



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

```
1
2 import 'package:flutter/material.dart';
3 import 'package:http/http.dart' as http;
4
5 void main() {
6   runApp(MyApp());
7 }
8
9 class MyApp extends StatelessWidget {
10   @override
11   Widget build(BuildContext context) {
12     return MaterialApp(
13       home: HomeScreen(), // Scaffold
14       debugShowCheckedModeBanner: false,
15     );
16   }
17 }
18
19 class HomeScreen extends StatelessWidget {
20   @override
21   Widget build(BuildContext context) {
22     return Scaffold(
23       appBar: AppBar(
24         title: const Text("          4"),
25         backgroundColor: Colors.greenAccent[400],
```

```

26         elevation: 50.0,
27     ), //AppBar
28     drawer: Drawer(
29         child: ListView(
30             padding: EdgeInsets.zero,
31             children: <Widget>[
32                 Container(
33                     height: 150.0,
34                     color: Colors.greenAccent[400],
35                     child: Center(
36                         child: Text(
37                             '          ',
38                             style: TextStyle(
39                                 color: Colors.black,
40                                 fontSize: 24,
41                             ),
42                         ),
43                     ),
44                 ),
45                 ListTile(
46                     title: Text('          2'),
47                     onTap: () {
48                         Navigator.push(
49                             context,
50                             MaterialPageRoute(builder: (context) => Lab2(title: '
Lab2',)),
51                     );
52                 },
53             ),
54                 ListTile(
55                     title: Text('          3'),
56                     onTap: () {
57                         Navigator.push(
58                             context,
59                             MaterialPageRoute(builder: (context) => Lab3(title: '
Lab3',)),
60                     );
61                 },
62             ),
63             ],
64         ),
65     ),
66     body: Center(
67         child: Lab4(),
68     ), //Center
69 );

```

```

70     }
71 }
72
73 class Lab4 extends StatelessWidget {
74     @override
75     Widget build(BuildContext context) {
76         return MaterialApp(
77             home: ParabolaPage(),
78             debugShowCheckedModeBanner: false,
79         );
80     }
81 }
82
83 class ParabolaPage extends StatefulWidget {
84     @override
85     _ParabolaPageState createState() => _ParabolaPageState();
86 }
87
88 class _ParabolaPageState extends State<ParabolaPage>
89     with SingleTickerProviderStateMixin {
90     late AnimationController _controller;
91     late Animation<double> _animation;
92
93     double a = 1.0;
94     double b = 0.0;
95     double c = 0.0;
96
97     @override
98     void initState() {
99         super.initState();
100         _controller = AnimationController(
101             duration: const Duration(seconds: 40),
102             vsync: this,
103         )..repeat(reverse: true);
104
105         _animation = Tween<double>(begin: -15.0, end: 15.0).animate(
106             _controller);
107
108     @override
109     void dispose() {
110         _controller.dispose();
111         super.dispose();
112     }
113
114     @override

```

```

115 Widget build(BuildContext context) {
116   return Scaffold(
117     body: Column(
118       children: [
119         Expanded(
120           child: Center(
121             child: AnimatedBuilder(
122               animation: _animation,
123               builder: (context, child) {
124                 return CustomPaint(
125                   size: Size(double.infinity, double.infinity),
126                   painter: ParabolaPainter(_animation.value, b, c),
127                 );
128               },
129             ),
130           ),
131         ),
132         _buildSlider("a", -100.0, 100.0, (value) {
133           setState(() {
134             a = value;
135           });
136         }),
137         _buildSlider("b", -100.0, 100.0, (value) {
138           setState(() {
139             b = value;
140           });
141         }),
142         _buildSlider("c", -100.0, 100.0, (value) {
143           setState(() {
144             c = value;
145           });
146         }),
147       ],
148     ),
149   );
150 }
151
152 Widget _buildSlider(String label, double min, double max, ValueChanged
153   <double> onChanged) {
154   return Column(
155     children: [
156       Text(label),
157       Slider(
158         value: label == "a" ? a : label == "b" ? b : c,
159         min: min,
160         max: max,

