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»

Лабораторная работа № 5

по курсу «Разработка мобильных приложений»

«Продвинутая 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 и реализовать приложения для работы с несколькими топиками.

2 Задание

1. Реализовать приложение для записи вводимых пользователем данных в несколько топиков.

2. Интегрировать данную лабораторную работу с лабораторной работой номер 4.

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                          ListTile(
64                title: Text('          5'),
65                onTap: () {
66                  Navigator.push(
67                    context,
```

```
68                    MaterialPageRoute(builder: (context) => Lab5(title: '
      Lab5',)),
69                  );
70                },
71              ),
72            ],
73          ),
74        ),
75        body: Center(
76          child: Lab4(),
77        ),//Center
78      );
79    }
80  }
81
82  class Lab4 extends StatelessWidget {
83    @override
84    Widget build(BuildContext context) {
85      return MaterialApp(
86        home: ParabolaPage(),
87        debugShowCheckedModeBanner: false,
88      );
89    }
90  }
91
92  class ParabolaPage extends StatefulWidget {
93    @override
94    _ParabolaPageState createState() => _ParabolaPageState();
95  }
96
97  class _ParabolaPageState extends State<ParabolaPage>
98      with SingleTickerProviderStateMixin {
99    late AnimationController _controller;
100   late Animation<double> _animation;
101
102   double a = 1.0;
103   double b = 0.0;
104   double c = 0.0;
105
106   @override
107   void initState() {
108     super.initState();
109     _controller = AnimationController(
110       duration: const Duration(seconds: 40),
111       vsync: this,
112     )..repeat(reverse: true);
```

```
113
114       _animation = Tween<double>(begin: -15.0, end: 15.0).animate(
          _controller);
115     }
116
117     @override
118     void dispose() {
119       _controller.dispose();
120       super.dispose();
121     }
122
123     @override
124     Widget build(BuildContext context) {
125       return Scaffold(
126         body: Column(
127           children: [
128             Expanded(
129               child: Center(
130                 child: AnimatedBuilder(
131                   animation: _animation,
132                   builder: (context, child) {
133                     return CustomPaint(
134                       size: Size(double.infinity, double.infinity),
135                       painter: ParabolaPainter(_animation.value, b, c),
136                     );
137                   },
138                 ),
139               ),
140             ),
141             _buildSlider("a", -100.0, 100.0, (value) {
142               setState(() {
143                 a = value;
144               });
145             }),
146             _buildSlider("b", -100.0, 100.0, (value) {
147               setState(() {
148                 b = value;
149               });
150             }),
151             _buildSlider("c", -100.0, 100.0, (value) {
152               setState(() {
153                 c = value;
154               });
155             }),
156           ],
157         ),
```

```
158     );
159   }
160
161   Widget _buildSlider(String label, double min, double max, ValueChanged
       <double> onChanged) {
162     return Column(
163       children: [
164         Text(label),
165         Slider(
166           value: label == "a" ? a : label == "b" ? b : c,
167           min: min,
168           max: max,
169           onChanged: onChanged,
170           divisions: 200,
171           label: label == "a" ? a.toStringAsFixed(2) : label == "b" ? b.
       toStringAsFixed(2) : c.toStringAsFixed(2),
172         ),
173       ],
174     );
175   }
176 }
177
178 class ParabolaPainter extends CustomPainter {
179   final double a;
180   final double b;
181   final double c;
182
183   ParabolaPainter(this.a, this.b, this.c);
184
185   @override
186   void paint(Canvas canvas, Size size) {
187     final paint = Paint()
188       ..color = Colors.blue
189       ..style = PaintingStyle.stroke
190       ..strokeWidth = 2;
191
192     final path = Path();
193
194     //
195     for (double x = -size.width / 2; x <= size.width / 2; x += 0.1) {
196       double y = a * x * x /100+ b * x + c; //
                         y = ax^2 + bx + c
197
198       if (x == -size.width / 2) {
199         path.moveTo(size.width / 2 + x, size.height / 2 - y);
200       } else {
```

