APPENDIX-A

Flutter Mobile App Codes

While the codes are in the left columns of the tables below, the file in which it is included and the explanations in the right columns.

CODE	EXPLAINATION

`import 'package:flutter/material.dart';`
`import 'HomePage.dart';`
 	
`void main() {`
`  runApp(`
`  MaterialApp(`
`  debugShowCheckedModeBanner: false,`
`  initialRoute: "/",`
`  routes: {`
`  "/": (context) => HomePage(),`
`  },`
`  ),`
`  );`
`}`
	Main.dart
	
	In this part, Mobile app routing system is done, firstly “/” comes and HomePage widget works.

`import 'package:google_fonts/google_fonts.dart';`
`import 'CameraComponent.dart';`
`import 'dart:convert';`
`import 'package:flutter/material.dart';`
`import 'package:flutter/services.dart';`
`import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';`
 	
`class HomePage extends StatefulWidget {`
`  @override`
`  _HomePageState createState() => _HomePageState();`
`}`
 	
`class _HomePageState extends State<HomePage> {`
`  void _portraitModeOnly() {`
`  SystemChrome.setPreferredOrientations(`
`  [DeviceOrientation.portraitDown, DeviceOrientation.portraitUp]);`
`  }`
 	
`  // Initializing the Bluetooth connection state to be unknown`
`  BluetoothState _bluetoothState = BluetoothState.UNKNOWN;`
`  // Get the instance of the Bluetooth`
`  FlutterBluetoothSerial _bluetooth = FlutterBluetoothSerial.instance;`
`  // Track the Bluetooth connection with the remote device`
`  BluetoothConnection connection;`
`  // To track whether the device is still connected to Bluetooth`
`  bool get isConnected => connection != null && connection.isConnected;`
`  List<BluetoothDevice> _devicesList = [];`
`  bool isDisconnecting = false;`
`  bool _connected = false;`
`  BluetoothDevice _device;`
	Homepage.dart
	
	HomePage widget is a stateful widget because bluetooth connections are done here and it requires some variables like _bluetoothState.

```dart

  @override
  void initState() {
    super.initState();

    // Get current state
    FlutterBluetoothSerial.instance.state.then((state) {
      setState(() {
        _bluetoothState = state;
      });
    });

    // If the Bluetooth of the device is not enabled,
    // then request permission to turn on Bluetooth
    // as the app starts up
    enableBluetooth();

    // Listen for further state changes
    FlutterBluetoothSerial.instance
        .onStateChanged()
        .listen((BluetoothState state) {
      setState(() {
        _bluetoothState = state;

        // For retrieving the paired devices list
        getPairedDevices();
      });
    });
  }

  Future<void> enableBluetooth() async {
    // Retrieving the current Bluetooth state
    _bluetoothState = await FlutterBluetoothSerial.instance.state;

    // If the Bluetooth is off, then turn it on first
    // and then retrieve the devices that are paired.
    if (_bluetoothState == BluetoothState.STATE_OFF) {
      await FlutterBluetoothSerial.instance.requestEnable();
      await getPairedDevices();
      return true;
    } else {
      await getPairedDevices();
    }
    return false;
  }
```

Homepage.dart

In this part, HomePage widget constructor part and when first initializing , using enableBluetooth function works and the app requires the user to open a bluetooth connection.

```dart

  Future<void> getPairedDevices() async {
    List<BluetoothDevice> devices = [];

    // To get the list of paired devices
    try {
      devices = await _bluetooth.getBondedDevices();
    } on PlatformException {
      print("Error");
    }

    // It is an error to call [setState] unless [mounted] is true.
    if (!mounted) {
      return;
    }

    // Store the [devices] list in the [_devicesList] for accessing
    // the list outside this class
    setState(() {
      _devicesList = devices;
    });
  }

  List<DropdownMenuItem<BluetoothDevice>> _getDeviceItems() {
    List<DropdownMenuItem<BluetoothDevice>> items = [];
    if (_devicesList.isEmpty) {
      items.add(DropdownMenuItem(
        child: Text(
          'NONE',
          style: TextStyle(color: Colors.white),
        ),
      ));
    } else {
      _devicesList.forEach((device) {
        items.add(DropdownMenuItem(
          child: Text(
            device.name,
            style: TextStyle(color: Colors.white),
          ),
          value: device,
        ));
      });
    }
    return items;
  }
```

