# ДОДАТОК А

## Текст програми міні-робота

```
#include <ServoSmooth.h>
#include <GyverMotor.h>
#include <NeoSWSerial.h>

//------------------------
//---Turn left/right servo
ServoSmooth turnServo;
uint8_t turnServo_pin = 12;

//---Robo Servo
uint8_t roboServo_pin1 = 11;
uint8_t roboServo_pin2 = 10;
uint8_t roboServo_pin3 = 6;
uint8_t roboServo_pin4 = 5;
ServoSmooth roboServo1; // distance servo
ServoSmooth roboServo2; // rotate servo
ServoSmooth roboServo3; // arm servo
ServoSmooth roboServo4; // height servo
//---fist position for Robo servo
uint8_t roboServo1_pos = 40;
uint8_t roboServo2_pos = 90;
uint8_t roboServo3_pos = 60;
uint8_t roboServo4_pos = 150;
```

```
//------------------------
//---Bluetooth conection
NeoSWSerial bt_hc06(4,2);

//------------------------
//---Connect Motor
uint8_t motor_pin1 = 7;
uint8_t motor_pin2 = 8;
uint8_t motor_pwmPin = 3;
GMotor motor(DRIVER3WIRE, motor_pin1, motor_pin2, motor_pwmPin, HIGH);

//------------------------
//---some value
String readData; //input data from serial
int degree = 0;
float distance = 0.00;
int temp;

//------------------------
//---init function
void setup() {
//---init serial and bluetooth transmit
 Serial.begin(9600);
 Serial.setTimeout(10);
 bt_hc06.begin(9600);
 bt_hc06.setTimeout(10);
```

```
//---init turnServo
turnServo.attach(turnServo_pin);
turnServo.write(90); // setup position

//---init roboServo
initRoboServo();

//---init motor
motor.setMode(FORWARD);
motor.setDirection(REVERSE);
}
void initRoboServo(){
   roboServo1.attach(roboServo_pin1); //distance
   roboServo1.write(roboServo1_pos);
   roboServo2.attach(roboServo_pin2); //rotate
   roboServo2.write(roboServo2_pos);
   roboServo3.attach(roboServo_pin3); //arm
   roboServo3.write(roboServo3_pos);
   roboServo4.attach(roboServo_pin4); //height
   roboServo4.write(roboServo4_pos);
}
//------------------------
//---infinity loop function
void loop() {
  //---check bluetooth
  if(bt_hc06.available()>0){
    readData = bt_hc06.readString();
    Serial.println(readData);
```

```
    if(readData.substring(0,2) == "mv")
      move_chassis(readData);
    else if(readData.substring(0,2) == "mr")
      move_robot(readData);
  }
}
//------------------------
//---move vechile function
void move_chassis(String data){
 degree = data.substring(3,6).toInt();
 distance = atof(data.substring(6,9).c_str());
 //---check rotate
 if(distance >= 0.4){
  if(degree >35 && degree <135){
    turnServo.write(70);
  } else
  if(degree >225 && degree <315){
    turnServo.write(100
    );
  }
  else turnServo.write(85);
 }
 else turnServo.write(85);
 //---check forward of reverse
 if(distance >= 0.4){
    if((degree >=270 && degree <=360)||(degree >=0 && degree <=90)){
      motor.setMode(FORWARD);
```

```
      motor.setSpeed(100);
    }
    else {
      motor.setMode(BACKWARD);
      motor.setSpeed(100);
    }
 }
 else {
  motor.setSpeed(0);
 }
}


//------------------------
//---move manipulator function
void move_robot(String data){
  if(data == "mrh+") {
    if(roboServo4_pos <180) {
      roboServo4_pos+=10;
      roboServo4.write(roboServo4_pos);
    }
  }
  else
  if(data == "mrh-") {
    if(roboServo4_pos >150) {
      roboServo4_pos-=10;
      roboServo4.write(roboServo4_pos);
    }
```

```
}
else
if(data == "mrr+") {
  if(roboServo2_pos <180) {
    roboServo2_pos+=10;
    roboServo2.write(roboServo2_pos);
  }
}
else
if(data == "mrr-") {
  if(roboServo2_pos >0) {
    roboServo2_pos-=10;
    roboServo2.write(roboServo2_pos);
  }
}
else
if(data == "mra") {
  if(roboServo3_pos == 0) {
    roboServo3_pos=60;
    roboServo3.write(roboServo3_pos);
  }
  else {
    roboServo3_pos=0;
    roboServo3.write(roboServo3_pos);
  }
}
else
if(data == "mrp+") {
```

```
    if(roboServo1_pos <160) {
      roboServo1_pos+=10;
      roboServo1.write(roboServo1_pos);
    }
  }
  else
  if(data == "mrp-") {
    if(roboServo1_pos >40) {
      roboServo1_pos-=10;
      roboServo1.write(roboServo1_pos);
    }
  }
}
```

