BBM460 Wireless and Mobile Networks Laboratory Final Report Bluetooth RC Car with Wi-Fi Camera

Muhammed Said Kaya, Mert Çökelek [1]
[1]Hacettepe University Computer Science Dept.

1504 (errors:3) Words Student ID:21627428, 21727082
Keywords: Bluetooth, Wi-Fi, Remote Controlled (RC) Car 29th December, 2020

Abstract

The main motivation behind this project is to build a remote-controlled car using Bluetooth and having live video surveillance with Wi-Fi technologies to experience these most commonly used two wireless connection types. This RC Car prototype can further be adapted to short-range military applications, daily-life assistant applications, and entertainment fields [1].

With advances in technology over the years, it is possible to remotely monitor areas of importance by using robots in place of humans, and in this context, we developed an RC Car with Bluetooth and Wi-Fi, for controlling and video surveillance purposes, respectively.

1 Introduction

As the final project for our BBM 460 - Wireless and Mobile Networks Laboratory, we decided to develop a remote-controlled car for video surveillance. This car is controlled by an Android phone, and the visual data is viewed on any web browser connected to the same network as the car.

For this purpose, we have used Arduino UNO RC3, HC-05 Bluetooth Module, ESP32 Wireless Camera Module, L298N Motor Driver, DC motors, breadboard, car chases, wheels, 9V battery, and jumpers. We have written the Arduino code in Arduino IDE on Linux and prepared the Flutter mobile app in Android Studio on Linux. The design and implementation details will be covered in the next sections.

2 Explanation of Technologies

2.1 Bluetooth

Bluetooth [2] is a wireless technology standard used for exchanging data between fixed and mobile devices over short distances using UHF radio waves in the industrial, scientific and medical radio bands, from 2.402 GHz to 2.480 GHz, and building personal area networks (PANs).

2.2 Wi-Fi

Wi-Fi [3] is a family of wireless network protocols, based on the IEEE 802.11 family of standards that are commonly used for local area networking of devices and Internet access.

3 Explanation of Components

3.1 Flutter Mobile App

Flutter is a free and open-source mobile UI framework created by Google and released in May 2017. In a few words, it allows you to create a native mobile application with only one codebase. This means that you can use one programming language and one codebase to create two different apps (for iOS and Android).[5]

Utilizing Flutter, The mobile app was created.The bluetooth connection between car and mobile app provided and in this way the forward, back, right, left, and stop commands sends to the car. Using Webview Plugin, the camera interface can be seen in the white space below.

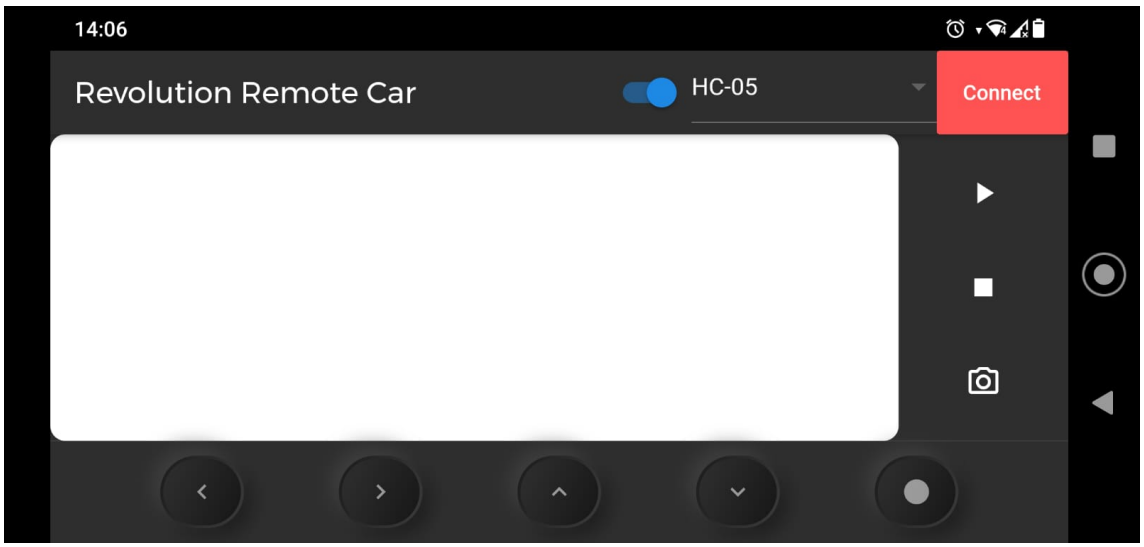

Fig. 1. Mobile App View

3.2 HC-05 Bluetooth Module

HC-05 Bluetooth Module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Its communication is via serial communication which makes an easy way to interface with controller or PC. [6]