```
201         path.lineTo(size.width / 2 + x, size.height / 2 - y);
202       }
203     }
204
205     canvas.drawPath(path, paint);
206   }
207
208   @override
209   bool shouldRepaint(ParabolaPainter oldDelegate) {
210     return true;
211   }
212 }
213
214 class Lab2 extends StatefulWidget {
215   const Lab2({Key? key, required this.title}) : super(key: key);
216   final String title;
217
218   @override
219   State<Lab2> createState() => _MyHomePageState();
220 }
221
222 class _MyHomePageState extends State<Lab2> {
223   int _counter = 0;
224
225   void _incrementCounter() {
226     setState(() {
227       _counter++;
228     });
229   }
230
231   void _getRequestOn() {
232     setState(() {
233       http.get(Uri.parse("http://iocontrol.ru/api/sendData/lab1_panel/
    lab1/1")).then((response) {
234         print("Response status: ${response.statusCode}");
235         print("Response body: ${response.body}");
236       }).catchError((error){
237         print("Error: $error");
238       });
239     });
240   }
241
242   void _getRequestOff() {
243     setState(() {
244       http.get(Uri.parse("http://iocontrol.ru/api/sendData/lab1_panel/
    lab1/0")).then((response) {
```

```
245         print("Response status: ${response.statusCode}");
246         print("Response body: ${response.body}");
247       }).catchError((error){
248         print("Error: $error");
249       });
250     });
251   }
252
253   @override
254   Widget build(BuildContext context) {
255     return Scaffold(
256       appBar: AppBar(
257         title: Text(widget.title),
258       ),
259       body: Center(
260         child: Column(
261           mainAxisAlignment: MainAxisAlignment.center,
262           children: <Widget>[
263             const Text(
264               'You have pushed the button this many times:',
265             ),
266             Text(
267               '$_counter',
268               style: Theme.of(context).textTheme.headlineLarge,
269             ),
270
271             TextButton(
272               style: ButtonStyle(
273                 foregroundColor: MaterialStateProperty.all<Color>(Colors
    .blue),
274               ),
275               onPressed: _getRequestOn,
276               child: Text('On'),
277             ),
278
279             TextButton(
280               style: ButtonStyle(
281                 foregroundColor: MaterialStateProperty.all<Color>(Colors
    .blue),
282               ),
283               onPressed: _getRequestOff,
284               child: Text('Off'),
285             )
286
287           ],
288         ),
```

```
289         ),
290         floatingActionButton: FloatingActionButton(
291           onPressed: _incrementCounter,
292           tooltip: 'Increment',
293           child: const Icon(Icons.add),
294         ),
295
296       );
297     }
298 }
299
300 class Lab3 extends StatefulWidget {
301   const Lab3({Key? key, required this.title}) : super(key: key);
302   final String title;
303
304   @override
305   State<Lab3> createState() => _NumberFormState();
306 }
307
308 class _NumberFormState extends State<Lab3> {
309   final _formKey = GlobalKey<FormState>();
310   final _numberController = TextEditingController();
311   int _currentNumber = 0;
312
313   Future<void> sendNumber() async {
314     final url = Uri.parse('http://195.19.55.124:8080/');
315     final response = await http.post(
316       url,
317       headers: {
318         'Content-Type': 'text/plain',
319       },
320       body: '$_currentNumber',
321     );
322
323     if (response.statusCode == 200) {
324       print('Num sent: ${_currentNumber}');
325     } else {
326       print('Error: ${response.statusCode}');
327     }
328   }
329   Future<void> sendNumberFromNumberController() async {
330     final url = Uri.parse('http://195.19.55.124:8080/');
331     final response = await http.post(
332       url,
333       headers: {
334         'Content-Type': 'text/plain',
```