# ДОДАТОК Б

## Текст програми мобільного додатку

Файл main.dart:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'ConectBluetoothDevice.dart';
import 'MainPage.dart';

void main() {
  runApp(RemoveControlForRobots());
}
class RemoveControlForRobots extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (context) => MyBlue(),
      child: MaterialApp(
          debugShowCheckedModeBanner: false,
          theme: ThemeData(
              primaryColor: Colors.orange,
              primaryIconTheme: Theme.of(context)
                  .primaryIconTheme
                  .copyWith(color: Colors.white),
              primaryTextTheme: Theme.of(context)
                  .primaryTextTheme
                  .apply(bodyColor: Colors.white)),
```

```
        home: MainPage()),
    );
  }
}
```

Файл MainPage.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:control_pad/control_pad.dart';
import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';
import 'package:provider/provider.dart';
import 'package:remove_control_for_robots/SettingsPage.dart';
import 'package:remove_control_for_robots/TerminalPage.dart';
import 'package:remove_control_for_robots/SelectBondedDevicePage.dart';
import 'ConectBluetoothDevice.dart';
class MainPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ControlPage();
  }
}
class ControlPage extends StatelessWidget {
  String settingsImage = "assets/svg/settings.svg";
  String terminalImage = "assets/svg/code-terminal.svg";
  BluetoothDevice serverr;
  BluetoothConnection connectionn;
  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      appBar: AppBar(
        leading: IconButton(
          icon: SvgPicture.asset(
            terminalImage,
            color: Colors.white,
          ),
          onPressed: (context.watch<MyBlue>().getisConnected)
              ? () {
                  serverr = context.read<MyBlue>().getSelectedDevice;
                  connectionn = context.read<MyBlue>().getConnection;
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) => TerminalPage(
                          server: this.serverr,
                          connection: this.connectionn,
                        )),);
                } : () {
                  showDialog(
                    context: context,
                    builder: (BuildContext context) {
                      return AlertDialog(
                        title: Text("Error!!!"),
                        content: Text(
                            "Please, connect to bluetooth device, before open terminal
page."),
                        actions: [
                          FlatButton(
```

```
                    onPressed: () {
                      Navigator.of(context).pop();
                    },
                    child: Text('Ok, close!'))
                ],);
              });
          },),
  title: ViewTopName(),
  actions: [
    ViewBlueSetting(),
    IconButton(
      icon: SvgPicture.asset(
        settingsImage,
        color: Colors.white,
      ),
      onPressed: () {
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => SettingsPage()),);},)],),
body: OrientationBuilder(builder: (context, orientation) {
  if (orientation == Orientation.portrait) {
    return Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [ViewImage(), ViewSettingBar(), ViewJoystick()]);
  } else {
    return Row(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          ViewSettingBarLandscapeMode(),
```

```
                ViewImageLandscapeMode(),
                ViewJoystickLandscapeMode()]);}}));}}

class ViewBlueSetting extends StatefulWidget {
  _ViewBlueSettingState createState() => _ViewBlueSettingState();
}

class _ViewBlueSettingState extends State<ViewBlueSetting> {
  BluetoothState _bluetoothState = BluetoothState.UNKNOWN;
  String blueImage = "assets/svg/bluetooth.svg";

  @override
  void initState() {
    FlutterBluetoothSerial.instance.state.then((state) {
      setState(() {
        _bluetoothState = state;
      });});
    FlutterBluetoothSerial.instance
        .onStateChanged()
        .listen((BluetoothState state) {
      setState(() {
        _bluetoothState = state;
      });});
    super.initState();}

  @override
  Widget build(BuildContext context) {
    return IconButton(
      icon: SvgPicture.asset(
```

```
        blueImage,
        color: Colors.white,
      ),
      onPressed: () {
        (_bluetoothState.isEnabled)
            ? Navigator.push(
                context,
                MaterialPageRoute(
                    builder: (context) => SelectBondedDevicePage()),
              ) : showDialog(
                context: context,
                builder: (BuildContext context) {
                  return AlertDialog(
                    title: Text("Error!!!"),
                    content: Text("Please, turn on bluetooth."),
                    actions: [
                      FlatButton(
                          onPressed: () {
                            Navigator.of(context).pop();
                          },
                          child: Text('Ok, close!'))
                    ],);});},);}}

class ViewTopName extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return (context.watch<MyBlue>().getisConnected)
        ? Text("Connected to " + context.watch<MyBlue>().getDeviceName)
        : Text("");}}
```

```
class ViewImage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      height: 230,
      margin: EdgeInsets.symmetric(horizontal: 30, vertical: 15),
      decoration: BoxDecoration(
          image: DecorationImage(
              image: Image.asset("assets/images/robot.jpg").image,
              fit: BoxFit.cover),
          borderRadius: BorderRadius.circular(15),
          boxShadow: [BoxShadow(color: Colors.grey[300], blurRadius: 10.0)]),
    );}}
class ViewImageLandscapeMode extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      height: 230,
      width: 230,
      margin: EdgeInsets.symmetric(horizontal: 30, vertical: 15),
      decoration: BoxDecoration(
          image: DecorationImage(
              image: Image.asset("assets/images/robot.jpg").image,
              fit: BoxFit.cover),
          borderRadius: BorderRadius.circular(15),
          boxShadow: [BoxShadow(color: Colors.grey[300], blurRadius: 10.0)]),
    );}}
class ViewSettingBar extends StatelessWidget {
