("<bluetooth LE not supported>");
        } else if(!bluetoothAdapter.isEnabled()) {
            setEmptyText("<bluetooth is disabled>");
            if (menu != null) {
                listItems.clear();
                listAdapter.notifyDataSetChanged();
                menu.findItem(R.id.ble_scan).setEnabled(false);
            }
        } else {
            setEmptyText("<use SCAN to refresh devices>");
            if (menu != null)
                menu.findItem(R.id.ble_scan).setEnabled(true);
        }
    }

    @Override
    public void onPause() {
        super.onPause();
        stopScan();
        getActivity().unregisterReceiver(discoveryBroadcastReceiver);
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        menu = null;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.ble_scan) {
            startScan();
            return true;
        } else if (id == R.id.ble_scan_stop) {
            stopScan();
            return true;
        } else if (id == R.id.bt_settings) {
```

```
            Intent intent = new Intent();
            intent.setAction(android.provider.Settings.ACTION_BLUETOOTH_SETTINGS);
            startActivity(intent);
            return true;
        } else {
            return super.onOptionsItemSelected(item);
        }
    }

    @SuppressLint("StaticFieldLeak") // AsyncTask needs reference to this fragment
    private void startScan() {
        if(scanState != ScanState.NONE)
            return;
        scanState = ScanState.LE_SCAN;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if
(getActivity().checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                scanState = ScanState.NONE;
                AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
                builder.setTitle(R.string.location_permission_title);
                builder.setMessage(R.string.location_permission_message);
                builder.setPositiveButton(android.R.string.ok,
                        (dialog, which) -> requestPermissions(new String[]
{Manifest.permission.ACCESS_FINE_LOCATION}, 0));
                builder.show();
                return;
            }
            LocationManager locationManager = (LocationManager)
getActivity().getSystemService(Context.LOCATION_SERVICE);
            boolean         locationEnabled = false;
            try {
                locationEnabled =
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
            } catch(Exception ignored) {}
            try {
                locationEnabled |=
locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
            } catch(Exception ignored) {}
            if(!locationEnabled)
                scanState = ScanState.DISCOVERY;
            // Starting with Android 6.0 a bluetooth scan requires
ACCESS_COARSE_LOCATION permission, but that's not all!
            // LESCAN also needs enabled 'location services', whereas DISCOVERY works
without.
            // Most users think of GPS as 'location service', but it includes more, as
we see here.
            // Instead of asking the user to enable something they consider unrelated,
            // we fall back to the older API that scans for bluetooth classic _and_ LE
            // sometimes the older API returns less results or slower
        }
        listItems.clear();
        listAdapter.notifyDataSetChanged();
        setEmptyText("<scanning...>");
        menu.findItem(R.id.ble_scan).setVisible(false);
        menu.findItem(R.id.ble_scan_stop).setVisible(true);
        if(scanState == ScanState.LE_SCAN) {
            leScanStopHandler.postDelayed(this::stopScan, LE_SCAN_PERIOD);
            new AsyncTask<Void, Void, Void>() {
                @Override
                protected Void doInBackground(Void[] params) {
                    bluetoothAdapter.startLeScan(null, leScanCallback);
```

```
                    return null;
                }
            }.execute(); // start async to prevent blocking UI, because startLeScan
sometimes take some seconds
        } else {
            bluetoothAdapter.startDiscovery();
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        // ignore requestCode as there is only one in this fragment
        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            new Handler(Looper.getMainLooper()).postDelayed(this::startScan,1); // run
after onResume to avoid wrong empty-text
        } else {
            AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
            builder.setTitle(getText(R.string.location_denied_title));
            builder.setMessage(getText(R.string.location_denied_message));
            builder.setPositiveButton(android.R.string.ok, null);
            builder.show();
        }
    }

    private void updateScan(BluetoothDevice device) {
        if(scanState == ScanState.NONE)
            return;
        if(!listItems.contains(device)) {
            listItems.add(device);
            Collections.sort(listItems, DevicesFragment::compareTo);
            listAdapter.notifyDataSetChanged();
        }
    }

    private void stopScan() {
        if(scanState == ScanState.NONE)
            return;
        setEmptyText("<no bluetooth devices found>");
        if(menu != null) {
            menu.findItem(R.id.ble_scan).setVisible(true);
            menu.findItem(R.id.ble_scan_stop).setVisible(false);
        }
        switch(scanState) {
            case LE_SCAN:
                leScanStopHandler.removeCallbacks(this::stopScan);
                bluetoothAdapter.stopLeScan(leScanCallback);
                break;
            case DISCOVERY:
                bluetoothAdapter.cancelDiscovery();
                break;
            default:
                // already canceled
        }
        scanState = ScanState.NONE;