```
335         },
336         body: _numberController.text,
337       );
338
339       if (response.statusCode == 200) {
340         print('Num sent: ${_numberController.text}');
341       } else {
342         print('Error: ${response.statusCode}');
343       }
344     }
345
346     Future<void> getNumber() async {
347       final url = Uri.parse('http://195.19.55.124:8080/');
348
349       final response = await http.get(url);
350
351       if (response.statusCode == 200) {
352         setState(() {
353           _currentNumber = int.parse(response.body);
354         });
355
356         print('Num: $_currentNumber');
357       } else {
358         print('Error: ${response.statusCode}');
359       }
360     }
361
362     void increment() {
363       setState(() {
364         _currentNumber++;
365       });
366
367       sendNumber();
368     }
369
370     void decrement() {
371       setState(() {
372         _currentNumber--;
373       });
374
375       sendNumber();
376     }
377
378     @override
379     Widget build(BuildContext context) {
380       return Scaffold(
```

```
381       appBar: AppBar(
382         title: Text('Lab3'),
383       ),
384       body: Padding(
385         padding: const EdgeInsets.all(16.0),
386         child: Form(
387           key: _formKey,
388           child: Column(
389             children: <Widget>[
390               TextFormField(
391                 controller: _numberController,
392                 decoration: InputDecoration(labelText: '
                  '),
393                 keyboardType: TextInputType.number,
394                 validator: (value) {
395                   if (value == null || value.isEmpty) {
396                     return '                          ';
397                   }
398                   return null;
399                 },
400               ),
401               const SizedBox(height: 20),
402               ElevatedButton(
403                 onPressed: () {
404                   if (_formKey.currentState!.validate()) {
405                     sendNumberFromNumberController();
406                   }
407                 },
408                 child: Text('                          '),
409               ),
410               SizedBox(height: 20),
411               ElevatedButton(
412                 onPressed: getNumber,
413                 child: Text('                          '),
414               ),
415               const SizedBox(height: 20),
416               Text('                      : $_currentNumber', style:
      TextStyle(fontSize: 20)),
417               SizedBox(height: 20),
418               Row(
419                 mainAxisAlignment: MainAxisAlignment.center,
420                 children: [
421                   ElevatedButton(
422                     onPressed: decrement,
423                     child: Text('                '),
424                   ),
```

```
425                     SizedBox(width: 20),
426                     ElevatedButton(
427                       onPressed: increment,
428                       child: Text('                    '),
429                     ),
430                   ],
431                 ),
432               ],
433             ),
434           ),
435         ),
436       );
437   }
438 }
439
440 class Lab5 extends StatefulWidget {
441   const Lab5({Key? key, required this.title}) : super(key: key);
442   final String title;
443
444   @override
445   State<Lab5> createState() => MqttFormState();
446 }
447
448 class MqttFormState extends State<Lab5> {
449   final _formKey = GlobalKey<FormState>();
450   String _message1 = "";
451   String _message2 = "";
452   String _message3 = "";
453   String _receivedMessage = "";
454   String _receivedMessageBuf = "";
455
456   final client = MqttServerClient('test.mosquitto.org', '');
457
458   var pongCount = 0; //                Pong
459
460   @override
461   void initState() {
462     super.initState();
463     connect();
464   }
465
466   Future<void> connect() async {
467     client.logging(on: true);
468     client.setProtocolV311();
469     client.keepAlivePeriod = 20;
470     client.onDisconnected = onDisconnected;
```

