Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Самостоятельная работа № 1

по курсу «Разработка мобильных приложений»

«Работа с протоколом MQTT в Dart»

Студент группы ИУ9-71Б Окутин Д. А.

Преподаватель Посевин Д. П.

Москва 2024

1 Цель

Цель данной лабораторной работы: разобраться с работой с MQTT в языке Dart.

2 Задание

1. Поставить библиотеку по работе с mqtt.
2. Запустить пример кода и проверить, что всё работает.
3. Реализовать обёртку над примером взаимодействия.

3 Реализация

Исходный код представлен в листинге 1.

Листинг 1: Исхоный код программы

```

  import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HomeScreen(), //Scaffold
      debugShowCheckedModeBanner: false,
    );
  }
}

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("          4"),
        backgroundColor: Colors.greenAccent[400],
        elevation: 50.0,
      ), //AppBar
      drawer: Drawer(
        child: ListView(
          padding: EdgeInsets.zero,
          children: <Widget>[
            Container(
              height: 150.0,
              color: Colors.greenAccent[400],
              child: Center(
                child: Text(
                  '          ',
                  style: TextStyle(
                    color: Colors.black,
                    fontSize: 24,
                  ),
                ),
              ),
            ),
            ListTile(
              title: Text('          2'),
              onTap: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => Lab2(title: 'Lab2',)),
                );
              },
            ),
            ListTile(
              title: Text('          3'),
              onTap: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => Lab3(title: 'Lab3',)),
                );
              },
            ),
          ],
        ),
      ),
      body: Center(
        child: Lab4(),
      ),//Center
    );
```

```dart
  }
}

class Lab4 extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: ParabolaPage(),
      debugShowCheckedModeBanner: false,
    );
  }
}

class ParabolaPage extends StatefulWidget {
  @override
  _ParabolaPageState createState() => _ParabolaPageState();
}

class _ParabolaPageState extends State<ParabolaPage>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  late Animation<double> _animation;

  double a = 1.0;
  double b = 0.0;
  double c = 0.0;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(seconds: 40),
      vsync: this,
    )..repeat(reverse: true);

    _animation = Tween<double>(begin: -15.0, end: 15.0).animate(_controller);
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        children: [
          Expanded(
            child: Center(
              child: AnimatedBuilder(
                animation: _animation,
                builder: (context, child) {
                  return CustomPaint(
                    size: Size(double.infinity, double.infinity),
                    painter: ParabolaPainter(_animation.value, b, c),
                  );
                },
              ),
            ),
          ),
          _buildSlider("a", -100.0, 100.0, (value) {
            setState(() {
              a = value;
            });
          }),
          _buildSlider("b", -100.0, 100.0, (value) {
            setState(() {
              b = value;
            });
          }),
          _buildSlider("c", -100.0, 100.0, (value) {
            setState(() {
              c = value;
            });
          }),
        ],
      ),
    );
  }

  Widget _buildSlider(String label, double min, double max, ValueChanged<double> onChanged) {
    return Column(
      children: [
        Text(label),
        Slider(
          value: label == "a" ? a : label == "b" ? b : c,
          min: min,
          max: max,
```

```dart
          onChanged: onChanged,
          divisions: 200,
          label: label == "a" ? a.toStringAsFixed(2) : label == "b" ? b.toStringAsFixed(2) : c.toStringAsFixed(2),
        ),
      ],
    );
  }
}

class ParabolaPainter extends CustomPainter {
  final double a;
  final double b;
  final double c;

  ParabolaPainter(this.a, this.b, this.c);

  @override
  void paint(Canvas canvas, Size size) {
    final paint = Paint()
      ..color = Colors.blue
      ..style = PaintingStyle.stroke
      ..strokeWidth = 2;

    final path = Path();

    //
    for (double x = -size.width / 2; x <= size.width / 2; x += 0.1) {
      double y = a * x * x /100+ b * x + c; //
                    y = ax^2 + bx + c

      if (x == -size.width / 2) {
        path.moveTo(size.width / 2 + x, size.height / 2 - y);
      } else {
        path.lineTo(size.width / 2 + x, size.height / 2 - y);
      }
    }

    canvas.drawPath(path, paint);
  }

  @override
  bool shouldRepaint(ParabolaPainter oldDelegate) {
    return true;
  }
}
```

```

class Lab2 extends StatefulWidget {
  const Lab2({Key? key, required this.title}) : super(key: key);
  final String title;

  @override
  State<Lab2> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<Lab2> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  void _getRequestOn() {
