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»

Летучка № 2

по курсу «Разработка мобильных приложений»

«Передача и чтение данных по протоколу MQTT»

Студент группы ИУ9-71Б Яровикова А. С.

Преподаватель Посевин Д. П.

Москва 2023

1 Цель

Цель данной лабораторной работы: создание приложения, которое передает и принимает данные по протоколу MQTT.

2 Задание

1. реализовать форму отправки и чтения данных из топика "IU/9"MQTT-брокера test.mosquitto.org

2. форма должна содержать следующие элементы:

- поле ввода строки сообщения, которое записывается в топик "IU/9"с соответствующей ей кнопкой "send";

- кнопка "read по нажатию на которую происходит чтение значения записанного в топик "IU/9 при этом результат получения значения из топика "IU/9"должно выводиться рядом с кнопкой "read".

3 Реализация

Исходный код представлен в листинге 1.

Листинг 1: main.dart

```
1  import 'dart:async';
2  import 'dart:convert'; // <--          JSON
3  import 'dart:io';
4  import 'package:flutter/material.dart';
5  import 'package:http/http.dart' as http;
6  import 'package:mqtt_client/mqtt_client.dart';
7  import 'package:mqtt_client/mqtt_server_client.dart';
8
9  class MyForm extends StatefulWidget {
10   @override
11   State<StatefulWidget> createState() => MyFormState();
12 }
13
14 class MyFormState extends State {
15
16   final _formKey = GlobalKey<FormState>();
17   String _body = "";
18   String _body2="";
```

```
19  final client = MqttServerClient('test.mosquitto.org', '');
20
21  var pongCount = 0; // Pong counter
22
23  Future AAA(String message) async {
24
25    client.logging(on: true);
26    client.setProtocolV311();
27    client.keepAlivePeriod = 20;
28    client.onDisconnected = onDisconnected;
29    client.onConnected = onConnected;
30    client.onSubscribed = onSubscribed;
31    client.pongCallback = pong;
32
33    print('Mosquitto client connecting....');
34
35    try {
36      await client.connect();
37    } on NoConnectionException catch (e) {
38      print('client exception - $e');
39      client.disconnect();
40    } on SocketException catch (e) {
41      print('socket exception - $e');
42      client.disconnect();
43    }
44
45    if (client.connectionStatus!.state == MqttConnectionState.connected)
      {
46      print('Mosquitto client connected');
47    } else {
48      print('ERROR Mosquitto client connection failed - disconnecting,
    status is ${client.connectionStatus}');
49      client.disconnect();
50      exit(-1);
51    }
52    client.updates!.listen((List<MqttReceivedMessage<MqttMessage?>>? c)
    {
53      final recMess = c![0].payload as MqttPublishMessage;
54      final pt = MqttPublishPayload.bytesToStringAsString(recMess.
    payload.message);
55      print('Change notification:: ---------------> topic is <${c[0].
    topic}>, payload is <-- $pt -->');
56      _body = "${pt}";
57      print('');
58    });
59    client.published!.listen((MqttPublishMessage message) {
```

```
60        print('Published notification:: topic is ${message.variableHeader
      !.topicName}, with Qos ${message.header!.qos}');
61      });
62
63      const pubTopic = 'IU/9';
64      final builder = MqttClientPayloadBuilder();
65      builder.addString('Dart say ${message}');
66      //_body = "--> ${message}";
67
68      print('Subscribing to the UI/9 topic');
69      client.subscribe(pubTopic, MqttQos.exactlyOnce);
70
71      print('Publishing our topic');
72      client.publishMessage(pubTopic, MqttQos.exactlyOnce, builder.payload
      !);
73
74      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.
75      await MqttUtilities.asyncSleep(60);
76      print('Awaked');
77      print('Unsubscribing....');
78      client.unsubscribe(pubTopic);
79
80      await MqttUtilities.asyncSleep(2); /// Wait for the unsubscribe
      message from the broker if you wish.
81      print('Disconnecting ...');
82      client.disconnect();
83      print('Stopped! Bye!....');
84
85    }
86
87    void onSubscribed(String topic) {
88      print('Subscription confirmed for topic $topic');
89    }
90
91    void onDisconnected() {
92      print('OnDisconnected client callback - Client disconnection');
93      if (client.connectionStatus!.disconnectionOrigin ==
94          MqttDisconnectionOrigin.solicited) {
95        print('OnDisconnected callback is solicited, this is correct');
96      } else {
97        print('OnDisconnected callback is unsolicited or none, this is
      incorrect - exiting');
98        exit(-1);
99      }
```

```
100       if (pongCount == 3) {
101         print('Pong count is correct');
102       } else {
103         print('Pong count is incorrect, expected 3. actual $pongCount');
104       }
105     }
106
107     void onConnected() {
108       print('OnConnected client callback - Client connection was
          successful');
109     }
110
111     void pong() {
112       print('Ping response client callback invoked');
113       _body = 'Ping response client callback invoked';
114       pongCount++;
115     }
116
117     Widget build(BuildContext context) {
118       return Container(padding: EdgeInsets.all(10.0), child: new Form(key:
          _formKey, child: new Column(children: <Widget>[
119         new Text('                              :', style: TextStyle(fontSize:
          20.0),),
120         new TextFormField(validator: (value) {
121           if (value == null || value.isEmpty)
122           {
123             return '                                -                          !';
124           }
125           else
126           {
127             print('---->'+value);
128             //_body = value;
129             AAA(value);
130           }
131         }),
132
133         new SizedBox(height: 20.0),
134
135         ElevatedButton(
136           child: Text('send'),
137           onPressed: () {
138             if(_formKey.currentState!.validate()) print("ok - data was
          sent");
139           },
140           style: ElevatedButton.styleFrom(
141               primary: Colors.green,
```

```
142             padding: EdgeInsets.symmetric(horizontal: 50, vertical: 20),
143             textStyle: TextStyle(
144                 fontSize: 30,
145                 fontWeight: FontWeight.bold)),
146       ),
147       new SizedBox(height: 40.0),
148
149       ElevatedButton(
150         child: Text('read'),
151         onPressed: () {
152           setState(() {
153             _body2=_body;
154           });
155         },
156         style: ElevatedButton.styleFrom(
157             primary: Colors.purple,
158             padding: EdgeInsets.symmetric(horizontal: 50, vertical: 20),
159             textStyle: TextStyle(
160                 fontSize: 30,
161                 fontWeight: FontWeight.bold)),
162       ),
163       new Text(_body2, style: TextStyle(fontSize: 20.0),),
164     ],)));
165   }
166 }
167
168 void main() => runApp(
169     new MaterialApp(
170         debugShowCheckedModeBanner: false,
171         home: new Scaffold(
172             appBar: new AppBar(title: new Text('IU9 -
                  ')),
173             body: new MyForm()
174         )
175     )
176 );
```