```
471     client.onConnected = onConnected;
472     client.onSubscribed = onSubscribed;
473     client.pongCallback = pong;
474
475     print('Mosquitto client connecting....');
476
477     try {
478       await client.connect();
479     } on NoConnectionException catch (e) {
480       print('client exception - $e');
481       client.disconnect();
482     } on SocketException catch (e) {
483       print('socket exception - $e');
484       client.disconnect();
485     }
486
487     if (client.connectionStatus!.state == MqttConnectionState.connected)
        {
488       print('Mosquitto client connected');
489
490       client.subscribe('UI9/a', MqttQos.exactlyOnce);
491       client.subscribe('UI9/b', MqttQos.exactlyOnce);
492       client.subscribe('UI9/c', MqttQos.exactlyOnce);
493
494       client.updates!.listen((List<MqttReceivedMessage<MqttMessage?>>? c
      ) {
495         final recMess = c![0].payload as MqttPublishMessage;
496         final pt =
497         MqttPublishPayload.bytesToStringAsString(recMess.payload.message
      );
498         setState(() {
499           _receivedMessageBuf += pt + ' '; //

500         });
501       });
502     } else {
503       print(
504           'ERROR Mosquitto client connection failed - disconnecting,
      status is ${client.connectionStatus}');
505       client.disconnect();
506     }
507   }
508
509   Future<void> sendMessage(String pubTopic, String msg) async {
510     final builder = MqttClientPayloadBuilder();
511
```

```
512       builder.addString('$msg');
513
514       print('Publishing to topic $pubTopic');
515       client.publishMessage(pubTopic, MqttQos.exactlyOnce, builder.payload
          !);
516     }
517
518     void onSubscribed(String topic) {
519       print('Subscription confirmed for topic $topic');
520     }
521
522     void onDisconnected() {
523       print('OnDisconnected client callback - Client disconnection');
524       if (client.connectionStatus!.disconnectionOrigin ==
525           MqttDisconnectionOrigin.solicited) {
526         print('OnDisconnected callback is solicited, this is correct');
527       } else {
528         print(
529             'OnDisconnected callback is unsolicited or none, this is
          incorrect - exiting');
530         exit(-1);
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       pongCount++;
541     }
542
543     @override
544     void dispose() {
545       client.disconnect();
546       super.dispose();
547     }
548
549     @override
550     Widget build(BuildContext context) {
551       return Scaffold(
552         appBar: AppBar(
553           title: Text('Lab5'),
554         ),
```

```
555         body: Padding(
556           padding: const EdgeInsets.all(16.0),
557           child: Form(
558             key: _formKey,
559             child: Column(
560               crossAxisAlignment: CrossAxisAlignment.stretch,
561               children: <Widget>[
562                 TextFormField(
563                   decoration: InputDecoration(labelText: '
                          '),
564                   onChanged: (value) {
565                     _message1 = value;
566                   },
567                   validator: (value) {
568                     if (value == null || value.isEmpty) {
569                       return '                    ,
                          ';
570                     }
571                     return null;
572                   },
573                 ),
574                 SizedBox(height: 16),
575                 TextFormField(
576                   decoration: InputDecoration(labelText: '
                          '),
577                   onChanged: (value) {
578                     _message2 = value;
579                   },
580                   validator: (value) {
581                     if (value == null || value.isEmpty) {
582                       return '                    ,
                          ';
583                     }
584                     return null;
585                   },
586                 ),
587                 SizedBox(height: 16),
588                 TextFormField(
589                   decoration: InputDecoration(labelText: '
                          '),
590                   onChanged: (value) {
591                     _message3 = value;
592                   },
593                   validator: (value) {
594                     if (value == null || value.isEmpty) {
```

```
595                     return '                    ,
                          ';
596                   }
597                   return null;
598                 },
599               ),
600               SizedBox(height: 16),
601               ElevatedButton(
602                 onPressed: () {
603                   if (_formKey.currentState!.validate()) {
604                     sendMessage("UI9/a", '[UI9/a]'+_message1);
605                     sendMessage("UI9/b", '[UI9/b]'+_message2);
606                     sendMessage("UI9/c", '[UI9/c]'+_message3);
607                   }
608                 },
609                 child: Text('                                    '),
610               ),
611               SizedBox(height: 20),
612               ElevatedButton(
613                 onPressed: () {
614                   setState(() {
615                     _receivedMessage = _receivedMessageBuf; //
                          