  @override
```

```
Widget build(BuildContext context) {
  return Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: [
      if(context.watch<MyBlue>().getCountOfButton == "Five"){
        AddButton(1),
      AddButton(2),
      AddButton(3),
      AddButton(4),
      AddButton(5),
      } else if(context.watch<MyBlue>().getCountOfButton == "Four"){
        AddButton(1),
      AddButton(2),
      AddButton(3),
      AddButton(4),
      }
      else if(context.watch<MyBlue>().getCountOfButton == "Three"){
        AddButton(1),
      AddButton(2),
      AddButton(3),
      }
      else if(context.watch<MyBlue>().getCountOfButton == "Two"){
        AddButton(1),
      AddButton(2),
      }
      else if(context.watch<MyBlue>().getCountOfButton == "One"){
        AddButton(1),
      }
      else if(context.watch<MyBlue>().getCountOfButton == "Null"){
```

```
        SizedBox(
          height: 80,
        )}],);}}

class ViewSettingBarLandscapeMode extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        if(context.watch<MyBlue>().getCountOfButton == "Five"){
          AddButton(1),
        AddButton(2),
        AddButton(3),
        AddButton(4),
        AddButton(5),
        } else if(context.watch<MyBlue>().getCountOfButton == "Four"){
          AddButton(1),
        AddButton(2),
        AddButton(3),
        AddButton(4),
        }
        else if(context.watch<MyBlue>().getCountOfButton == "Three"){
          AddButton(1),
        AddButton(2),
        AddButton(3),
        }
        else if(context.watch<MyBlue>().getCountOfButton == "Two"){
          AddButton(1),
```

```
      AddButton(2),
      }
      else if(context.watch<MyBlue>().getCountOfButton == "One"){
       AddButton(1),
      }
      else if(context.watch<MyBlue>().getCountOfButton == "Null"){
       SizedBox(
        width: 80,
       )
      }],);}}

class AddButton extends StatefulWidget {
 final int count;
 AddButton(this.count);
 @override
 _AddButtonState createState() => _AddButtonState();}
class _AddButtonState extends State<AddButton> {
 TextEditingController _controller;
 TextEditingController _controllerm;
 void initState() {
  super.initState();
  _controller = TextEditingController();
  _controllerm = TextEditingController();  }
 void dispose() {
  _controller.dispose();
  _controllerm.dispose();
  super.dispose();  }
 @override
 Widget build(BuildContext context) {
```

```
    return Container(
      width: 70,
      height: 50,
      child: FlatButton(
        onPressed: (context.watch<MyBlue>().getisConnected) ? () {
              if (context.read<MyBlue>().buttonMessage(widget.count) != "")
                context.read<MyBlue>().sendMessage(
context.read<MyBlue>().buttonMessage(widget.count));
              else
                showDialog(
                    context: context,
                    builder: (BuildContext context) {
                      return AlertDialog(
                        title: Text("Please edit button №${widget.count}"),
                        content: Text("Add commant to transmit."),
                        actions: [
                          FlatButton(
                              onPressed: () {
                                Navigator.of(context).pop();
                              },
                              child: Text('Ok, close!'))
                        ],);});} : () {
              showDialog(
                  context: context,
                  builder: (BuildContext context) {
                    return AlertDialog(
                      title:
                          Text("Error!!! Button №${widget.count} not work!"),
                      content: Text(
```

```
                    "Please, connect to bluetooth device, before tap on this button again."),
                  actions: [
                   FlatButton(
                      onPressed: () {
                        Navigator.of(context).pop();
                      },
                      child: Text('Ok, close!'))
                  ],);}); },
        onLongPress: () {
          showModalBottomSheet<dynamic>(
              context: context,
              isScrollControlled: true,
              backgroundColor: Colors.transparent,
              builder: (BuildContext context) {
                return Container(
                  height: (MediaQuery.of(context).viewInsets.bottom == 0)
                      ? 265
                      : MediaQuery.of(context).size.height -
                  MediaQuery.of(context).viewInsets.bottom + 40,
                  decoration: BoxDecoration(
                      color: Colors.white,
                      borderRadius: BorderRadius.only(
                          topLeft: const Radius.circular(25.0),
                          topRight: const Radius.circular(25.0))),
                  child: Column(
                    children: [
                      Container(
                        padding: EdgeInsets.only(top: 10.0, bottom: 10.0),
```

```
  child: Text('Button №${widget.count} name:',
      style:
          TextStyle(color: Colors.orange, fontSize: 20)),                    ),
Container(
  width: MediaQuery.of(context).size.width - 50.0,
  child: TextField(
    decoration: InputDecoration(
      hintText: "Enter button №${widget.count} name",
      border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(25.0),),),
    controller: _controller,
  ), ),
Container(
  padding: EdgeInsets.only(top: 10.0, bottom: 10.0),
  child: Text('Button №${widget.count} message:',
      style:
          TextStyle(color: Colors.orange, fontSize: 20)),
),
Container(
  width: MediaQuery.of(context).size.width.toInt() - 50.0,
  child: TextField(
    decoration: InputDecoration(
      hintText: "Enter button №${widget.count} message",
      border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(25.0),),  ),
    controller: _controllerm,
  ),  ),
FlatButton.icon(
    icon: Icon(Icons.check),
```

```
                label: Text('Apply changes'),
                textColor: Colors.white,
                color: Colors.orange,
                onPressed: () {
                  if (_controller.text != '') {
                    context.read<MyBlue>().setButtonname(
                        _controller.text, widget.count);
                    _controller.clear();