- **Type**: Bluetooth Module
- Purpose: Provides wireless one way synchronized transmission between a cellphone and Arduino Uno single-board computer
- **Function**: Turn On/Off , Connect
- **Subordinates**
  - Arduino UNO
- **Interfaces**
  - **In:** Analog Signal, Digital Data, Flutter App
  - **Out:** Digital Signal, Digital Data, Arduino UNO

3.3 Arduino UNO

Arduino [4] is a microcontroller board based on the ATmega328P. It contains everything needed to support the microcontroller; simply connect it to a computer with a cable or power it with an AC-to-DC adapter or battery to get started. It can be upgraded with several integrations such as Wi-Fi and BT modules, motors, additional hardware.

- **Type**: Mainboard
- **Purpose**: Provides controlling the car remotely and the corresponding data-signal transmission, according to input commands.
- **Function**: Start, Stop, Steer, Gas, Break
- **Subordinates**
  - L298N Motor Driver
- **Interfaces**
  - **In:** Digital signal, Digital data, HC-05 Bluetooth Module
  - **Out:** Digital signal, Digital data, L298N Motor Driver

3.4 L298N Motor Driver Module

The L298N is a dual H-Bridge motor driver which allows speed and direction control of two DC motors at the same time. The module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A.[12]

- **Type**: Motor Driver Module
- **Purpose**: Interface between the mainboard and the DC motors
- **Function**
  - All wheels turning forward(Forward command)
  - All wheels return(Back command)
  - Right wheels spin forward(Turn Left command)
  - Left wheels spin forward(Turn right command)
- **Subordinates**
  - DC Motors(x4)
- **Interfaces**
  - **In:** Digital Signal, Digital Data, Arduino UNO
  - **Out:** DC current to the DC motors

3.5 ESP32 Camera Module

ESP32-CAM is a low cost development board with WiFi camera. It allows creating IP camera projects for video streaming with different resolutions. ESP32-CAM has build in PCB antenna.[8]

- **Type**: Camera Module(with AI Features)
- **Purpose**: For visual surveillance component. Starts web server fixed IP Address.
- **Function**
  - Start Stream
  - Stop Stream
  - Face Recognition
  - Face Enroll
  - Taking Photo

4 Software Details

In order for the Flutter application to work properly, the mobile phone SDK range version must be the following:

- minSdkVersion 19
- targetSdkVersion 29

The codes and their explainations are covered in the APPENDICES.

4.1 Languages, Frameworks, Plugins and IDE

- Dart(Programming Language for Flutter)
- C/C++(Programming Language for Arduino and Camera code)
- Flutter(Cross-platform framework)
- Android Studio(IDE): is used for writing Dart codes, debug and testing code.
- Arduino IDE: is used for writing C/C++ codes, debug and testing code.
- Genymotion(Cross-platform Android emulator): is used for virtual device and testing codes.
- webview_flutter(Plugin): is used for Camera Interface integrating to Flutter App.
  Requirements
  - minSdkVersion 19
- flutter_bluetooth_serial(Plugin): is used for Flutter app bluetooth connection with HC-05 Module and sending commands.
  Requirements
  - minSdkVersion 18