```

```

160         onChanged: onChanged,
161         divisions: 200,
162         label: label == "a" ? a.toStringAsFixed(2) : label == "b" ? b.
toStringAsFixed(2) : c.toStringAsFixed(2),
163     ),
164 ],
165 );
166 }
167 }
168
169 class ParabolaPainter extends CustomPainter {
170     final double a;
171     final double b;
172     final double c;
173
174     ParabolaPainter(this.a, this.b, this.c);
175
176     @override
177     void paint(Canvas canvas, Size size) {
178         final paint = Paint()
179             ..color = Colors.blue
180             ..style = PaintingStyle.stroke
181             ..strokeWidth = 2;
182
183         final path = Path();
184
185         //
186         for (double x = -size.width / 2; x <= size.width / 2; x += 0.1) {
187             double y = a * x * x / 100 + b * x + c; //

$$y = ax^2 + bx + c$$

188
189             if (x == -size.width / 2) {
190                 path.moveTo(size.width / 2 + x, size.height / 2 - y);
191             } else {
192                 path.lineTo(size.width / 2 + x, size.height / 2 - y);
193             }
194         }
195
196         canvas.drawPath(path, paint);
197     }
198
199     @override
200     bool shouldRepaint(ParabolaPainter oldDelegate) {
201         return true;
202     }
203 }

```

```

204
205 class Lab2 extends StatefulWidget {
206     const Lab2({Key? key, required this.title}) : super(key: key);
207     final String title;
208
209     @override
210     State<Lab2> createState() => _MyHomePageState();
211 }
212
213 class _MyHomePageState extends State<Lab2> {
214     int _counter = 0;
215
216     void _incrementCounter() {
217         setState(() {
218             _counter++;
219         });
220     }
221
222     void _getRequestOn() {
223         setState(() {
224             http.get(Uri.parse("http://iocontrol.ru/api/sendData/lab1_panel/
lab1/1")).then((response) {
225                 print("Response status: ${response.statusCode}");
226                 print("Response body: ${response.body}");
227             }).catchError((error){
228                 print("Error: $error");
229             });
230         });
231     }
232
233     void _getRequestOff() {
234         setState(() {
235             http.get(Uri.parse("http://iocontrol.ru/api/sendData/lab1_panel/
lab1/0")).then((response) {
236                 print("Response status: ${response.statusCode}");
237                 print("Response body: ${response.body}");
238             }).catchError((error){
239                 print("Error: $error");
240             });
241         });
242     }
243
244     @override
245     Widget build(BuildContext context) {
246         return Scaffold(
247             appBar: AppBar(

```

```

248         title: Text(widget.title),
249     ),
250     body: Center(
251         child: Column(
252             mainAxisAlignment: MainAxisAlignment.center,
253             children: <Widget>[
254                 const Text(
255                     'You have pushed the button this many times:',
256                 ),
257                 Text(
258                     '$_counter',
259                     style: Theme.of(context).textTheme.headlineLarge,
260                 ),
261
262                 TextButton(
263                     style: ButtonStyle(
264                         foregroundColor: MaterialStateProperty.all<Color>(Colors
265 .blue),
266                     ),
267                     onPressed: _getRequestOn,
268                     child: Text('On'),
269                 ),
270
271                 TextButton(
272                     style: ButtonStyle(
273                         foregroundColor: MaterialStateProperty.all<Color>(Colors
274 .blue),
275                     ),
276                     onPressed: _getRequestOff,
277                     child: Text('Off'),
278                 ),
279             ],
280         ),
281         floatingActionButton: FloatingActionButton(
282             onPressed: _incrementCounter,
283             tooltip: 'Increment',
284             child: const Icon(Icons.add),
285         ),
286     );
287 }
288 }
289 }
290
291 class Lab3 extends StatefulWidget {

