



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

Отчет по практическим работам №9–12

по дисциплине «Технологические основы Интернета вещей»

Выполнили:

Студенты группы ИКБО-20-19

Городнов С.А.

Ильин А.Ю.

Московка А.А.

Николаев-Аксёнов И.С.

Семенов Д.Ю.

Марданян А.С.

Проверил:

Евтихов В. Г.

2021 г.

Оглавление

Практическая работа №9	3
Практическая работа №10	6
Практическая работа №11	16
Практическая работа №12	24
Дополнительное задание к ПР №9	31
Дополнительное задание к ПР №10	34
ЗАКЛЮЧЕНИЕ	36

Практическая работа №9

Создание виртуальных устройств в облаке и отправка данных в облако на устройства по MQTT согласно индивидуальному варианту.

В облако добавлены 3 устройства:

- Датчик движения (Motion_sensor)
- Датчик температуры (Temperature_sensor)
- Датчик напряжения (Voltage_sensor)

Для каждого датчика были добавлены индивидуальные профили.

Результаты выполнения работы отображены на рисунках 1 – 5.

<input type="checkbox"/>	Created time ↓	Name	Device profile
<input type="checkbox"/>	2021-11-13 12:25:14	Voltage_sensor	Voltage_sensor
<input type="checkbox"/>	2021-11-13 12:22:00	Temperature_sensor	Temperature_sensor
<input type="checkbox"/>	2021-11-13 12:08:53	Motion_sensor	Motion_sensor

Рисунок 1 – Список добавленных датчиков

```

Командная строка

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "BDz5WQdzCGBRb
ANjz7Uk" -m '{"motion": 203}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (13 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "onTVcPe11EVs3
twSbcWR" -m '{"temperature": 23.45}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (20 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "fmoLx6E3Yq3Hv
Y052tCh" -m '{"voltage": 3.3}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (14 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT

C:\Program Files\Mosquitto>

```

Рисунок 2 – Отправка данных в облако с помощью утилиты mosquitto_pub

The screenshot shows the ThingsBoard web interface. On the left is a sidebar with navigation options: Home, Rule chains, Customers, Assets, Devices (selected), Device profiles, OTA updates, Entity Views, Edge Instances, Edge management, Widgets Library, Dashboards, Audit Logs, Api Usage, and System Settings. The main area is divided into two panels. The left panel, titled 'Devices', shows a table of device profiles. The right panel, titled 'Motion_sensor', shows the details of a specific device, including tabs for Details, Attributes, Latest telemetry, Alarms, Events, Relations, and Audit Logs. The 'Latest telemetry' tab is active, showing a table with the latest data point.

Created time	Name	Device profile
2021-11-13 12:22:00	Temperature_sensor	Temperature
2021-11-13 12:08:53	Motion_sensor	Motion
2021-11-13 11:58:30	Charging Port 2	Charging
2021-11-13 11:58:29	Charging Port 1	Charging
2021-11-13 11:58:29	Air Quality Sensor T1	Air Quality
2021-11-13 11:58:29	Air Quality Sensor C1	Air Quality
2021-11-13 11:58:29	Sensor C1	Temperature
2021-11-13 11:58:29	Sensor T1	Temperature
2021-11-13 11:58:21	Test Device C1	default