5 Hardware Details

5.1 Arduino UNO Mainboard

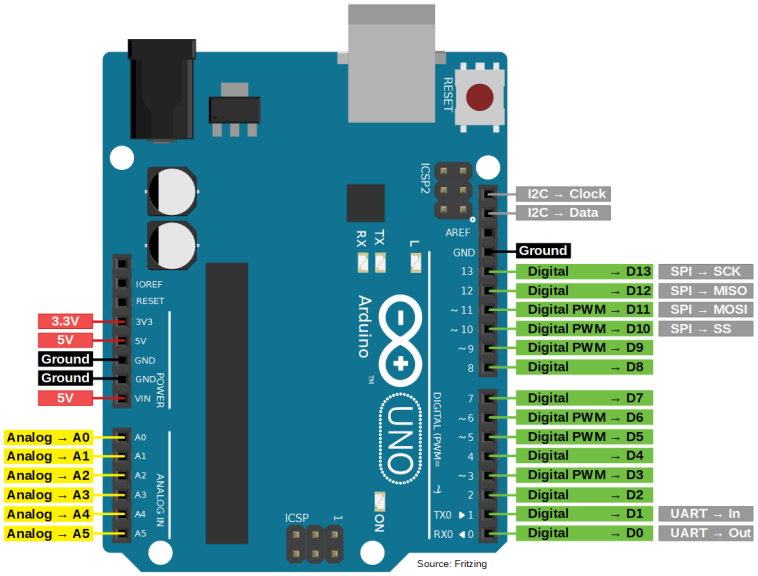

Fig. 2. Arduino UNO Pinout [9]

This mainboard is the main hardware component of the project, since all the other hardware components are connected to each other by it. These connections and the overall circuit diagram will be covered in the following sections. For now, only the pinout scheme of Arduino UNO is shown in Fig. 2.

5.2 HC-05 Bluetooth Module

The HC-05 Bluetooth Module provides the Bluetooth (2.0) communication between the Arduino UNO and other Bluetooth devices, as the name suggests. It works on a logic level of 3.3V. So, a 3.3V Regulator is used on the board.

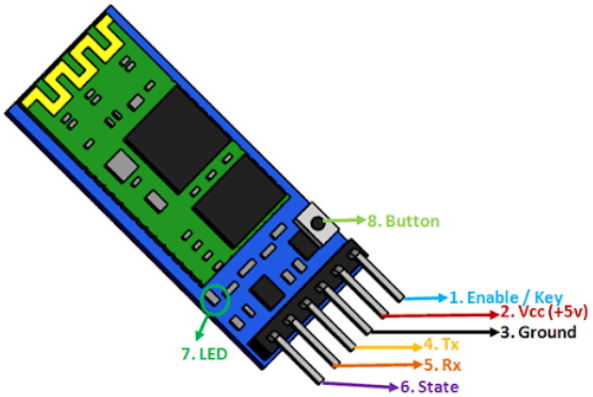

Fig. 3. HC-05 Module Pinout [10]

5.2.1 Pin Descriptions

- EN: Enable pin. When this pin is connected to 3.3V, the module is enabled. If this pin is connected to GND, the module is disabled.
- +5V: This is the supply pin for connecting +5V, but since the module has a 3.3V regulator, we can safely use 5V power supply.
- GND: Ground pin.
- TX: Transmitter pin of the communication.
- RX: Receive pin of the communication.
- STATE: This is a status indicator pin. This pin goes LOW when the module is not connected to any device. When the module is paired with any device, this pin goes HIGH.

5.3 L298N Motor Driver Module

This module provides the required DC current to the motors (4x). The L298N Motor Driver has a dual full driver IC, so it can control upto 2 motors simultaneously. Hence, while controlling our car, we run only two wheels at the same time. For example, to turn right, only the motors at the left-hand side run forward, and vice versa.
The L298N Motor Driver Module will directly take the input data from Arduino's 9-10-11-12. pins. The overall circuit diagram of the car, covering the Bluetooth module, motor driver and wheels is shown in the next figure.