    setState(() {
      http.get(Uri.parse("http://iocontrol.ru/api/sendData/lab1_panel/
lab1/1")).then((response) {
        print("Response status: ${response.statusCode}");
        print("Response body: ${response.body}");
      }).catchError((error){
        print("Error: $error");
      });
    });
  }

  void _getRequestOff() {
    setState(() {
      http.get(Uri.parse("http://iocontrol.ru/api/sendData/lab1_panel/
lab1/0")).then((response) {
        print("Response status: ${response.statusCode}");
        print("Response body: ${response.body}");
      }).catchError((error){
        print("Error: $error");
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: Text(widget.title),
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              const Text(
                'You have pushed the button this many times:',
              ),
              Text(
                '$_counter',
                style: Theme.of(context).textTheme.headlineLarge,
              ),

              TextButton(
                style: ButtonStyle(
                  foregroundColor: MaterialStateProperty.all<Color>(Colors
    .blue),
                ),
                onPressed: _getRequestOn,
                child: Text('On'),
              ),

              TextButton(
                style: ButtonStyle(
                  foregroundColor: MaterialStateProperty.all<Color>(Colors
    .blue),
                ),
                onPressed: _getRequestOff,
                child: Text('Off'),
              )

            ],
          ),
        ),
        floatingActionButton: FloatingActionButton(
          onPressed: _incrementCounter,
          tooltip: 'Increment',
          child: const Icon(Icons.add),
        ),

      );
  }
}

class Lab3 extends StatefulWidget {
  const Lab3({Key? key, required this.title}) : super(key: key);
  final String title;

  @override
  State<Lab3> createState() => _NumberFormState();
}

class _NumberFormState extends State<Lab3> {
  final _formKey = GlobalKey<FormState>();
  final _numberController = TextEditingController();
  int _currentNumber = 0;

  Future<void> sendNumber() async {
    final url = Uri.parse('http://195.19.55.124:8080/');
    final response = await http.post(
      url,
      headers: {
        'Content-Type': 'text/plain',
      },
      body: '$_currentNumber',
    );

    if (response.statusCode == 200) {
      print('Num sent: ${_currentNumber}');
    } else {
      print('Error: ${response.statusCode}');
    }
  }
  Future<void> sendNumberFromNumberController() async {
    final url = Uri.parse('http://195.19.55.124:8080/');
    final response = await http.post(
      url,
      headers: {
        'Content-Type': 'text/plain',
      },
      body: _numberController.text,
    );

    if (response.statusCode == 200) {
      print('Num sent: ${_numberController.text}');
    } else {
      print('Error: ${response.statusCode}');
    }
  }

  Future<void> getNumber() async {
    final url = Uri.parse('http://195.19.55.124:8080/');

    final response = await http.get(url);

    if (response.statusCode == 200) {
      setState(() {
        _currentNumber = int.parse(response.body);
      });

      print('Num: $_currentNumber');
    } else {
      print('Error: ${response.statusCode}');
    }
  }

  void increment() {
    setState(() {
      _currentNumber++;
    });

    sendNumber();
  }

  void decrement() {
    setState(() {
      _currentNumber--;
    });

    sendNumber();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Lab3'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: Column(
            children: <Widget>[
              TextFormField(
                controller: _numberController,
                decoration: InputDecoration(labelText: '
              '),
                keyboardType: TextInputType.number,
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return '                          ';
                  }
                  return null;
                },
              ),
              const SizedBox(height: 20),
              ElevatedButton(
                onPressed: () {
                  if (_formKey.currentState!.validate()) {
                    sendNumberFromNumberController();
                  }
                },
                child: Text('                              '),
              ),
              SizedBox(height: 20),
              ElevatedButton(
                onPressed: getNumber,
                child: Text('                            '),
              ),
              const SizedBox(height: 20),
              Text('                         : $_currentNumber', style:
    TextStyle(fontSize: 20)),
              SizedBox(height: 20),
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  ElevatedButton(
                    onPressed: decrement,
                    child: Text('                  '),
                  ),
                  SizedBox(width: 20),
                  ElevatedButton(
                    onPressed: increment,