                  }
                  if (_controllerm.text != '') {
                    context.read<MyBlue>().setButtonMessage(
                        _controllerm.text, widget.count);
                    _controllerm.clear();
                  }
                  Navigator.of(context).pop();
                })],),);});},
        color: Colors.blue,
        padding: EdgeInsets.all(2),
        shape: OutlineInputBorder(),
        child: Text(
          (context.watch<MyBlue>().buttonName(widget.count) == '')
              ? 'But ${widget.count}'
              : context.watch<MyBlue>().buttonName(widget.count),
          style: TextStyle(
            fontSize: 18,  ),
          textAlign: TextAlign.center,   ),
        textColor: Colors.white, ),
    );}}
class ViewJoystick extends StatelessWidget {
```

```
  @override
  Widget build(BuildContext context) {
    return (context.watch<MyBlue>().getSelectJoystick)
        ? ControlJoystick()
        : ControlArrows();
  }}
class ViewJoystickLandscapeMode extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return (context.watch<MyBlue>().getSelectJoystick)
        ? ControlJoystickLandscapeMode()
        : ControlArrowsLandscapeMode();
  }}
class ControlJoystick extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    JoystickDirectionCallback onDirectionChanged(
        double degrees, double distance) {
      if (context.read<MyBlue>().getisConnected) {
        String data = "mv ${degrees.toInt()}";
        if (data.length == 6)
          data += "${distance.toStringAsFixed(2)}";
        else if (data.length == 5)
          data += " ${distance.toStringAsFixed(2)}";
        else if (data.length == 4) data += "  ${distance.toStringAsFixed(2)}";
        print(data);
        if (degrees == 0.00 && distance == 0.00) {
          context.read<MyBlue>().sendMessage(data);        }
        context.read<MyBlue>().sendMessage(data);
```

```
    } else {
      showDialog(
          context: context,
          builder: (BuildContext context) {
            return AlertDialog(
              title: Text("Error!!! Joystick not work!"),
              content: Text(
                  "Please, connect to bluetooth device, before tap on joystick again."),
              actions: [
                FlatButton(
                    onPressed: () {
                      Navigator.of(context).pop();
                    },
                    child: Text('Ok, close!'))],
          );});}}  return FittedBox(
      child: Row(crossAxisAlignment: CrossAxisAlignment.start, children: [
    SizedBox(
      width: 90,
),
    JoystickView(
      size: 200,
      onDirectionChanged: onDirectionChanged,
      interval: Duration(milliseconds: 150),
      innerCircleColor: Colors.green,
    ),
    Container(
        width: 90,
        alignment: Alignment.topRight,
        child: (context.watch<MyBlue>().isEnableAdditionControl)
```

```
            ? IconButton(
                iconSize: 40,
                icon: SvgPicture.asset(
                  context.watch<MyBlue>().imageRobo,
                  color: Colors.blueAccent,
                ),
                onPressed: (context.watch<MyBlue>().getisConnected) ? () {
context.read<MyBlue>().setSelectJoystick();}
                    : () {showDialog(
                          context: context,
                          builder: (BuildContext context) {
                            return AlertDialog(
                              title: Text(
                                  "Error!!! Change control button not work!"),
                              content: Text(
                                  "Please, connect to bluetooth device, before tap on this button
again."),
                              actions: [
                                FlatButton(                    onPressed: () {
Navigator.of(context).pop();                    },
child: Text('Ok, close!'))],);});}): Text('')),]));}}
class ControlArrows extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return FittedBox(
      child: Row(crossAxisAlignment: CrossAxisAlignment.start, children: [
     Row(children: [
      Container(
        padding: EdgeInsets.symmetric(horizontal: 12),
        child: Column(
```

```
children: [
  Ink(
    decoration: BoxDecoration(
        color: Color(0xFFF08B2D),
        border: Border.all(width: 2, color: Colors.black),
        borderRadius: BorderRadius.circular(5)),
    child: IconButton(
        icon: Icon(
          Icons.arrow_circle_up,
          color: Colors.white,
        ),
        iconSize: 50,
        onPressed: () {
          context.read<MyBlue>().sendMessage("mrh+");
        }),
  ),
  SizedBox(
    height: 20,
    width: 50,
  ),
  Ink(
      decoration: BoxDecoration(
          color: Color(0xFFF08B2D),
          border: Border.all(width: 2, color: Colors.black),
          borderRadius: BorderRadius.circular(5)),
      child: IconButton(
          icon: Icon(
            Icons.arrow_circle_down,
            color: Colors.white,
```

```
            ),
            //color: Colors.white,
            iconSize: 50,
            onPressed: () {
              context.read<MyBlue>().sendMessage("mrh-");
            }))],),),
Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Ink(
        decoration: BoxDecoration(
            color: Color(0xFF177E03),
            border: Border.all(width: 2, color: Colors.black),
            borderRadius: BorderRadius.circular(5)),
        child: IconButton(
            icon: Icon(
              Icons.arrow_back,
              color: Colors.white,
            ),
            iconSize: 50,
            onPressed: () {
              context.read<MyBlue>().sendMessage("mrl");
            })), ],),
Column(
  children: [
    Ink(
        decoration: BoxDecoration(
            color: Color(0xFF177E03),
            border: Border.all(width: 2, color: Colors.black),
```

```
        borderRadius: BorderRadius.circular(5)),
    child: IconButton(
        icon: Icon(
          Icons.arrow_upward,
          color: Colors.white,
        ),
        iconSize: 50,
        onPressed: () {
          context.read<MyBlue>().sendMessage("mrf");
        })),
Container(
  padding: EdgeInsets.all(6),
  child: Ink(
      decoration: BoxDecoration(
          color: Colors.blue,
          border: Border.all(width: 2, color: Colors.black),