616                     _receivedMessageBuf = '';
617                   });
618                 },
619                 child: Text('                                  '),
620               ),
621               Text('                                  :
    $_receivedMessage'),
622             ],
623           ),
624         ),
625       ),
626     );
627   }
628 }
629
630 class Fly2 extends StatefulWidget {
631   const Fly2({Key? key, required this.title}) : super(key: key);
632   final String title;
633
634   @override
635   State<Fly2> createState() => MyFormState();
636 }
637
```

```
638 class MyFormState extends State<Fly2> {
639   final _formKey = GlobalKey<FormState>();
640   String _body = "";
641
642   final client = MqttServerClient('test.mosquitto.org', '');
643
644   var pongCount = 0; // Pong counter
645
646   Future AAA(String message) async {
647
648     client.logging(on: true);
649     client.setProtocolV311();
650     client.keepAlivePeriod = 20;
651     client.onDisconnected = onDisconnected;
652     client.onConnected = onConnected;
653     client.onSubscribed = onSubscribed;
654     client.pongCallback = pong;
655
656     print('Mosquitto client connecting....');
657
658
659     try {
660       await client.connect();
661     } on NoConnectionException catch (e) {
662       print('client exception - $e');
663       client.disconnect();
664     } on SocketException catch (e) {
665       print('socket exception - $e');
666       client.disconnect();
667     }
668
669     if (client.connectionStatus!.state == MqttConnectionState.connected)
      {
670       print('Mosquitto client connected');
671     } else {
672       print('ERROR Mosquitto client connection failed - disconnecting,
     status is ${client.connectionStatus}');
673       client.disconnect();
674       exit(-1);
675     }
676
677     client.updates!.listen((List<MqttReceivedMessage<MqttMessage?>>? c)
     {
678       final recMess = c![0].payload as MqttPublishMessage;
679       final pt = MqttPublishPayload.bytesToStringAsString(recMess.
     payload.message);
```

```
680       print('Change notification:: ---------------> topic is <${c[0].
      topic}>, payload is <-- $pt -->');
681       _body = "--> ${pt}";
682       print('');
683     });
684     client.published!.listen((MqttPublishMessage message) {
685       print('Published notification:: topic is ${message.variableHeader
      !.topicName}, with Qos ${message.header!.qos}');
686     });
687
688     const pubTopic = 'IU/9';
689     final builder = MqttClientPayloadBuilder();
690     builder.addString('Dart say ${message}');
691     _body = "--> ${message}";
692
693     print('Subscribing to the UI/9 topic');
694     client.subscribe(pubTopic, MqttQos.exactlyOnce);
695
696     print('Publishing our topic');
697     client.publishMessage(pubTopic, MqttQos.exactlyOnce, builder.payload
      !);
698
699     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.
700     await MqttUtilities.asyncSleep(60);
701     print('Awaked');
702     print('Unsubscribing....');
703     client.unsubscribe(pubTopic);
704
705
706     await MqttUtilities.asyncSleep(2); /// Wait for the unsubscribe
      message from the broker if you wish.
707     print('Disconnecting ...');
708     client.disconnect();
709     print('Stopped! Bye!....');
710
711   }
712
713   void onSubscribed(String topic) {
714     print('Subscription confirmed for topic $topic');
715   }
716
717   void onDisconnected() {
718     print('OnDisconnected client callback - Client disconnection');
719     if (client.connectionStatus!.disconnectionOrigin ==
```