```



```

292     const Lab3({Key? key, required this.title}) : super(key: key);
293     final String title;
294
295     @override
296     State<Lab3> createState() => _NumberFormState();
297 }
298
299 class _NumberFormState extends State<Lab3> {
300     final _formKey = GlobalKey<FormState>();
301     final _numberController = TextEditingController();
302     int _currentNumber = 0;
303
304     Future<void> sendNumber() async {
305         final url = Uri.parse('http://195.19.55.124:8080/');
306         final response = await http.post(
307             url,
308             headers: {
309                 'Content-Type': 'text/plain',
310             },
311             body: '$_currentNumber',
312         );
313
314         if (response.statusCode == 200) {
315             print('Num sent: ${_currentNumber}');
316         } else {
317             print('Error: ${response.statusCode}');
318         }
319     }
320
321     Future<void> sendNumberFromNumberController() async {
322         final url = Uri.parse('http://195.19.55.124:8080/');
323         final response = await http.post(
324             url,
325             headers: {
326                 'Content-Type': 'text/plain',
327             },
328             body: _numberController.text,
329         );
330
331         if (response.statusCode == 200) {
332             print('Num sent: ${_numberController.text}');
333         } else {
334             print('Error: ${response.statusCode}');
335         }
336     }
337
338     Future<void> getNumber() async {

```

```

338     final url = Uri.parse('http://195.19.55.124:8080/');
339
340     final response = await http.get(url);
341
342     if (response.statusCode == 200) {
343         setState(() {
344             _currentNumber = int.parse(response.body);
345         });
346
347         print('Num: $_currentNumber');
348     } else {
349         print('Error: ${response.statusCode}');
350     }
351 }
352
353 void increment() {
354     setState(() {
355         _currentNumber++;
356     });
357
358     sendNumber();
359 }
360
361 void decrement() {
362     setState(() {
363         _currentNumber--;
364     });
365
366     sendNumber();
367 }
368
369 @override
370 Widget build(BuildContext context) {
371     return Scaffold(
372         appBar: AppBar(
373             title: Text('Lab3'),
374         ),
375         body: Padding(
376             padding: const EdgeInsets.all(16.0),
377             child: Form(
378                 key: _formKey,
379                 child: Column(
380                     children: <Widget>[
381                         TextFormField(
382                             controller: _numberController,

```

```

383         decoration: InputDecoration(labelText: '
384     '),
385     keyboardType: TextInputType.number,
386     validator: (value) {
387         if (value == null || value.isEmpty) {
388             return '
389         }
390         return null;
391     },
392     ),
393     const SizedBox(height: 20),
394     ElevatedButton(
395         onPressed: () {
396             if (_formKey.currentState!.validate()) {
397                 sendNumberFromNumberController();
398             }
399             },
400         child: Text('
401     ),
402     const SizedBox(height: 20),
403     ElevatedButton(
404         onPressed: getNumber,
405         child: Text('
406     ),
407     const SizedBox(height: 20),
408     Text('
409     : $_currentNumber', style:
410     TextStyle(fontSize: 20)),
411     const SizedBox(height: 20),
412     Row(
413         mainAxisAlignment: MainAxisAlignment.center,
414         children: [
415             ElevatedButton(
416                 onPressed: decrement,
417                 child: Text('
418             ),
419             const SizedBox(width: 20),
420             ElevatedButton(
421                 onPressed: increment,
422                 child: Text('
423             ),
424         ],
425     ),
426 )