                    child: Text('                  '),
                  ),
                ],
              ),
            ],
          ),
        ),
      ),
    );
  }
}

class Fly2 extends StatefulWidget {
  const Fly2({Key? key, required this.title}) : super(key: key);
  final String title;

  @override
  State<Fly2> createState() => MyFormState();
}

class MyFormState extends State<Fly2> {
  final _formKey = GlobalKey<FormState>();
  String _body = "";

  final client = MqttServerClient('test.mosquitto.org', '');

  var pongCount = 0; // Pong counter

  Future AAA(String message) async {

    client.logging(on: true);
    client.setProtocolV311();
    client.keepAlivePeriod = 20;
    client.onDisconnected = onDisconnected;
    client.onConnected = onConnected;
    client.onSubscribed = onSubscribed;
    client.pongCallback = pong;

    print('Mosquitto client connecting....');


    try {
      await client.connect();
    } on NoConnectionException catch (e) {
      print('client exception - $e');
      client.disconnect();
    } on SocketException catch (e) {
      print('socket exception - $e');
      client.disconnect();
    }

    if (client.connectionStatus!.state == MqttConnectionState.connected)
    {
      print('Mosquitto client connected');
    } else {
      print('ERROR Mosquitto client connection failed - disconnecting, status is ${client.connectionStatus}');
      client.disconnect();
      exit(-1);
    }

    client.updates!.listen((List<MqttReceivedMessage<MqttMessage?>>? c) {
      final recMess = c![0].payload as MqttPublishMessage;
      final pt = MqttPublishPayload.bytesToStringAsString(recMess.payload.message);
      print('Change notification:: ---------------> topic is <${c[0].topic}>, payload is <-- $pt -->');
      _body = "--> ${pt}";
      print('');
    });
    client.published!.listen((MqttPublishMessage message) {
      print('Published notification:: topic is ${message.variableHeader!.topicName}, with Qos ${message.header!.qos}');
    });

    const pubTopic = 'IU/9';
    final builder = MqttClientPayloadBuilder();
    builder.addString('Dart say ${message}');
    _body = "--> ${message}";

    print('Subscribing to the UI/9 topic');
    client.subscribe(pubTopic, MqttQos.exactlyOnce);

    print('Publishing our topic');
    client.publishMessage(pubTopic, MqttQos.exactlyOnce, builder.payload!);

    print('Sleeping.... 60 sec');    /// Ok, we will now sleep a while, in this gap you will see ping request/response messages being exchanged by the keep alive mechanism.
    await MqttUtilities.asyncSleep(60);
    print('Awaked');
    print('Unsubscribing....');
    client.unsubscribe(pubTopic);


    await MqttUtilities.asyncSleep(2); /// Wait for the unsubscribe message from the broker if you wish.
    print('Disconnecting ...');
```

```
    client.disconnect();
    print('Stopped! Bye!....');

  }

  void onSubscribed(String topic) {
    print('Subscription confirmed for topic $topic');
  }

  void onDisconnected() {
    print('OnDisconnected client callback - Client disconnection');
    if (client.connectionStatus!.disconnectionOrigin ==
        MqttDisconnectionOrigin.solicited) {
      print('OnDisconnected callback is solicited, this is correct');
    } else {
      print('OnDisconnected callback is unsolicited or none, this is
   incorrect - exiting');
      exit(-1);
    }
    if (pongCount == 3) {
      print('Pong count is correct');
    } else {
      print('Pong count is incorrect, expected 3. actual $pongCount');
    }
  }

  void onConnected() {
    print('OnConnected client callback - Client connection was
   successful');
  }

  void pong() {
    print('Ping response client callback invoked');
    _body = 'Ping response client callback invoked';
    pongCount++;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: Text('Lab3'),
        ),
        body:      Padding(
        padding: EdgeInsets.all(10.0),
        child: new Form(
            key: _formKey,
            child: new Column(
              children: <Widget>[
      new Text('                        :', style: TextStyle(fontSize:
    20.0),),
      new TextFormField(validator: (value) {
        if (value == null || value.isEmpty)
        {
          return '                        -                        !';
        }
        else
        {
          print('---->'+value);
          _body = value;


          AAA(value);

        }
      }),

      new SizedBox(height: 20.0),

      ElevatedButton(
        child: Text('Button'),
        onPressed: () {

          if(_formKey.currentState!.validate()) ScaffoldMessenger.of(
    context).showSnackBar(SnackBar(content: Text('
                  !'+_body), backgroundColor: Colors.red,));

        },
        style: ElevatedButton.styleFrom(
            backgroundColor: Colors.purple,
            padding: EdgeInsets.symmetric(horizontal: 50, vertical: 20),
            textStyle: TextStyle(
                fontSize: 30,
                fontWeight: FontWeight.bold)),
      ),

    ],)))),);
  }
}
```