4 Результаты

Результат представлен на рисунке 1.

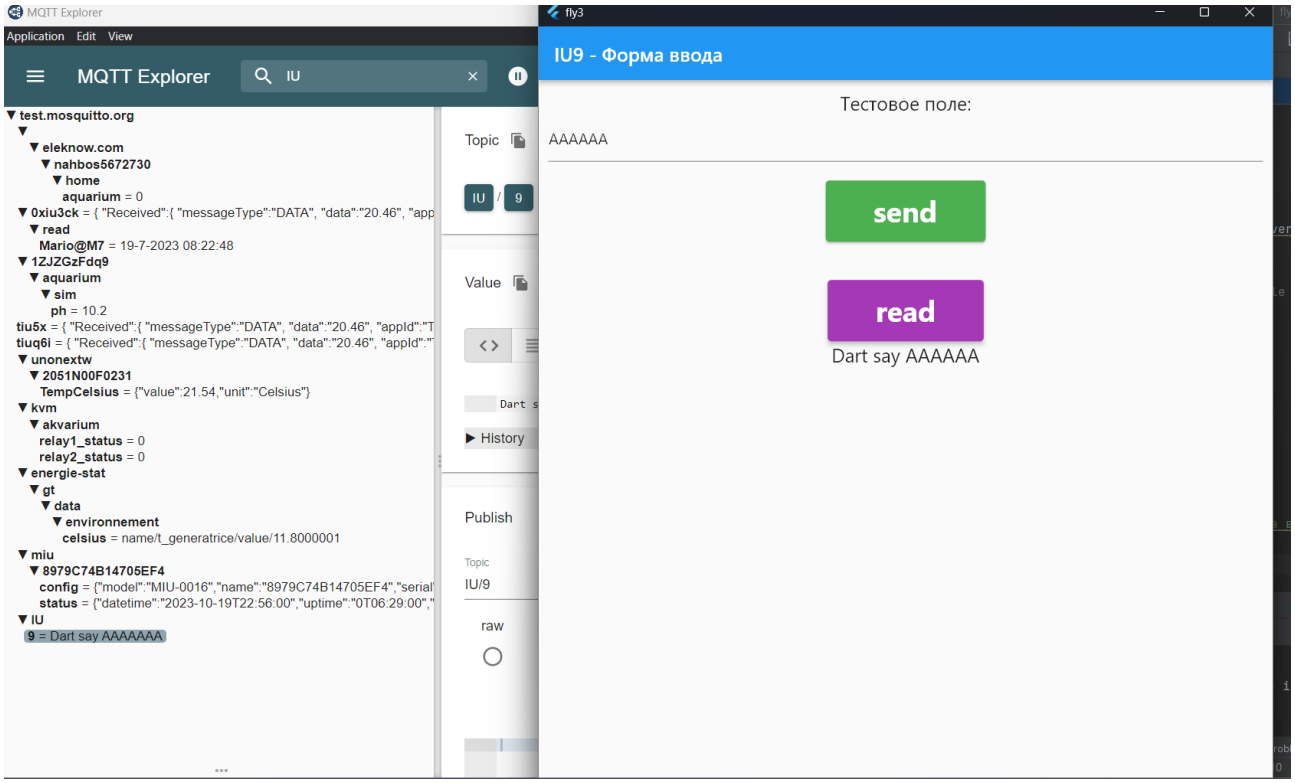

Рис. 1 — Результат работы

5 Выводы

В результате работы было создано небольшое приложение, которое использует сетевой протокол MQTT для передачи и получения данных.