Latest telemetry		
Last update time	Key	Value
2021-11-13 12:20:44	motion	203

Рисунок 3 – Поступившие данные датчика движения

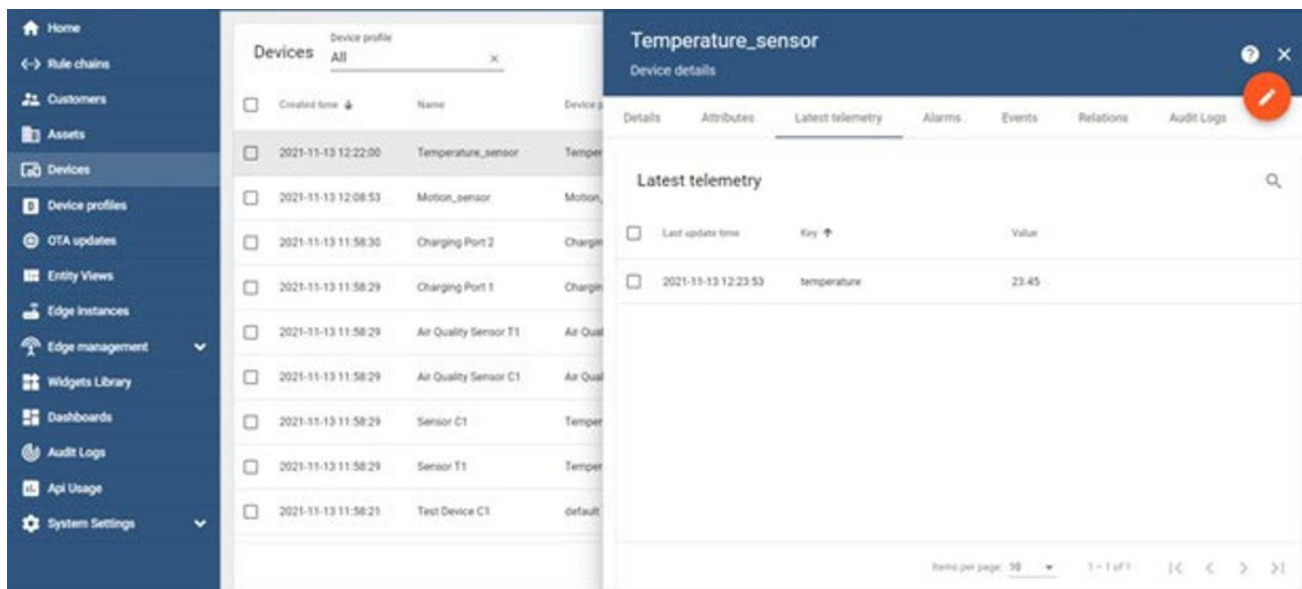


Рисунок 4 – Поступившие данные датчика температуры

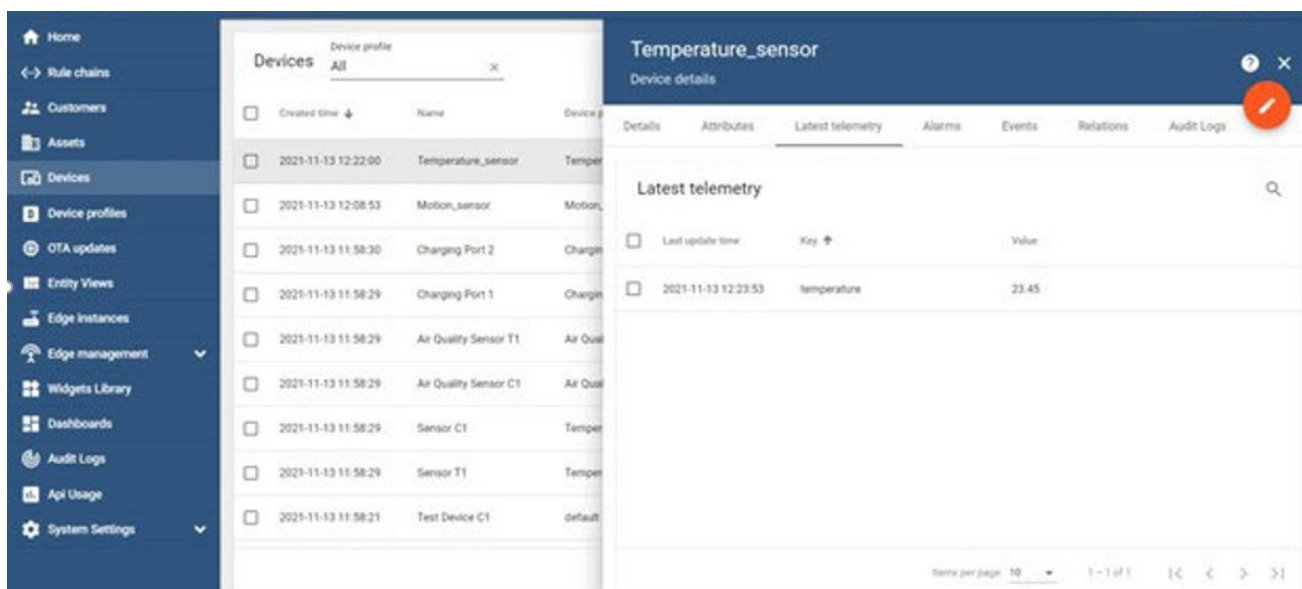


Рисунок 5 – Поступившие данные датчика напряжения

Практическая работа №10

Требуется реализовать скрипты по индивидуальному варианту из практической работы №3 при помощи цепочек правил ThingsBoard.

Сценарии для индивидуального варианта:

1. Включение и выключение вентилятора по температуре
2. Открытие и закрытие шарового крана при одновременном нажатии двух кнопок

Реализация первого сценария:

Сначала добавим устройство “Fan”, выступающее в качестве вентилятора (рис. 6).

<input type="checkbox"/>	Created time ↓	Name	Device profile
<input type="checkbox"/>	2021-11-13 13:41:54	Fan	Fan

Рисунок 6 – Добавление вентилятора

Далее создадим Rule Chain “Fan_control” (рис. 7).

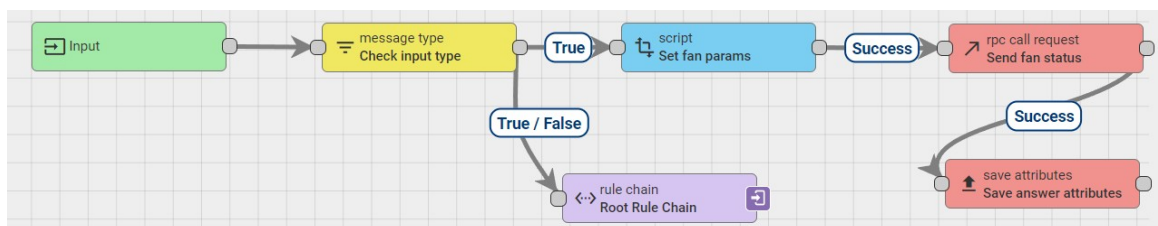


Рисунок 7 – Rule Chain “Fan_control”