    }

    @Override
    public void onListItemClick(@NonNull ListView l, @NonNull View v, int position,
```

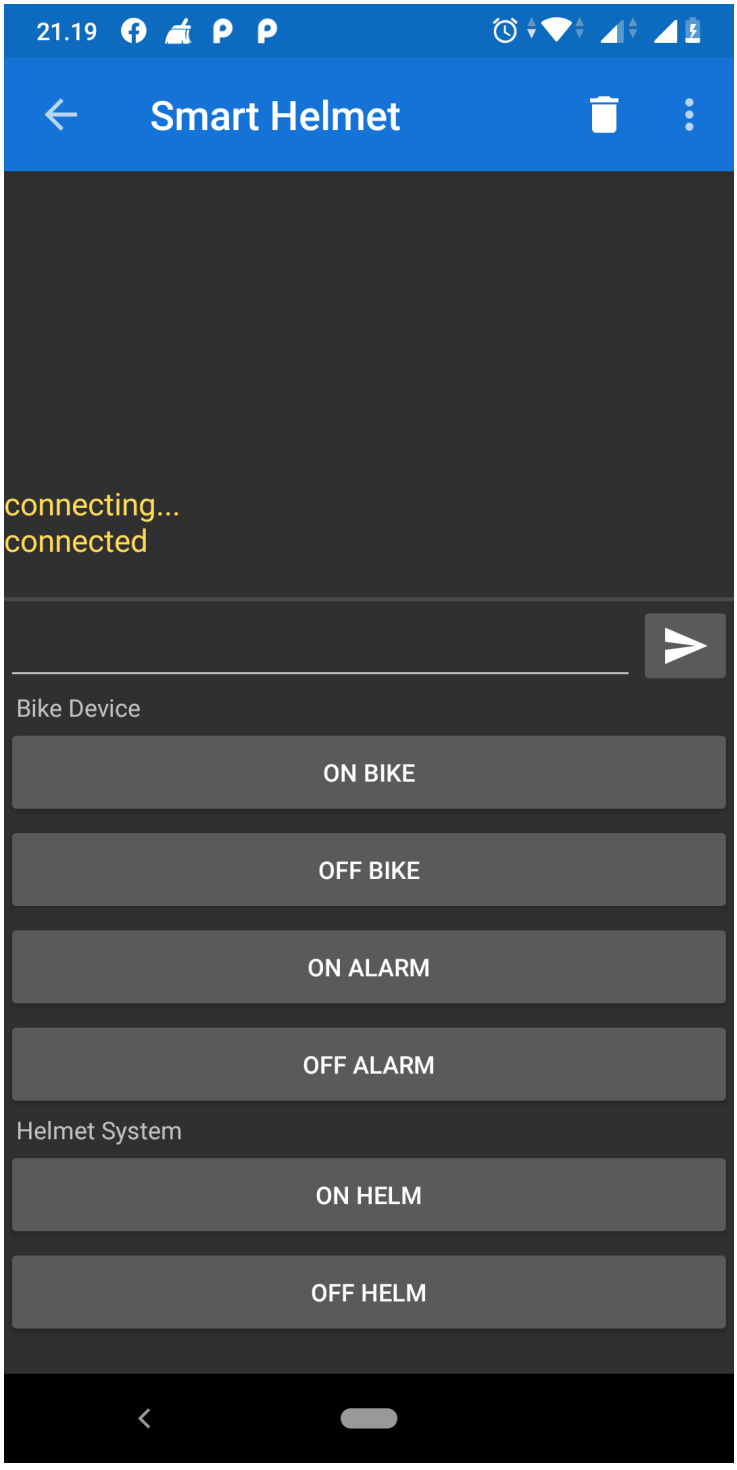
```
long id) {
        stopScan();
        BluetoothDevice device = listItems.get(position-1);
        Bundle args = new Bundle();
        args.putString("device", device.getAddress());
        Fragment fragment = new TerminalFragment();
        fragment.setArguments(args);
        getFragmentManager().beginTransaction().replace(R.id.fragment, fragment,
"terminal").addToBackStack(null).commit();
    }