          borderRadius: BorderRadius.circular(5)),
      child: IconButton(
          icon: Icon(
            Icons.adjust,
            color: Colors.white,
          ),
          iconSize: 44,
          onPressed: () {
            context.read<MyBlue>().sendMessage("mra");
          }),),
Ink(
    decoration: BoxDecoration(
        color: Color(0xFF177E03),
```

```
            border: Border.all(width: 2, color: Colors.black),
            borderRadius: BorderRadius.circular(5)),
          child: IconButton(
            icon: Icon(
              Icons.arrow_downward,
              color: Colors.white,
            ),
            iconSize: 50,
            onPressed: () {
              context.read<MyBlue>().sendMessage("mrb");
            }))
      ],
    ),
    Column(
      children: [
        Ink(
          decoration: BoxDecoration(
            color: Color(0xFF177E03),
            border: Border.all(width: 2, color: Colors.black),
            borderRadius: BorderRadius.circular(5)),
          child: IconButton(
            icon: Icon(
              Icons.arrow_forward,
              color: Colors.white,
            ),
            iconSize: 50,
            onPressed: () {
              context.read<MyBlue>().sendMessage("mrr");
            })),
```

```
          ],
        ),
      ]),
      Container(
        width: 90,
        alignment: Alignment.topRight,
        child: IconButton(
            iconSize: 40,
            icon: SvgPicture.asset(
              context.watch<MyBlue>().imageVechile,
              color: Colors.blueAccent,
            ),
            onPressed: () {
              context.read<MyBlue>().setSelectJoystick();
            }),
      ),
    ]));
  }
}

class ControlJoystickLandscapeMode extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    JoystickDirectionCallback onDirectionChanged(
        double degrees, double distance) {
      String data = "mv ${degrees.toInt()}";
      if (data.length == 6)
        data += "${distance.toStringAsFixed(2)}";
      else if (data.length == 5)
```

```
    data += " ${distance.toStringAsFixed(2)}";
  else if (data.length == 4) data += "  ${distance.toStringAsFixed(2)}";
  print(data);
  if (degrees == 0.00 && distance == 0.00) {
    context.read<MyBlue>().sendMessage(data);
  }
  context.read<MyBlue>().sendMessage(data);
}

return FittedBox(
    child: Column(crossAxisAlignment: CrossAxisAlignment.end, children: [
  Container(
      height: 80,
      alignment: Alignment.topRight,
      child: (context.watch<MyBlue>().isEnableAdditionControl)
          ? IconButton(
              iconSize: 40,
              icon: SvgPicture.asset(
                context.watch<MyBlue>().imageRobo,
                color: Colors.blueAccent,
              ),
              onPressed: () {
                context.read<MyBlue>().setSelectJoystick();
              })
          : Text('')),
  JoystickView(
    size: 200,
    onDirectionChanged: onDirectionChanged,
    interval: Duration(milliseconds: 150),
```

```
        innerCircleColor: Colors.green,
      ),
    ]));
  }
}

class ControlArrowsLandscapeMode extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return FittedBox(
      child: Column(crossAxisAlignment: CrossAxisAlignment.end, children: [
      Container(
        height: 80,
        alignment: Alignment.topRight,
        child: IconButton(
            iconSize: 40,
            icon: SvgPicture.asset(
              context.watch<MyBlue>().imageVechile,
              color: Colors.blueAccent,
            ),
            onPressed: () {
              context.read<MyBlue>().setSelectJoystick();
            }),
      ),
      Row(children: [
        Container(
          padding: EdgeInsets.symmetric(horizontal: 12),
          child: Column(
            children: [
```

```
Ink(
  decoration: BoxDecoration(
      color: Color(0xFFF08B2D),
      border: Border.all(width: 2, color: Colors.black),
      borderRadius: BorderRadius.circular(5)),
  child: IconButton(
      icon: Icon(
        Icons.arrow_circle_up,
        color: Colors.white,
      ),
      iconSize: 50,
      onPressed: () {
        context.read<MyBlue>().sendMessage("mrh+");
      }),
),
SizedBox(
  height: 20,
  width: 50,
),
Ink(
    decoration: BoxDecoration(
        color: Color(0xFFF08B2D),
        border: Border.all(width: 2, color: Colors.black),
        borderRadius: BorderRadius.circular(5)),
    child: IconButton(
        icon: Icon(
          Icons.arrow_circle_down,
          color: Colors.white,
        ),
```

```
            iconSize: 50,
            onPressed: () {
              context.read<MyBlue>().sendMessage("mrh-");
            }))          ],         ),       ),
Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Ink(
        decoration: BoxDecoration(
            color: Color(0xFF177E03),
            border: Border.all(width: 2, color: Colors.black),
            borderRadius: BorderRadius.circular(5)),
        child: IconButton(
            icon: Icon(
              Icons.arrow_back,
              color: Colors.white,
            ),
            iconSize: 50,
            onPressed: () {
              context.read<MyBlue>().sendMessage("mrl");
            })),   ],     ),
Column(
  children: [
    Ink(
        decoration: BoxDecoration(
            color: Color(0xFF177E03),
            border: Border.all(width: 2, color: Colors.black),
            borderRadius: BorderRadius.circular(5)),
        child: IconButton(
```

```
        icon: Icon(
          Icons.arrow_upward,
          color: Colors.white,
        ),
        iconSize: 50,
        onPressed: () {
          context.read<MyBlue>().sendMessage("mrf");
        })),
Container(
  padding: EdgeInsets.all(6),
  child: Ink(
      decoration: BoxDecoration(
          color: Colors.blue,
          border: Border.all(width: 2, color: Colors.black),
          borderRadius: BorderRadius.circular(5)),
      child: IconButton(
          icon: Icon(
            Icons.adjust,
            color: Colors.white,
          ),
          iconSize: 44,
          onPressed: () {
            context.read<MyBlue>().sendMessage("mra");
          })),           ),
Ink(
    decoration: BoxDecoration(
        color: Color(0xFF177E03),
        border: Border.all(width: 2, color: Colors.black),
        borderRadius: BorderRadius.circular(5)),
```