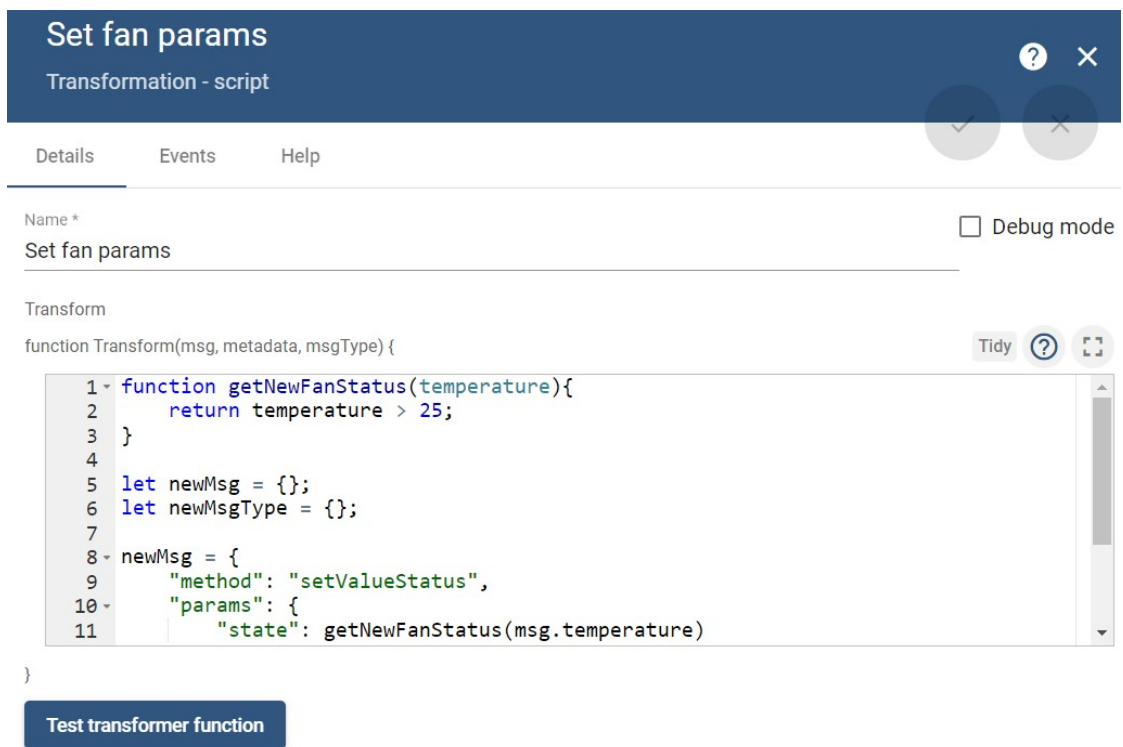


Рисунок 8 – Параметры цепочки правил

Листинг 1 – Код узла формирования параметра вентилятора

```
function getNewFanStatus(temperature){ return temperature > 25;  
}  
  
let newMsg = {}; let newMsgType = {};  
  
newMsg = {  
  "method": "setValueStatus", "params": {  
    "state": getNewFanStatus(msg.temperature)  
  }  
}  
  
newMsgType = "POST_ATTRIBUTES_REQUEST"  
  
return {msg: newMsg, metadata: metadata, msgType: newMsgType};
```

На рисунках 9 – 12 показаны тестирование и результаты работы Rule Chain при значении температуры больше 25.

```
Командная строка

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "h0BcdhaZ1czSxMtkySrQ" -m '{"temperature": 31}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (17 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/rpc/response/5" -u "h0BcdhaZ1czSxMtkySrQ" -m '{"status": 1}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/rpc/response/5', ... (11 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT

C:\Program Files\Mosquitto>
```

Рисунок 9 – Отправка значения температуры и изменение статуса вентилятора

```
Командная строка - mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rpc/request/+" -...

C:\Program Files\Mosquitto>

C:\Program Files\Mosquitto>

C:\Program Files\Mosquitto>

C:\Program Files\Mosquitto>mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rpc/request/+" -u "h0BcdhaZ1czSxMtkySrQ"
v1/devices/me/rpc/request/5 {"method":"setValueStatus","params":{"state":true}}
```

Рисунок 10 – Получение сообщения в mosquitto_sub (5 запрос)