4 Результаты

Результаты представлен на рисунке 1.

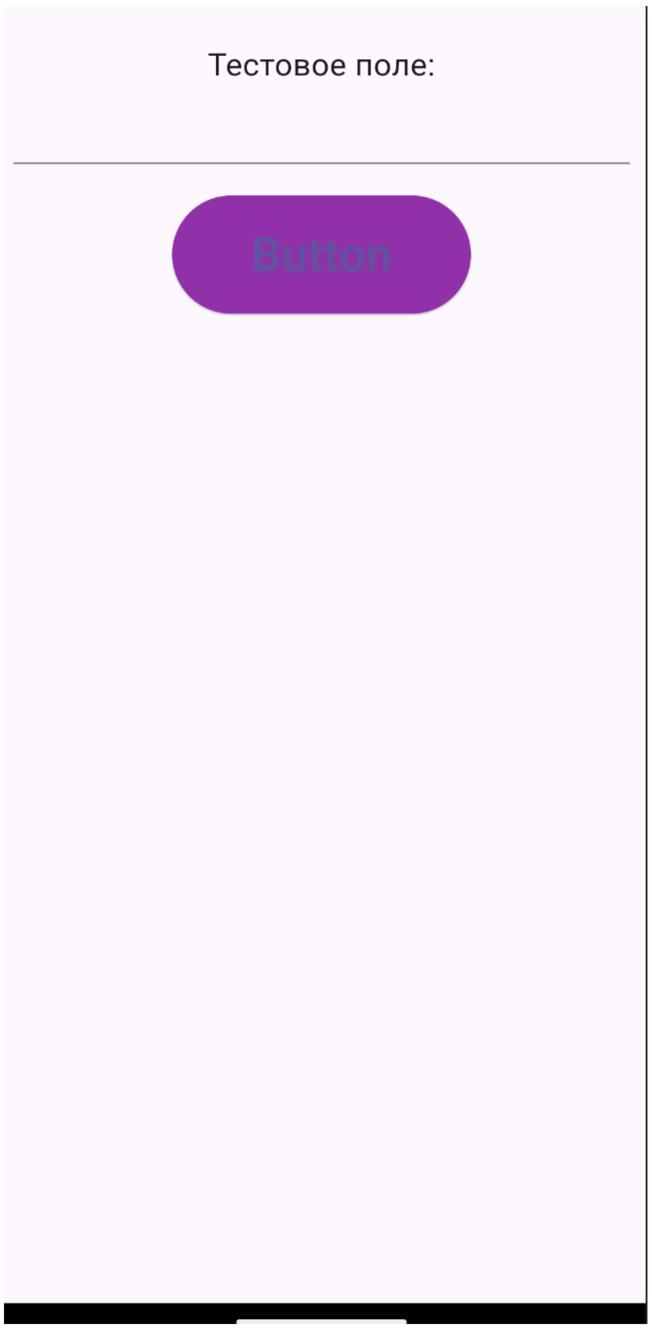

Рис. 1 — Интерфейс взаимодействия

5 Выводы

В результате данной лабороторной работы были изучены методологии работы с MQTT в языке Dart и получено практическиое применение освоенного материала.