Homepage.dart

In this part, in order to decrease the complexity of code , functions which will be used in the build part, were written here. getPairedDevices function and getDeviceItems are included..

```dart
void _connect() async {
  if (_device == null) {
    print('No device selected');
  } else {
    if (!isConnected) {
      // Trying to connect to the device using
      // its address
      await BluetoothConnection.toAddress(_device.address)
          .then((_connection) {
        print('Connected to the device');
        connection = _connection;

        // Updating the device connectivity
        // status to [true]
        setState(() {
          _connected = true;
        });

        // This is for tracking when the disconnecting process
        // is in progress which uses the [isDisconnecting] variable
        // defined before.
        // Whenever we make a disconnection call, this [onDone]
        // method is fired.
        connection.input.listen(null).onDone(() {
          if (isDisconnecting) {
            print('Disconnecting locally!');
          } else {
            print('Disconnected remotely!');
          }
          if (this.mounted) {
            setState(() {});
          }
        });
      }).catchError((error) {
        print('Cannot connect, exception occurred');
        print(error);
      });
      print('Device connected');
    }
  }
}
```

Homepage.dart

In this part, in order to decrease the complexity of code , functions which will be used in the build part, were written here. Connect function is included.

```dart
  void _disconnect() async {
    // Closing the Bluetooth connection
    await connection.close();
    print('Device disconnected');

    // Update the [_connected] variable
    if (!connection.isConnected) {
      setState(() {
        _connected = false;
      });
    }
  }

  void _sendMessageToBluetooth(String val) async {
    connection.output.add(utf8.encode(val + "\r\n"));
    await connection.output.allSent;
  }

  @override
  void dispose() {
    if (isConnected) {
      isDisconnecting = true;
      connection.dispose();
      connection = null;
    }

    super.dispose();
  }

```

Homepage.dart

In this part, in order to decrease the complexity of code , functions which will be used in the build part, were written here.
Disconnect and send messages to bluetooth functions are included.

```dart
@override
Widget build(BuildContext context) {
  _portraitModeOnly();
  return SafeArea(
    child: Scaffold(
      appBar: new AppBar(
        backgroundColor: Color(0XFF2e2e2e),

        actions: [
          Switch(
            value: _bluetoothState.isEnabled,
            onChanged: (bool value) {
              future() async {
                if (value) {
                  // Enable Bluetooth
                  await FlutterBluetoothSerial.instance.requestEnable();
                } else {
                  // Disable Bluetooth
                  await FlutterBluetoothSerial.instance.requestDisable();
                }

                // In order to update the devices list
                await getPairedDevices();

                // Disconnect from any device before
                // turning off Bluetooth
                if (_connected) {
                  _disconnect();
                }
              }

              future().then((_) {
                setState(() {});
              });
            },
          ),
```

Homepage.dart

In this part, By using the portraidModeOnly function, it is ensured that the phone works only in landscape mode. Also bluetooth connections and listing paired devices is done.

```
            DropdownButton(
              items: _getDeviceItems(),
              onChanged: (value) => setState(() => _device = value),
              value: _devicesList.isNotEmpty ? _device : null,
              dropdownColor: Color(0XFF2e2e2e),

            ),
            RaisedButton(
              color: Colors.redAccent,
              onPressed: _connected ? _disconnect : _connect,
              child: Text(_connected ? 'Disconnect' : 'Connect',style: TextStyle(co
            ),
          ],
        ),
        body: SafeArea(
          child: Column(
            children: [
              Expanded(
                flex: 6,
                child: Container(
                  color: Color(0XFF2e2e2e), child: CameraComponent(),),
              ),
              Expanded(
                flex: 2,
                child: Container(
                  padding: EdgeInsets.all(10.0),
                  color: Color(0XFF2e2e2e),
                  child: Row(
                    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                    children: [
                      ControllerButton(
                        child: Icon(Icons.keyboard_arrow_left,
                            size: 20, color: Colors.white54),
                        onPressed: () => _sendMessageToBluetooth("3"),
                      ),
                      ControllerButton(
                        child: Icon(Icons.keyboard_arrow_right,
                            size: 20, color: Colors.white54),
                        onPressed: () => _sendMessageToBluetooth("4"),
                      ),
                      ControllerButton(
                        child: Icon(Icons.keyboard_arrow_up,
                            size: 20, color: Colors.white54),
                        onPressed: () => _sendMessageToBluetooth("1"),
                      ),
                      ControllerButton(
                        child: Icon(Icons.keyboard_arrow_down,
                            size: 20, color: Colors.white54),
                        onPressed: () => _sendMessageToBluetooth("2"),
                      ),
```

Homepage.dart

In this part,Button widgets and what actions should be performed when they are pressed are written.

```dart
import 'package:flutter/material.dart';
import 'package:webview_flutter/webview_flutter.dart';

class CameraComponent extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return WebView(
      initialUrl: "http://192.168.43.165/",
      javascriptMode: JavascriptMode.unrestricted,

    );
  }
}
```

CameraComponent.dart

In this stateless component, the WebView widget is used by the webview plugin. This widget provides opening a url in the mobile app. Camera web server is running on 192.168.43.165, so it is used as an initialUrl.