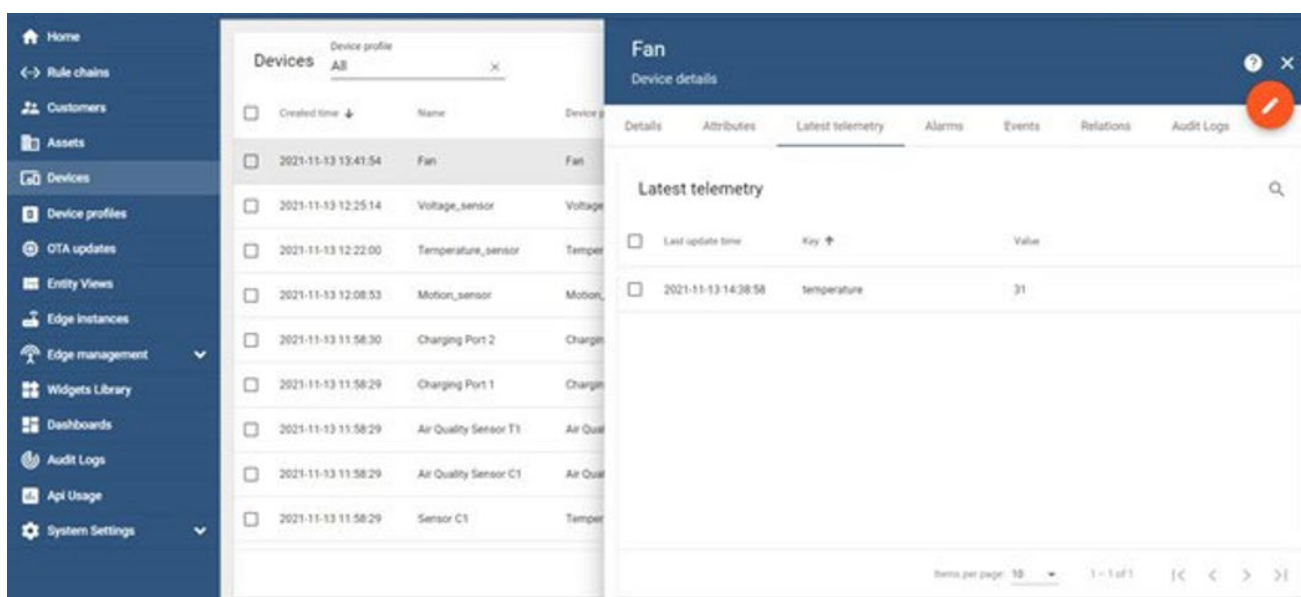


Рисунок 11 – Полученное значение температуры

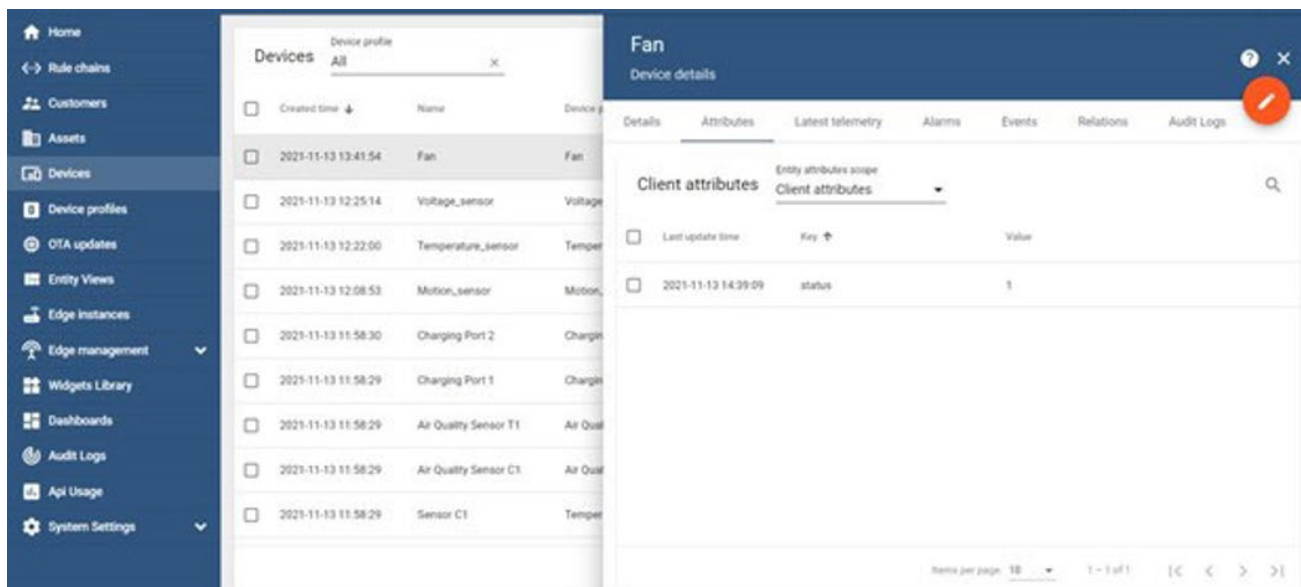


Рисунок 12 – Изменённый статус вентилятора

На рисунках 13 – 16 показаны тестирование и результаты работы Rule Chain’a при значении температуры больше 25.

```

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "h0BcdhaZ1czSxMtkySrQ" -m '{"temperature": 20}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (17 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/rpc/response/6" -u "h0BcdhaZ1czSxMtkySrQ" -m '{"status": 0}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/rpc/response/6', ... (11 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT

C:\Program Files\Mosquitto>

```

Рисунок 13 – Отправка значения температуры и изменение статуса вентилятора

```
cmd: Командная строка - mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rpc/request/+" -...
C:\Program Files\Mosquitto>
C:\Program Files\Mosquitto>
C:\Program Files\Mosquitto>mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rpc/request/+" -u "h0BcdhaZ1czSxMtkySrQ"
v1/devices/me/rpc/request/5 {"method":"setValueStatus","params":{"state":true}}
v1/devices/me/rpc/request/6 {"method":"setValueStatus","params":{"state":false}}
```

Рисунок 14 Получение сообщения в mosquitto_sub (6 запрос)

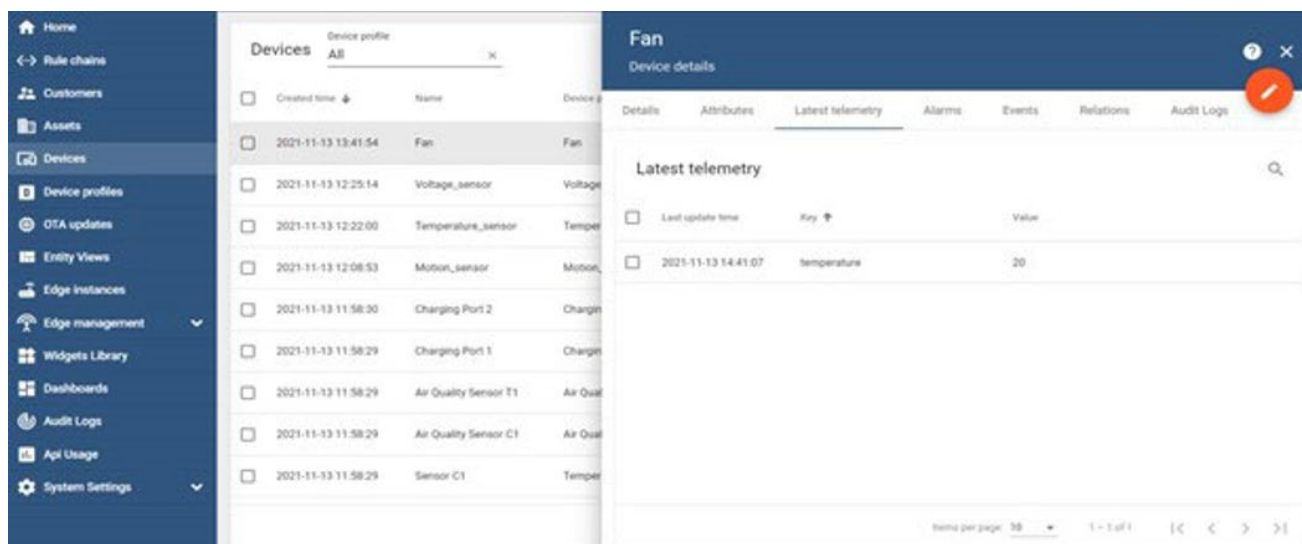


Рисунок 15 – Полученное значение температуры

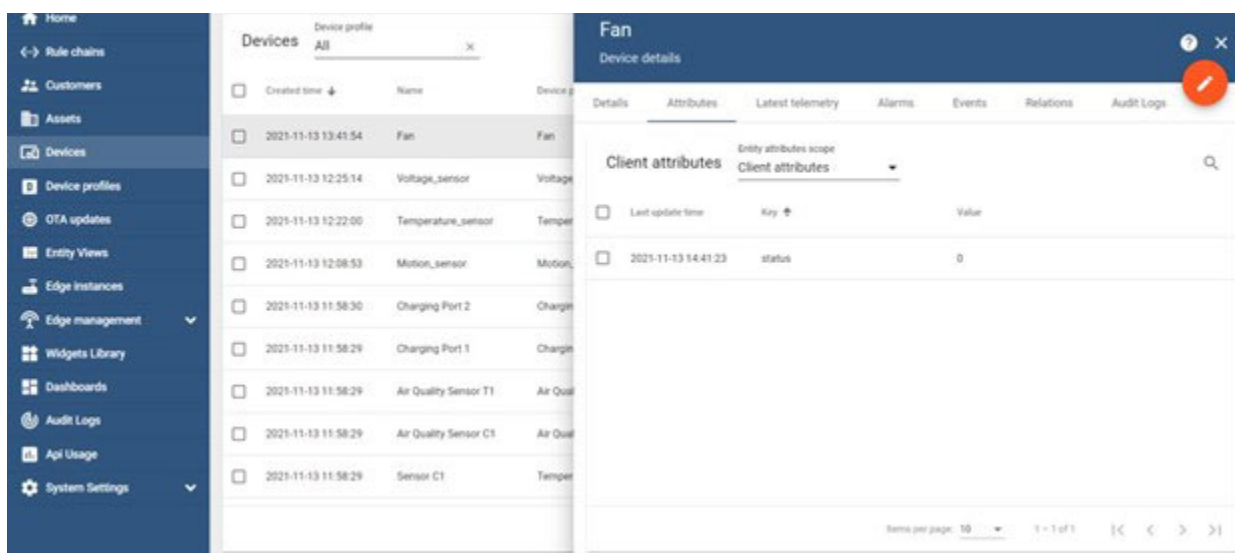


Рисунок 16 – Изменённый статус вентилятора

Реализация второго сценария:

Добавим устройство “Valve”, выступающее в качестве шарового крана (рис. 17).

<input type="checkbox"/>	Created time ↓	Name	Device profile
<input type="checkbox"/>	2021-11-13 15:16:24	Valve	Valve

Рисунок 17 – Добавление шарового крана

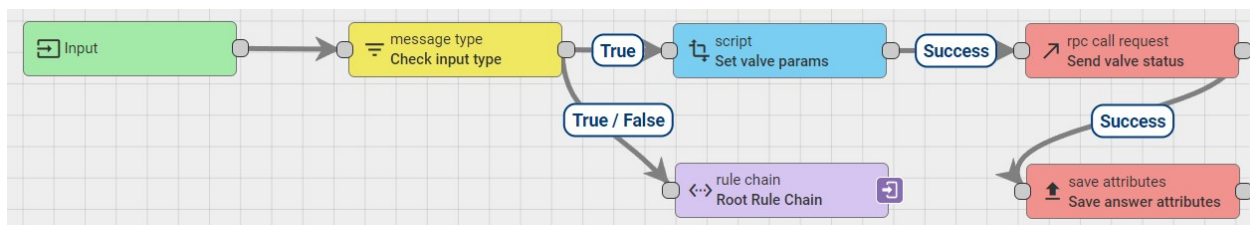


Рисунок 18 – Rule Chain “Valve_control”

Set valve params

Transformation - script

Details Events Help

Name * ☐ Debug mode

Set valve params

Transform

```
function Transform(msg, metadata, msgType) {  
  1 function getNewValveStatus(button1, button2){  
  2   return (button1 && button2) || (!button1 && !button2);  
  3 }  
  4  
  5 let newMsg = {};  
  6 let newMsgType = {};  
  7  
  8 newMsg = {  
  9   "method": "setValueStatus",  
10   "params": {  
11     "state": getNewValveStatus(msg.button1, msg.button2)  
  }  
}  
}
```

Tidy ?

Test transformer function

Рисунок 19 – Параметры цепочки правил

Листинг 2 – Код узла формирования параметра шарового крана

```
function getNewValveStatus(button1, button2){  
  return (typeof button1 == "boolean") && (typeof button2 == "boolean");  
}  
  
let newMsg = {}; let newMsgType = {};  
  
newMsg = {  
  "method": "setValueStatus", "params": {  
    "state": getNewValveStatus(msg.button1, msg.button2)  
  }  
}  
  
newMsgType = "POST_ATTRIBUTES_REQUEST"  
  
return {msg: newMsg, metadata: metadata, msgType: newMsgType};
```

На рисунках 20 – 23 показаны тестирование и результаты работы Rule Chain'а при нажатии двух кнопок (изменение статуса крана на 1).

```
Командная строка
Client (null) sending DISCONNECT

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "GLG4fcKpAH409QL06NF9" -m '{"button1": true, "button2": true}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (30 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/rpc/response/24" -u "GLG4fcKpAH409QL06NF9" -m '{"status": 1}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/rpc/response/24', ... (11 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT
```

Рисунок 20 – Отправка значений кнопок и изменение статуса крана на 1

```
Командная строка - mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rpc/request/+" -...
C:\Program Files\Mosquitto>
C:\Program Files\Mosquitto>
C:\Program Files\Mosquitto>

C:\Program Files\Mosquitto>mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rpc/request/+" -u "GLG4fcKpAH409QL06NF9"
v1/devices/me/rpc/request/24 {"method":"setValueStatus","params":{"state":true}}
```

Рисунок 21 – Получение сообщения в mosquitto_sub (24 запрос)