```

```

427     );
428 }
429 }
430
431 class Fly2 extends StatefulWidget {
432     const Fly2({Key? key, required this.title}) : super(key: key);
433     final String title;
434
435     @override
436     State<Fly2> createState() => MyFormState();
437 }
438
439 class MyFormState extends State<Fly2> {
440     final _formKey = GlobalKey<FormState>();
441     String _body = "";
442
443     final client = MqttServerClient('test.mosquitto.org', '');
444
445     var pongCount = 0; // Pong counter
446
447     Future AAA(String message) async {
448
449         client.logging(on: true);
450         client.setProtocolV311();
451         client.keepAlivePeriod = 20;
452         client.onDisconnected = onDisconnected;
453         client.onConnected = onConnected;
454         client.onSubscribed = onSubscribed;
455         client.pongCallback = pong;
456
457         print('Mosquitto client connecting....');
458
459
460         try {
461             await client.connect();
462         } on NoConnectionException catch (e) {
463             print('client exception - $e');
464             client.disconnect();
465         } on SocketException catch (e) {
466             print('socket exception - $e');
467             client.disconnect();
468         }
469
470         if (client.connectionStatus!.state == MqttConnectionState.connected)
471         {
472             print('Mosquitto client connected');

```

```

472     } else {
473         print('ERROR Mosquitto client connection failed - disconnecting ,
status is ${client.connectionStatus}');
474         client.disconnect();
475         exit(-1);
476     }
477
478     client.updates!.listen((List<MqttReceivedMessage<MqttMessage?>>? c)
{
479         final recMess = c![0].payload as MqttPublishMessage;
480         final pt = MqttPublishPayload.bytesToStringAsString(recMess.
payload.message);
481         print('Change notification:: -----> topic is <${c[0].
topic}>, payload is <-- $pt -->');
482         _body = "--> ${pt}";
483         print('');
484     });
485     client.published!.listen((MqttPublishMessage message) {
486         print('Published notification:: topic is ${message.variableHeader
!.topicName}, with Qos ${message.header!.qos}');
487     });
488
489     const pubTopic = 'IU/9';
490     final builder = MqttClientPayloadBuilder();
491     builder.addString('Dart say ${message}');
492     _body = "--> ${message}";
493
494     print('Subscribing to the UI/9 topic');
495     client.subscribe(pubTopic, MqttQos.exactlyOnce);
496
497     print('Publishing our topic');
498     client.publishMessage(pubTopic, MqttQos.exactlyOnce, builder.payload
!);
499
500     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.
501     await MqttUtilities.asyncSleep(60);
502     print('Awaked');
503     print('Unsubscribing .... ');
504     client.unsubscribe(pubTopic);
505
506
507     await MqttUtilities.asyncSleep(2); /// Wait for the unsubscribe
message from the broker if you wish.
508     print('Disconnecting ... ');

```

```

509     client.disconnect();
510     print('Stopped! Bye!.... ');
511
512 }
513
514 void onSubscribed(String topic) {
515     print('Subscription confirmed for topic $topic');
516 }
517
518 void onDisconnected() {
519     print('OnDisconnected client callback - Client disconnection');
520     if (client.connectionStatus!.disconnectionOrigin ==
521         MqttDisconnectionOrigin.solicited) {
522         print('OnDisconnected callback is solicited, this is correct');
523     } else {
524         print('OnDisconnected callback is unsolicited or none, this is
incorrect - exiting');
525         exit(-1);
526     }
527     if (pongCount == 3) {
528         print('Pong count is correct');
529     } else {
530         print('Pong count is incorrect, expected 3. actual $pongCount');
531     }
532 }
533
534 void onConnected() {
535     print('OnConnected client callback - Client connection was
successful');
536 }
537
538 void pong() {
539     print('Ping response client callback invoked');
540     _body = 'Ping response client callback invoked';
541     pongCount++;
542 }
543
544 @override
545 Widget build(BuildContext context) {
546     return Scaffold(
547         appBar: AppBar(
548             title: Text('Lab3'),
549         ),
550         body: Padding(
551             padding: EdgeInsets.all(10.0),
552             child: new Form(