    /**
     * sort by name, then address. sort named devices first
     */
    static int compareTo(BluetoothDevice a, BluetoothDevice b) {
        boolean aValid = a.getName()!=null && !a.getName().isEmpty();
        boolean bValid = b.getName()!=null && !b.getName().isEmpty();
        if(aValid && bValid) {
            int ret = a.getName().compareTo(b.getName());
            if (ret != 0) return ret;
            return a.getAddress().compareTo(b.getAddress());
        }
        if(aValid) return -1;
        if(bValid) return +1;
        return a.getAddress().compareTo(b.getAddress());
    }
}
```

Listing Program Menu Kendali

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/receive_text"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:freezesText="true"
        android:gravity="bottom"
        android:scrollbars="vertical"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium" />

    <View
        android:layout_width="match_parent"
        android:layout_height="2dp"
        android:background="?android:attr/listDivider" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <EditText
            android:id="@+id/send_text"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:inputType="text|textNoSuggestions"
            android:singleLine="true" />

        <ImageButton
            android:id="@+id/send_btn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:srcCompat="@drawable/ic_send_white_24dp" />
    </LinearLayout>

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="  Bike Device" />

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="ON Bike" />
```

```
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="OFF Bike" />

    <Button
        android:id="@+id/button3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="ON Alarm" />

    <Button
        android:id="@+id/button4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="OFF Alarm" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="  Helmet System" />

    <Button
        android:id="@+id/button5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="ON Helm" />

    <Button
        android:id="@+id/button6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="OFF Helm" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="" />
</LinearLayout>
```