```
720         MqttDisconnectionOrigin.solicited) {
721       print('OnDisconnected callback is solicited, this is correct');
722     } else {
723       print('OnDisconnected callback is unsolicited or none, this is
      incorrect - exiting');
724       exit(-1);
725     }
726     if (pongCount == 3) {
727       print('Pong count is correct');
728     } else {
729       print('Pong count is incorrect, expected 3. actual $pongCount');
730     }
731   }
732
733   void onConnected() {
734     print('OnConnected client callback - Client connection was
      successful');
735   }
736
737   void pong() {
738     print('Ping response client callback invoked');
739     _body = 'Ping response client callback invoked';
740     pongCount++;
741   }
742
743   @override
744   Widget build(BuildContext context) {
745     return Scaffold(
746         appBar: AppBar(
747           title: Text('Lab3'),
748         ),
749         body:      Padding(
750         padding: EdgeInsets.all(10.0),
751         child: new Form(
752             key: _formKey,
753             child: new Column(
754               children: <Widget>[
755       new Text('                          :', style: TextStyle(fontSize:
      20.0),),
756       new TextFormField(validator: (value) {
757         if (value == null || value.isEmpty)
758         {
759           return '                          -                          !';
760         }
761         else
762         {
```

```
763             print('---->'+value);
764             _body = value;
765
766
767             AAA(value);
768
769           }
770         }),
771
772         new SizedBox(height: 20.0),
773
774         ElevatedButton(
775           child: Text('Button'),
776           onPressed: () {
777
778             if(_formKey.currentState!.validate()) ScaffoldMessenger.of(
    context).showSnackBar(SnackBar(content: Text('
                          !'+_body), backgroundColor: Colors.red,));
779
780           },
781           style: ElevatedButton.styleFrom(
782               backgroundColor: Colors.purple,
783               padding: EdgeInsets.symmetric(horizontal: 50, vertical: 20),
784               textStyle: TextStyle(
785                   fontSize: 30,
786                   fontWeight: FontWeight.bold)),
787         ),
788
789       ],))),);
790   }
791 }
```

4 Результаты

Результаты представлен на рисунке 1.

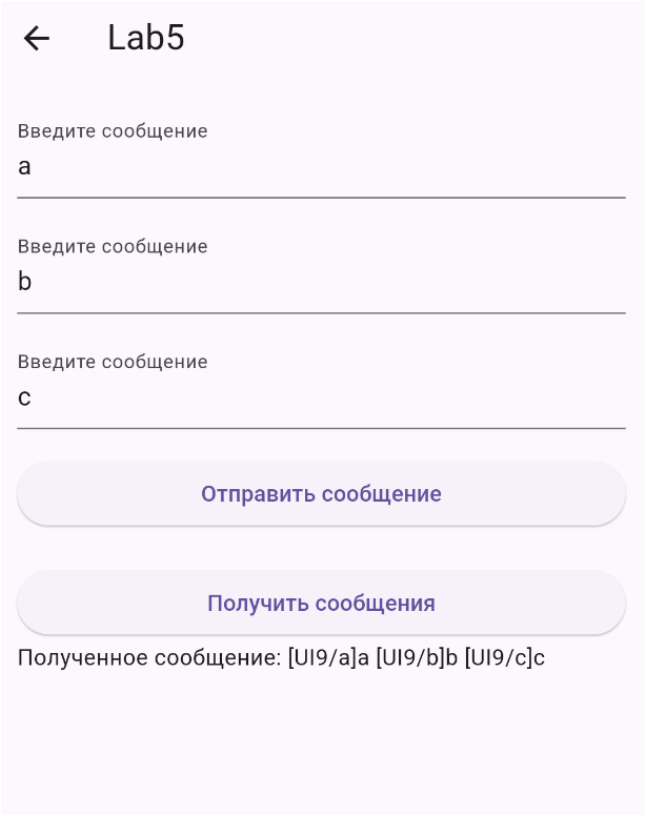

Рис. 1 — Интерфейс взаимодействия

5 Выводы

В результате данной лабороторной работы были изучены методологии работы с MQTT в языке Dart и получено практическиое применение освоенного материала. Реализовано приложение, поддерживающее работу с несколькими топиками.