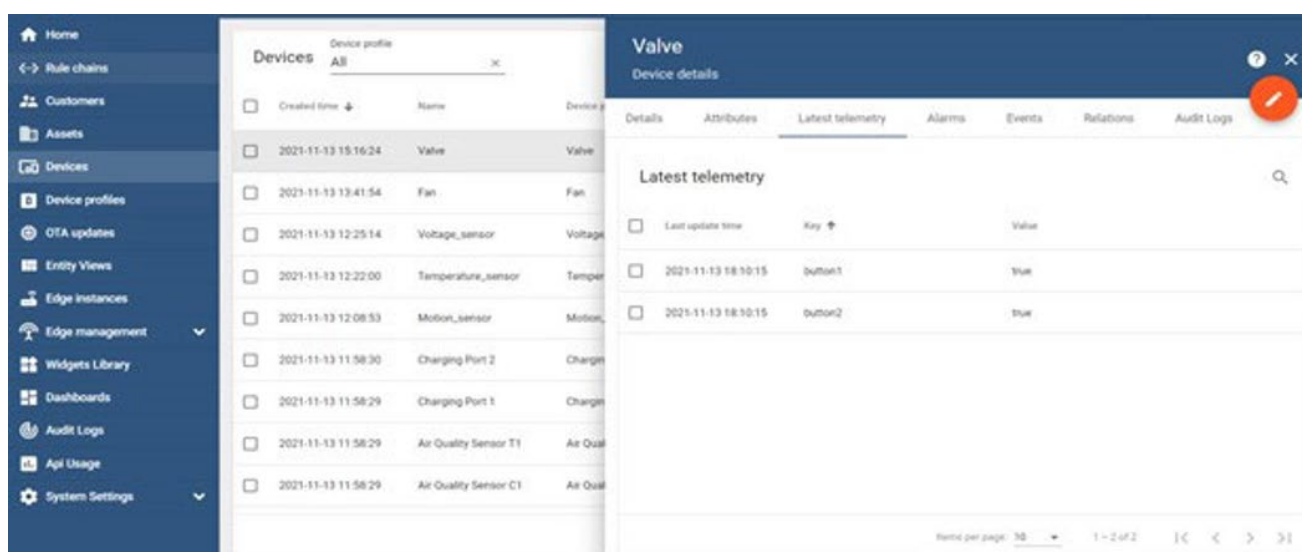


Рисунок 22 – Полученные значения кнопок

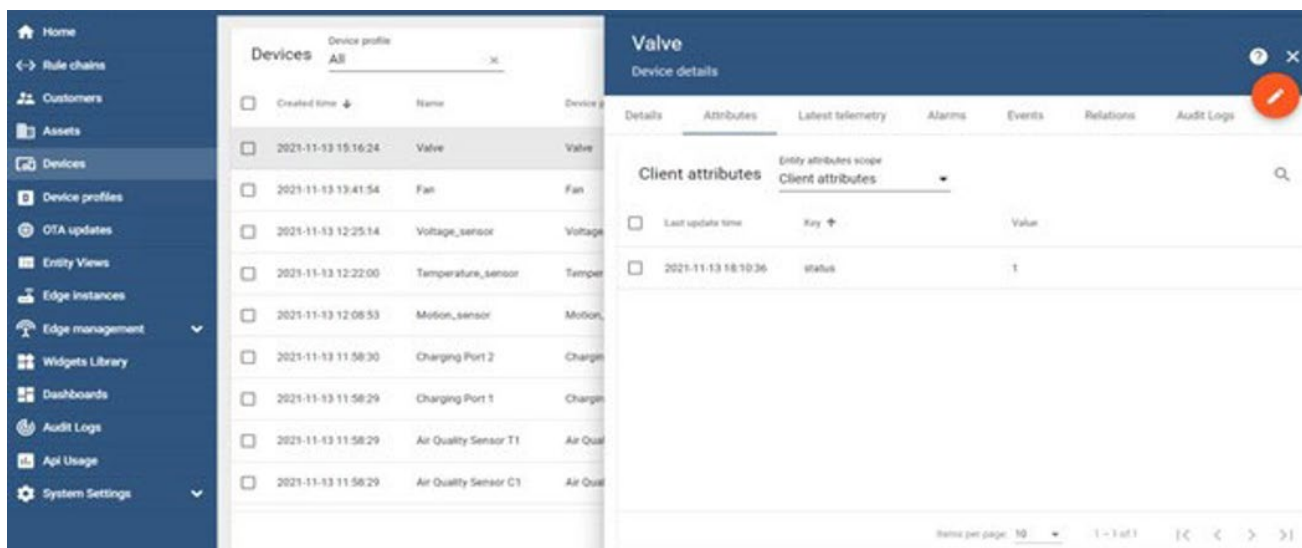


Рисунок 23 – Изменённый статус крана

На рисунках 24 – 27 показаны тестирование и результаты работы Rule Chain'a при нажатии двух кнопок (изменение статуса крана на 0).

```

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "GLG4fcKpAH409QL06NF9" -m '{"button1": false, "button2": false}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (32 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/rpc/response/25" -u "GLG4fcKpAH409QL06NF9" -m '{"status": 0}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/rpc/response/25', ... (11 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT

C:\Program Files\Mosquitto>

```

Рисунок 24 – Отправка значений кнопок и изменение статуса крана на 0

```
Командная строка - mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rpc/request/+" -...
C:\Program Files\Mosquitto>
C:\Program Files\Mosquitto>
C:\Program Files\Mosquitto>mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rpc/request/+" -u "GLG4fcKpAH409QL06NF9"
v1/devices/me/rpc/request/24 {"method":"setValueStatus","params":{"state":true}}
v1/devices/me/rpc/request/25 {"method":"setValueStatus","params":{"state":true}}
```

Рисунок 25 – Получение сообщения в mosquitto_sub (25 запрос)