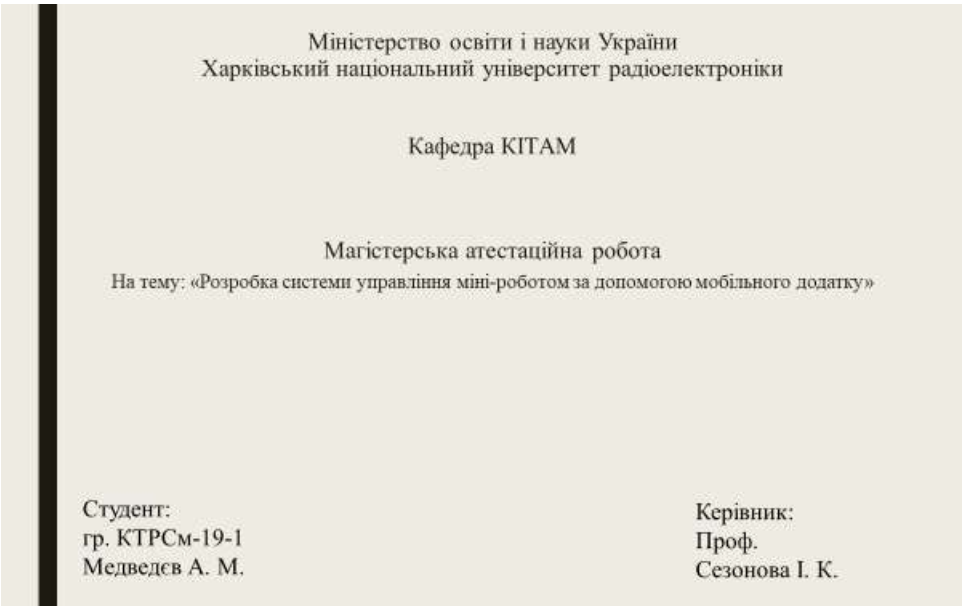
```
        child: IconButton(
          icon: Icon(
            Icons.arrow_downward,
            color: Colors.white,
          ),
          iconSize: 50,
          onPressed: () {
            context.read<MyBlue>().sendMessage("mrb");
          }))        ],        ),
  Column(
    children: [
      Ink(
        decoration: BoxDecoration(
          color: Color(0xFF177E03),
          border: Border.all(width: 2, color: Colors.black),
          borderRadius: BorderRadius.circular(5)),
        child: IconButton(
          icon: Icon(
            Icons.arrow_forward,
            color: Colors.white,
          ),
          iconSize: 50,
          onPressed: () {
            context.read<MyBlue>().sendMessage("mrr");
          })),       ],    ),    ]), ]));}}
```