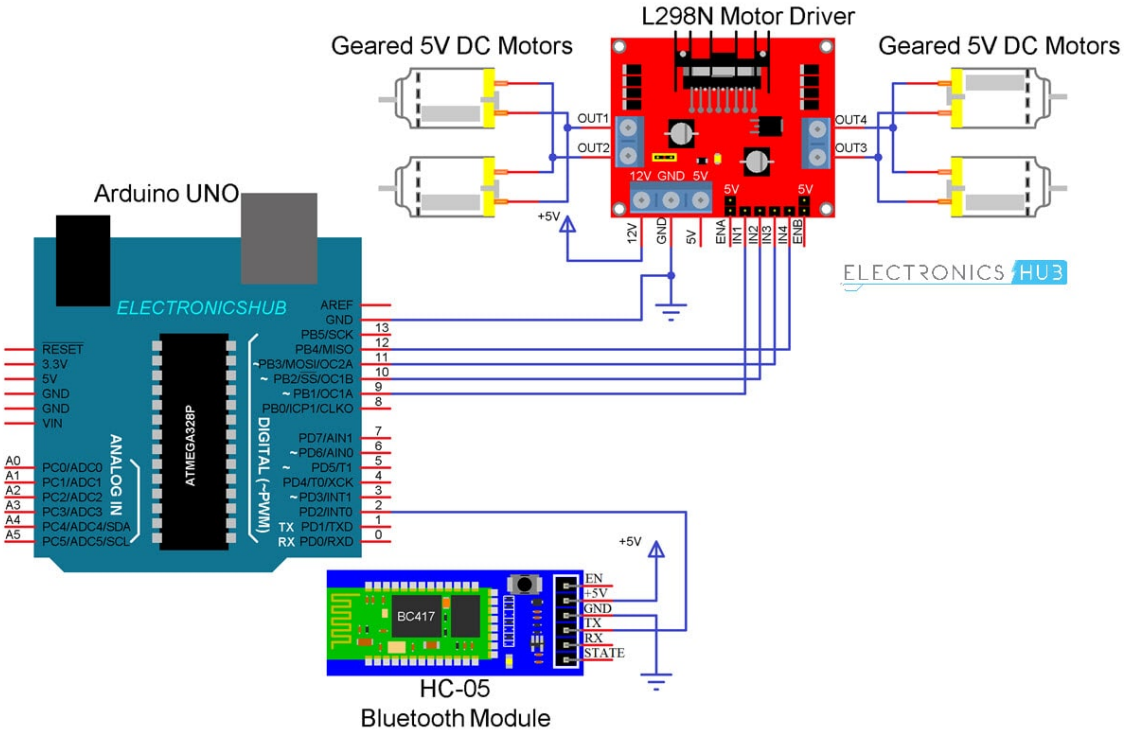

Fig. 4. Circuit Diagram of Bluetooth Controlled RC Car [11]

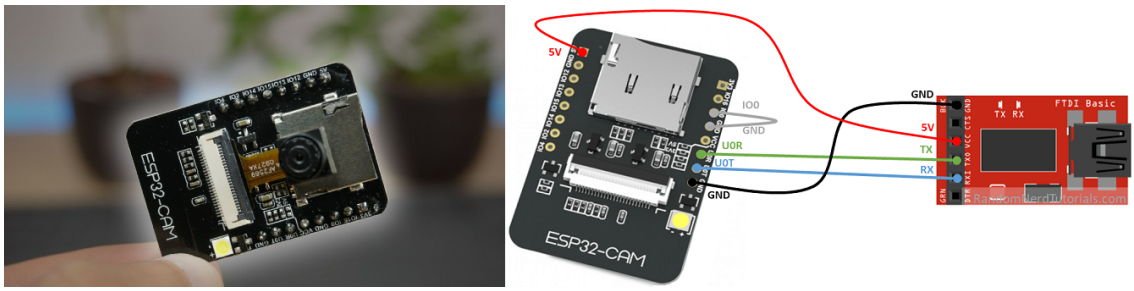

Fig. 5. ESP32-CAM and Programmer Diagram [7]

5.4 ESP32-CAM Module and Programmer

ESP32-CAM is a very useful component since it contains an embedded Wi-Fi endpoint alongside the camera itself. Also, it does not require any additional connections to the mainboard apart from the 5V and GND connections. After the power connections are done, the camera can start streaming realtime videos to a specific IP address, in the same network. To make the first-time connection and registration, the camera module must be loaded with the FTDI Programmer.
The steps can be summarized as:

1. Installing the ESP32 add-on to Arduino IDE
2. Loading the pre-written code (File/Examples/ESP32/Camera/CameraWebServer)
3. Uploading the code to camera with FTDI Programmer (the pinout is shown in Fig. 5)
4. Obtaining IP Address from Serial Input
5. Opening the IP Address from the URL of the browser

6 Conclusion

Via the Android application, forward, backward, right, left and turn on the camera commands, using bluetooth, a signal will be sent to the HC-05 Bluetooth module connected on the Arduino Uno single board computer. Then this module transmits the incoming signals to Arduino Uno. According to the content of the incoming command, signals are directed to the correct inputs of the L298N Engine Module. This engine module directs signals to 4 motors connected to it and the vehicle moves. For camera surveillance, data from the camera is sent to the android application using Wi-Fi technology.
Finally, we have experienced the Bluetooth and Wi-Fi communication technologies and their real life applications on a simple RC car prototype. As a result of this work, we made an inference that the data transfer rate is 3mbps and the maximum range of a Bluetooth device can be 10-20 meters. For wireless communication, the default baud rate is 38400 and other supported baud rates are 9600, 19200, 57600, 115200, 230400 and 460800. The criteria for choosing the baud rate are as follows.

- Data speed (Higher baud rate = faster transmission)
- Connection range (long distance $==more distortion/loss == can only support lower baud rate$)

7 Resources

[1] https://www.electronicshub.org/bluetooth-controlled-robot-using-arduino

[2] https://en.wikipedia.org/wiki/Bluetooth

[3] https://en.wikipedia.org/wiki/Wi-Fi

[4] https://store.arduino.cc/usa/arduino-uno-rev3

[5] freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020

[6] https://www.gme.cz/data/attachments/dsh.772-148.1.pdf

[7] randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide

[8] https://www.olimex.com/Products/IoT/ESP32/ESP32-CAM

[9] https://diyi0t.com/arduino-uno-tutorial

[10] http://blog.ikizoglu.com/2018/05/arduino-hc-05-bluetooth-kullanimi/

[11] https://www.electronicshub.org/bluetooth-controlled-robot-using-arduino/

[12] howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge

APPENDIX-A

Flutter Mobile App Codes

CODE	EXPLAINATION
``dart 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.
``dart 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.

APPENDIX-B

Arduino UNO Codes

CODE	EXPLAINATION
1 char t; 2 3 void setup() { 4 pinMode(9,OUTPUT); //left motors forward 5 pinMode(10,OUTPUT); //left motors reverse 6 pinMode(11,OUTPUT); //right motors forward 7 pinMode(12,OUTPUT); //right motors reverse 8 9 Serial.begin(9600); 10 11 }	In this part of code, the global variable named t is created and in setup function, it has been determined that the output will be given from the pins 9,10,11,12. Modulation rate(Symbol **rate)** declared 9600 Baud.

```
void loop() {
if(Serial.available()){
  t = Serial.read();
  Serial.println(t);
}

if(t == '1'){
  digitalWrite(9,HIGH);
  digitalWrite(10,LOW);
  digitalWrite(11,HIGH);
  digitalWrite(12,LOW);
}

else if(t == '2'){      /
  digitalWrite(9,LOW);
  digitalWrite(10,HIGH);
  digitalWrite(11,LOW);
  digitalWrite(12,HIGH);
}

else if(t == '3'){      /
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,HIGH);
  digitalWrite(12,LOW);
}

else if(t == '4'){      /
  digitalWrite(9,HIGH);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
  digitalWrite(12,LOW);
}

else if(t == '5'){      /
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
  digitalWrite(12,LOW);
}
delay(100);
}
```

In this part of code, while the Arduino has power , this function again and again will work, in the previous stage of code the global variable named t was created.

The signals coming from the TX of the Bluetooth to the RX pin of the arduino are assigned to the t value.

According to the content of this value, the volt values given from pins 9,10,11 and 12 vary. These pins determine the wheel's situation by the L298N Motor Driver.

- If t is equal to 1, then all wheels turn forward
- If t is equal to 2, then all wheels turn back
- If t is equal to 3, then right
- wheels turn so the car turns left.
- If t is equal to 4, then left wheels turn then the car turns right.
- If t is equal to 5, then all wheels stop.