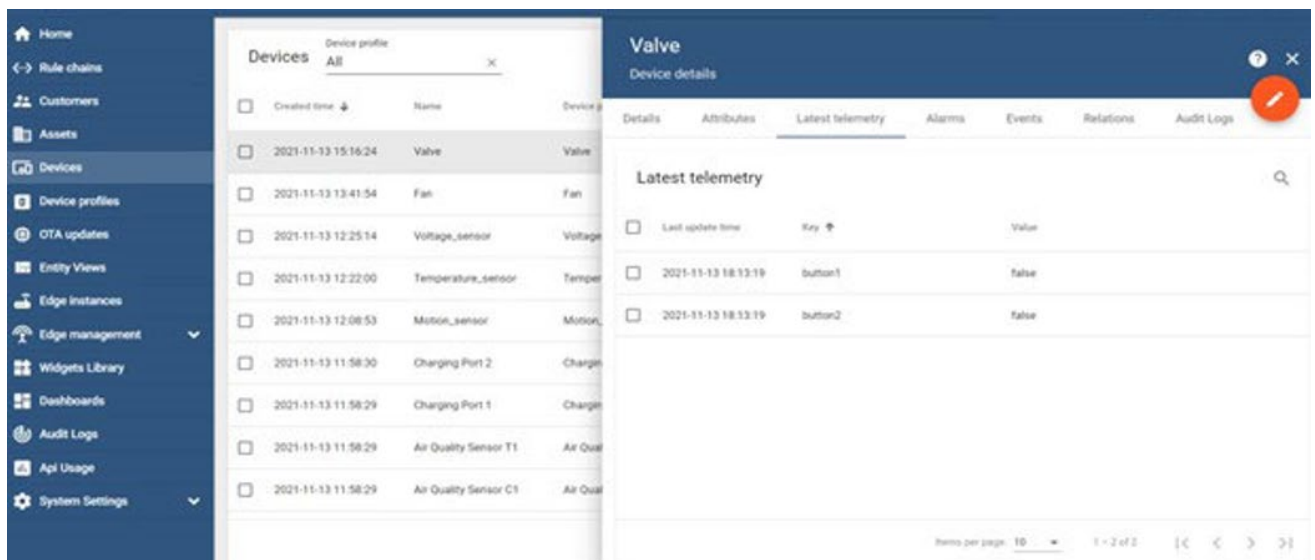


Рисунок 26 – Полученные значения кнопок

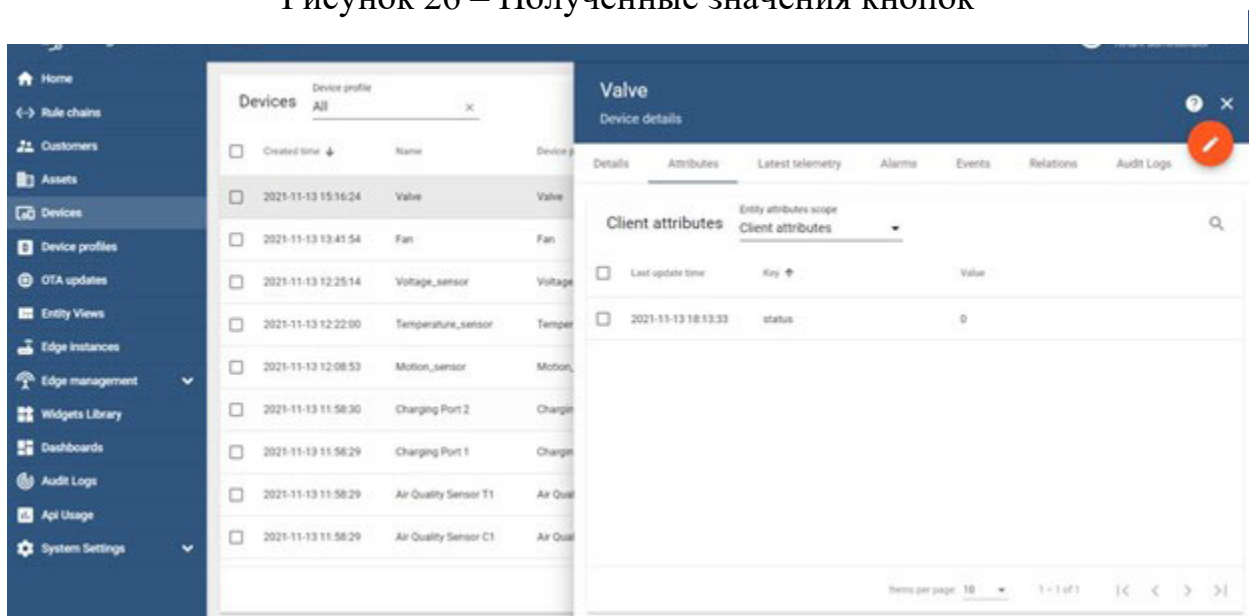


Рисунок 27 – Изменённый статус крана

Практическая работа №11

Часть 1, цепочка правил для вентилятора:

В цепочку правил была добавлена проверка поступающих значений на их соответствие определённому диапазону (от -30 до 50), а также формирование тревоги при несоответствии этому диапазону. Скрипт проверки представлен в листинге 3, скрипт формирования тревоги – в листинге 4.

Также была добавлена проверка правильности ответа и формирование тревоги, если ответ неправильный. Их скрипты представлены в листингах 5 и 6.

Листинг 3 – Проверка температуры на соответствие диапазону

```
return (msg.temperature >= -30) && (msg.temperature <= 50);
```

Листинг 4 – Формирование тревоги при несоответствии диапазону

```
var details = {};  
if (metadata.prevAlarmDetails) {  
  details = JSON.parse(metadata.prevAlarmDetails);  
  //remove prevAlarmDetails from metadata delete metadata.prevAlarmDetails;  
  //now metadata is the same as it comes IN this rule node  
}  
return details;
```

Листинг 5 – Проверка правильности ответа

```
let request_params = JSON.parse(metadata.ss_params);  
return msg.status === request_params.status;
```

Листинг 6 – Формирование тревоги при неправильном ответе

```
var details = {};  
var request_params = JSON.parse(metadata.ss_params); if (metadata.prevAlarmDetails)  
{  
  details = JSON.parse(metadata.prevAlarmDetails);  
  // Удаление поля prevAlarmDetails из метаданных  
  delete metadata.prevAlarmDetails;  
  // Теперь метаданные содержат только данные, которые были на входе  
}  
details.send_status = request_params.status; details.answer_status = msg.status;  
return details;
```


Итоговый вид цепочки правил представлен на рисунках 28 и 29.

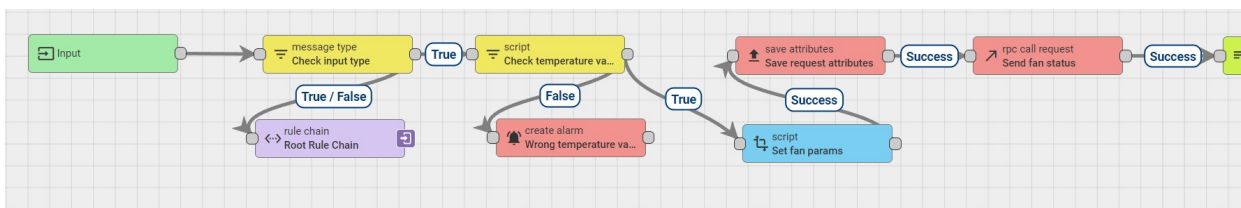


Рисунок 28 – Цепочка правил вентилятора (часть 1)

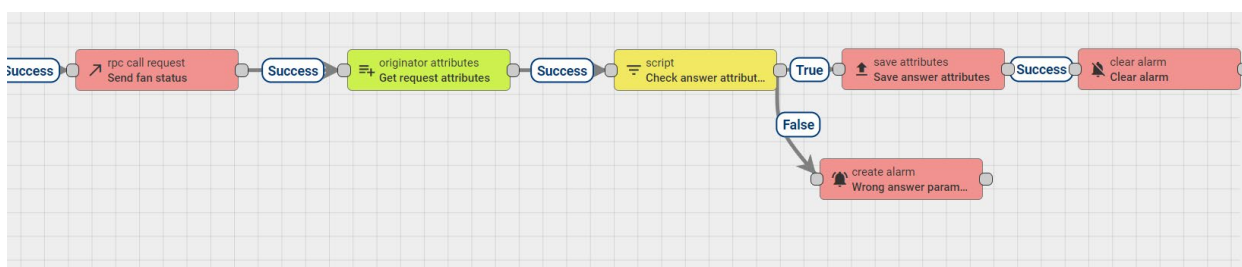


Рисунок 29 – Цепочка правил вентилятора (часть 2)

Протестируем работу созданной цепочки.