# ДОДАТОК В

## Демонстраційний матеріал

### Слайд 1

### ТИТУЛЬНИЙ ЛИСТ РОБОТИ



Слайд 2

МЕТА ТА ЗАДАЧІ АТЕСТАЦІЙНОЇ РОБОТИ

## МЕТА ТА ЗАДАЧІ АТЕСТАЦІЙНОЇ РОБОТИ

Метою магістерської роботи є розробка системи дистанційного управління міні-роботом за допомогою мобільного додатку.

Для вирішення актуального завдання необхідно виконати наступні задачі:

- аналіз вхідних даних та літератури з теми атестаційної роботи;
- вибір електронних компонентів системи керування роботом;
- розробка макетних та принципових схем підключення компонентів;
- виконати розрахунок блоку живлення;
- розробити систему команд для керування роботом;
- розробити мобільний додаток;
- розробити інструкцію користувача мобільного додатку.

2

Слайд 3

МОБІЛЬНІ РОБОТИ ЯК ОБ’ЄКТИ КЕРУВАННЯ

## МОБІЛЬНІ РОБОТИ ЯК ОБ’ЄКТИ КЕРУВАННЯ

Мобільний робот – це робот, який може пересуватися і переміщатися в просторі.

Однією з найпоширеніших та дешевих конструкцій вважають колісні роботи.

На рисунку зображено приклад колісного роботу.