```

```

553         key: _formKey,
554         child: new Column(
555             children: <Widget>[
556                 new Text('                :', style: TextStyle(fontSize:
20.0)),),
557                 new TextFormField(validator: (value) {
558                     if (value == null || value.isEmpty)
559                     {
560                         return '                -                !';
561                     }
562                     else
563                     {
564                         print('---->' + value);
565                         _body = value;
566
567
568                         AAA(value);
569
570                     }
571                 })),
572
573                 new SizedBox(height: 20.0),
574
575                 ElevatedButton(
576                     child: Text('Button'),
577                     onPressed: () {
578
579                         if(_formKey.currentState!.validate()) ScaffoldMessenger.of(
context).showSnackBar(SnackBar(content: Text('
!'+_body), backgroundColor: Colors.red,));
580
581                     },
582                     style: ElevatedButton.styleFrom(
583                         backgroundColor: Colors.purple,
584                         padding: EdgeInsets.symmetric(horizontal: 50, vertical: 20),
585                         textStyle: TextStyle(
586                             fontSize: 30,
587                             fontWeight: FontWeight.bold)),
588                 ),
589
590             ],)))););
591     }
592 }

```

4 Результаты

Результаты представен на рисунке 1.

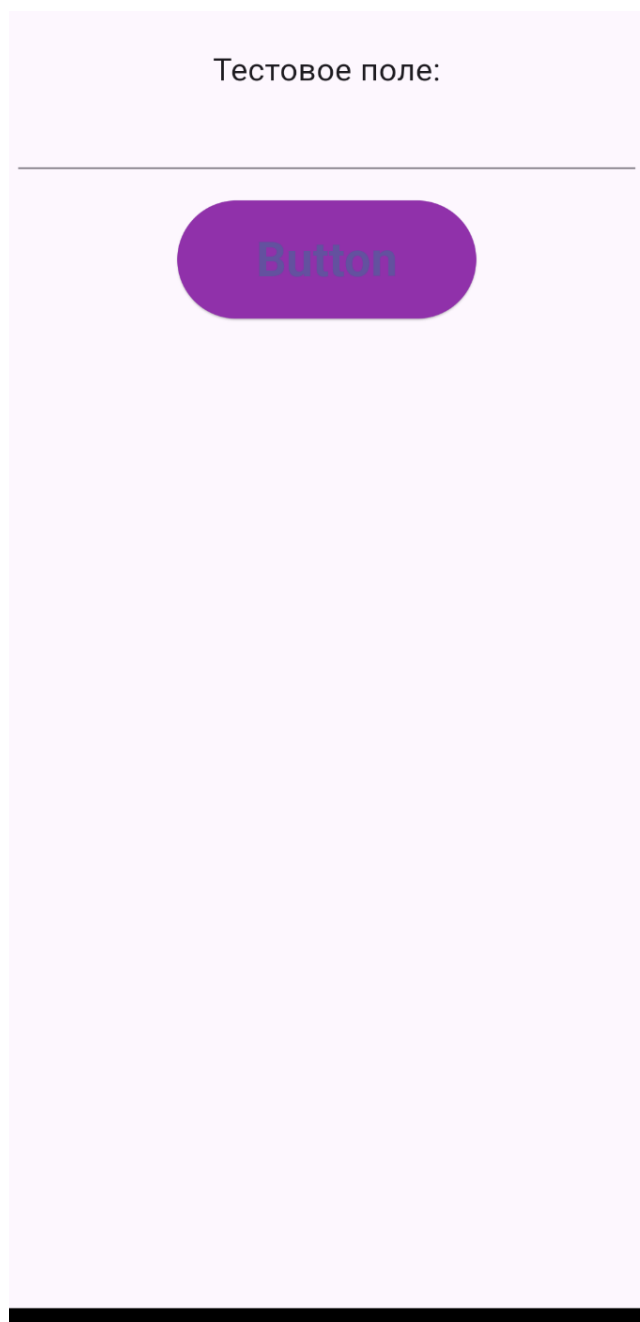


Рис. 1 — Интерфейс взаимодействия

5 Выводы

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