Tampilan Menu Kendali Smart Helmet



Listing data komunikasi bluetooth Aplikasi ke Hardware

```
package bens.SmartHelmet.SH;

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.Editable;
import android.text.Spannable;
import android.text.SpannableStringBuilder;
import android.text.method.ScrollingMovementMethod;
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;

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 sendText;
    private TextUtil.HexWatcher hexWatcher;

    private Connected connected = Connected.False;
    private boolean initialStart = true;
    private boolean hexEnabled = false;
    private boolean pendingNewline = false;
    private String newline = TextUtil.newline_crlf;
    //================================

    private String Bike_ON = "3";
    private String Bike_OFF = "2";
    private String Alarm_ON = "8";
```

```
    private String Alarm_OFF = "9";

    private String Helm_ON = "A";
    private String Helm_OFF = "B";
    /*
     * Lifecycle
     */
    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setHasOptionsMenu(true);
        setRetainInstance(true);
        deviceAddress = getArguments().getString("device");
    }

    @Override
    public void onDestroy() {
        if (connected != Connected.False)
            disconnect();
        getActivity().stopService(new Intent(getActivity(), SerialService.class));
        super.onDestroy();
    }

    @Override
    public void onStart() {
        super.onStart();
        if(service != null)
            service.attach(this);
        else
            getActivity().startService(new Intent(getActivity(), SerialService.class));
// prevents service destroy on unbind from recreated activity caused by orientation
change
    }

    @Override
    public void onStop() {
        if(service != null && !getActivity().isChangingConfigurations())
            service.detach();
        super.onStop();
    }

    @SuppressWarnings("deprecation") // onAttach(context) was added with API 23.
onAttach(activity) works for all API versions
    @Override
    public void onAttach(@NonNull Activity activity) {
        super.onAttach(activity);
        getActivity().bindService(new Intent(getActivity(), SerialService.class), this,
Context.BIND_AUTO_CREATE);
    }

    @Override
    public void onDetach() {
        try { getActivity().unbindService(this); } catch(Exception ignored) {}
        super.onDetach();
    }

    @Override
    public void onResume() {
        super.onResume();
        if(initialStart && service != null) {
```

```
            initialStart = false;
            getActivity().runOnUiThread(this::connect);
        }
    }

    @Override
    public void onServiceConnected(ComponentName name, IBinder binder) {
        service = ((SerialService.SerialBinder) binder).getService();
        service.attach(this);
        if(initialStart && isResumed()) {
            initialStart = false;
            getActivity().runOnUiThread(this::connect);
        }
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {
        service = null;
    }

    /*
     * UI
     */
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_terminal, container, false);
        receiveText = view.findViewById(R.id.receive_text);                          //
TextView performance decreases with number of spans
        receiveText.setTextColor(getResources().getColor(R.color.colorRecieveText)); //
set as default color to reduce number of spans
        receiveText.setMovementMethod(ScrollingMovementMethod.getInstance());

        sendText = view.findViewById(R.id.send_text);
        hexWatcher = new TextUtil.HexWatcher(sendText);
        hexWatcher.enable(hexEnabled);
        sendText.addTextChangedListener(hexWatcher);
        sendText.setHint(hexEnabled ? "HEX mode" : "");

        View sendBtn = view.findViewById(R.id.send_btn);
        sendBtn.setOnClickListener(v -> send(sendText.getText().toString()));

        View bikeon = view.findViewById(R.id.button);
        bikeon.setOnClickListener(v -> send(Bike_ON));

        View bikeoff = view.findViewById(R.id.button2);
        bikeoff.setOnClickListener(v -> send(Bike_OFF));

        View alarmon = view.findViewById(R.id.button3);
        alarmon.setOnClickListener(v -> send(Alarm_ON));

        View alarmoff = view.findViewById(R.id.button4);
        alarmoff.setOnClickListener(v -> send(Alarm_OFF));

        View helmon = view.findViewById(R.id.button5);
        helmon.setOnClickListener(v -> send(Helm_ON));

        View helmoff = view.findViewById(R.id.button6);
        helmoff.setOnClickListener(v -> send(Helm_OFF));
```

```
        return view;
    }

    @Override
    public void onCreateOptionsMenu(@NonNull Menu menu, MenuInflater inflater) {
        inflater.inflate(R.menu.menu_terminal, menu);
        menu.findItem(R.id.hex).setChecked(hexEnabled);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.clear) {
            receiveText.setText("");
            return true;
        } else if (id == R.id.newline) {
            String[] newlineNames =
getResources().getStringArray(R.array.newline_names);
            String[] newlineValues =
getResources().getStringArray(R.array.newline_values);
            int pos = java.util.Arrays.asList(newlineValues).indexOf(newline);
            AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
            builder.setTitle("Newline");
            builder.setSingleChoiceItems(newlineNames, pos, (dialog, item1) -> {
                newline = newlineValues[item1];
                dialog.dismiss();
            });
            builder.create().show();
            return true;
        } else if (id == R.id.hex) {
            hexEnabled = !hexEnabled;
            sendText.setText("");
            hexWatcher.enable(hexEnabled);
            sendText.setHint(hexEnabled ? "HEX mode" : "");
            item.setChecked(hexEnabled);
            return true;
        } else {
            return super.onOptionsItemSelected(item);
        }
    }

    /*
     * Serial + UI
     */
    private void connect() {
        try {
            BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
            BluetoothDevice device = bluetoothAdapter.getRemoteDevice(deviceAddress);
            status("connecting...");
            connected = Connected.Pending;
            SerialSocket socket = new
SerialSocket(getActivity().getApplicationContext(), device);
            service.connect(socket);
        } catch (Exception e) {
            onSerialConnectError(e);
        }
    }

    private void disconnect() {
        connected = Connected.False;
```

```
        service.disconnect();
    }

    private void send(String str) {
        if(connected != Connected.True) {
            Toast.makeText(getActivity(), "not connected", Toast.LENGTH_SHORT).show();
            return;
        }
        try {
            String msg;
            byte[] data;
            if(hexEnabled) {
                StringBuilder sb = new StringBuilder();
                TextUtil.toHexString(sb, TextUtil.fromHexString(str));
                TextUtil.toHexString(sb, newline.getBytes());
                msg = sb.toString();
                data = TextUtil.fromHexString(msg);
            } else {
                msg = str;
                data = (str + newline).getBytes();
            }
            SpannableStringBuilder spn = new SpannableStringBuilder(msg + '\n');
            spn.setSpan(new
ForegroundColorSpan(getResources().getColor(R.color.colorSendText)), 0, spn.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
            receiveText.append(spn);
            service.write(data);
        } catch (Exception e) {
            onSerialIoError(e);
        }
    }

    private void receive(byte[] data) {
        if(hexEnabled) {
            receiveText.append(TextUtil.toHexString(data) + '\n');
        } else {
            String msg = new String(data);
            if(newline.equals(TextUtil.newline_crlf) && msg.length() > 0) {
                // don't show CR as ^M if directly before LF
                msg = msg.replace(TextUtil.newline_crlf, TextUtil.newline_lf);
                // special handling if CR and LF come in separate fragments
                if (pendingNewline && msg.charAt(0) == '\n') {
                    Editable edt = receiveText.getEditableText();
                    if (edt != null && edt.length() > 1)
                        edt.replace(edt.length() - 2, edt.length(), "");
                }
                pendingNewline = msg.charAt(msg.length() - 1) == '\r';
            }
            receiveText.append(TextUtil.toCaretString(msg, newline.length() != 0));
        }
    }

    private void status(String str) {
        SpannableStringBuilder spn = new SpannableStringBuilder(str + '\n');
        spn.setSpan(new
ForegroundColorSpan(getResources().getColor(R.color.colorStatusText)), 0, spn.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
        receiveText.append(spn);
    }

    /*
```

```
     * SerialListener
     */
    @Override
    public void onSerialConnect() {
        status("connected");
        connected = Connected.True;
    }

    @Override
    public void onSerialConnectError(Exception e) {
        status("Gagal terkoneksi: " + e.getMessage());
        disconnect();
    }

    @Override
    public void onSerialRead(byte[] data) {
        receive(data);
    }

    @Override
    public void onSerialIoError(Exception e) {
        status("Koneksi terputus: " + e.getMessage());
        disconnect();
    }

}
```