3

Слайд 4

ВАРІАНТИ ДИСТАНЦІЙНОГО КЕРУВАННЯ


Слайд 5

ВИБІР ЕЛЕКТРОННИХ КОМПОНЕНТІВ

Слайд 6

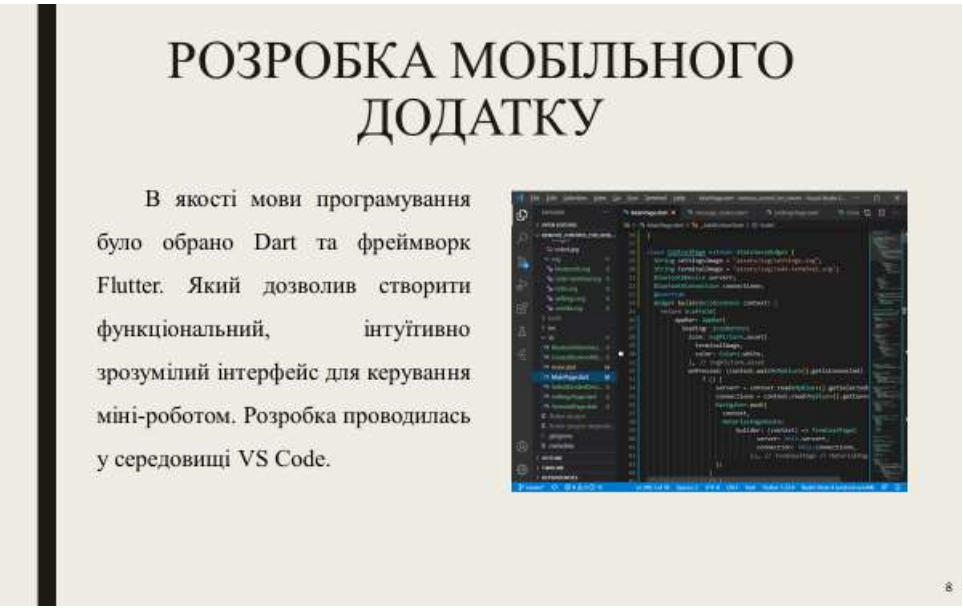
РОЗРОБКА МАКЕТНИХ ТА ПРИНЦИПОВИХ СХЕМ


Слайд 7

РЕЗУЛЬТАТ РОЗРОБКИ ТА ЗБІРКИ ПРОТОТИПУ МІНІ-РОБОТУ

Слайд 8

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

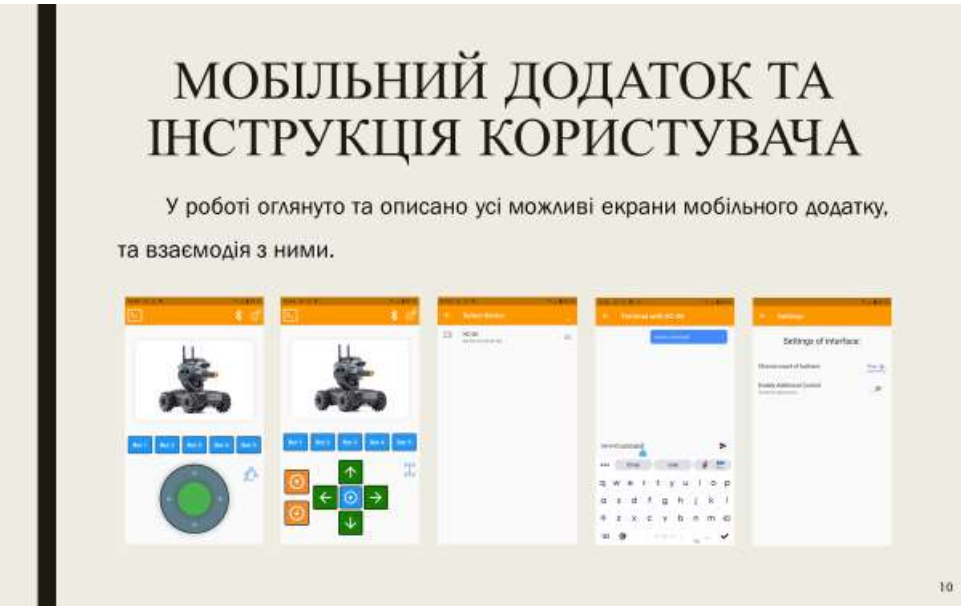



Слайд 9

МОБІЛЬНИЙ ДОДАТОК ТА ІНСТРУКЦІЯ КОРИСТУВАЧА

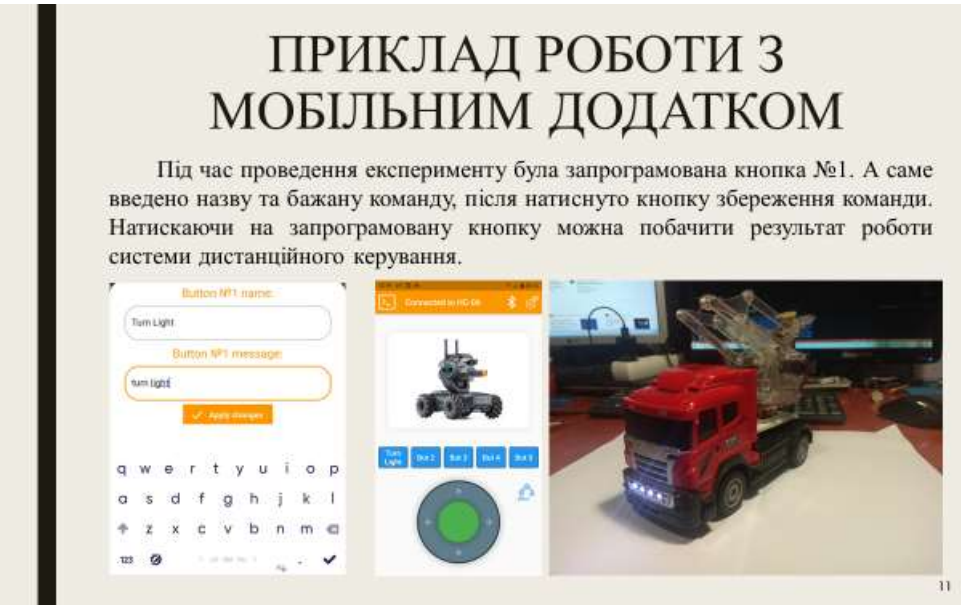
Слайд 10

МОБІЛЬНИЙ ДОДАТОК ТА ІНСТРУКЦІЯ КОРИСТУВАЧА




Слайд 11

ПРИКЛАД РОБОТИ З МОБІЛЬНИМ ДОДАТКОМ



Слайд 12

ВИСНОВКИ

## ВИСНОВКИ

В атестаційній роботі розглянуто задачі розробки систем дистанційного керування мобільними роботами.

У першому розділі зроблено огляд сучасного стану проблеми дистанційного керування мобільними роботами, а саме, організацію систем, методи передачі даних, структурні елементи мобільних роботів. Для розв'язання проблеми системи дистанційного керування мобільними роботами необхідно розробити апаратну та програмну частини системи дистанційного керування міні-роботом.

За результатами огляду сучасних рішень з розробки систем дистанційного керування обрано технологію IoT. В якості протоколу передачі даних було обрано технологію Bluetooth. За типом організації системи зв'язку було обрано метод point-to-point.

Розроблено схеми підключення електронних компонентів системи керування міні-робота, розраховано блок живлення. Розроблено схеми алгоритмів роботи програми міні-робота. Розроблено програмний засіб для системи дистанційного керування, а саме, мобільний додаток, який відправляє команди мережею Bluetooth на приймач міні-робота, та програму керування для контролеру, яка приймає дані від Bluetooth модулю HC-06, та обробляє їх.

Розроблено інструкцію користувача, в коді якої:

- визначено системні вимоги розробленого мобільного додатку;
- визначено шляхи інсталяції, оновлення та видалення додатку з OEM пристрою користувача;
- детального оглянуто можливості додатку, з описом усіх доступних методів взаємодії;
- наведено приклади роботи з мобільним додатком.

12

Додаток Г

Відомість атестаційної роботи

| № | Позначення | Найменування | Додаткові відомості |
|---|---|---|---|
| | | Текстові документи | |
| | | | |
| 1 | ГЮИК.505330.008 ПЗ | Пояснювальна записка | 120 с., ф. А4 |
| | | | |
| | | Додаткові матеріали | |
| | | | |
| 2 | | Демонстраційний матеріал | у форматі |
| | | | *.ppt |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ГЮИК.505330.008 ВД | |
| Зм. | Арк. | № документа | Підп. | Дата | | |
| Розробив | | Медведєв А.М. | | | Розробка системи управління міні-роботом за допомогою мобільного додатку<br><br>Відомість атестаційної роботи | Літера: <br> Аркуш: <br> Аркушів: 1 |
| Перевір. | | Сезонова І.К. | | | | |
| | | | | | | ХНУРЕ<br>кафедра КІТАМ |
| Н. контр. | | Яшков І.О. | | | | |
| Затв. | | Невлюдов І.Ш. | | | | |