Изначально активных тревог нет. Отправим на устройство данные, находящиеся вне указанного выше диапазона, а затем отправим данные, попадающие в диапазон, но вернём неверный ответ на запрос. В итоге, во вкладке тревог устройства у должны появиться активные тревоги типа «Critical temperature» и «Wrong answer alarm».

Весь процесс тестирования цепочки изображён на рисунках 30 – 36.

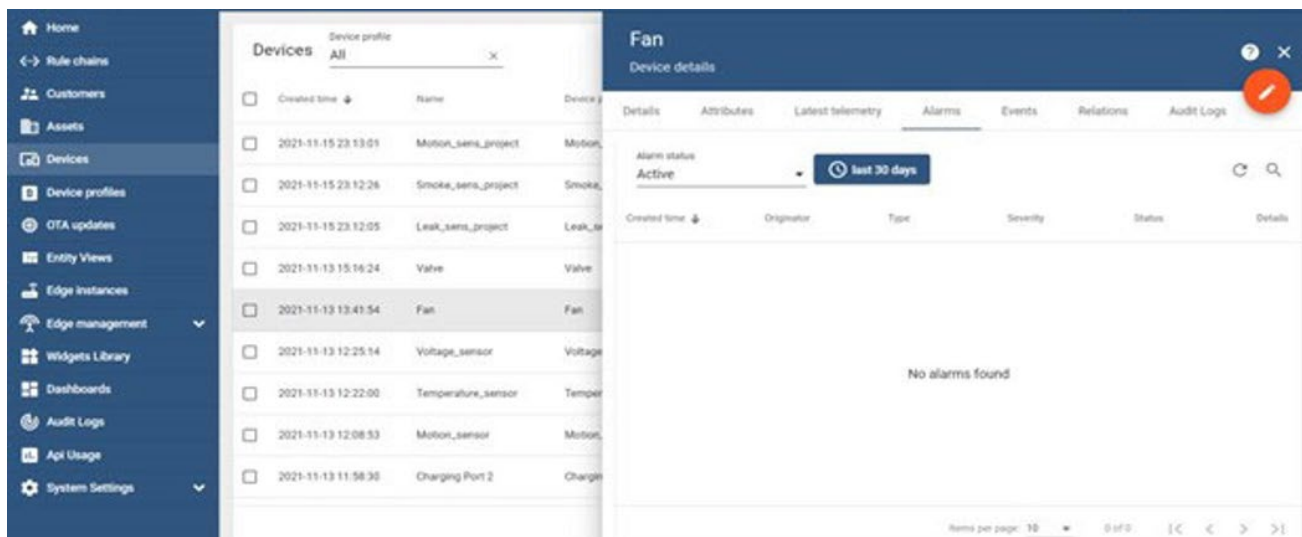


Рисунок 30 – Вкладка тревог перед тестированием

```

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "h0BcdhaZ1czSxMtkySrQ" -m '{"temperature": 90}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (17 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT
C:\Program Files\Mosquitto>

```

Рисунок 31 – Отправка данных температуры вне диапазона

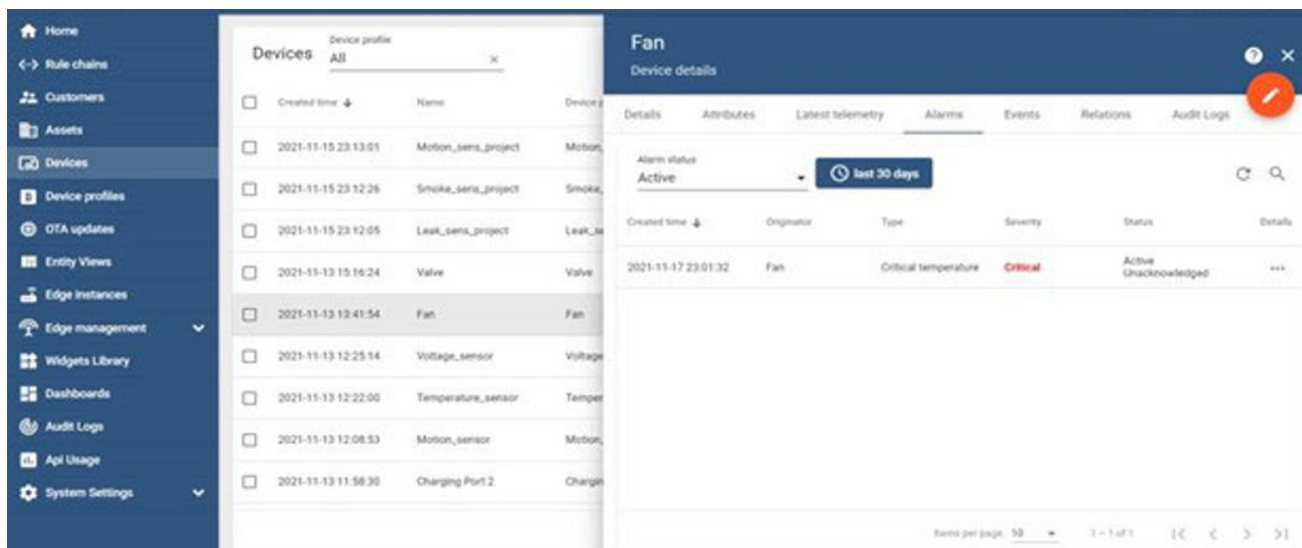


Рисунок 32 – Появление активной тревоги по температуре

```
Командная строка

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "h0BcdhaZ1czSxMtkySrQ" -m
{"temperature": 30}
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (17 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT
```

Рисунок 33 – Отправка температуры в указанном диапазоне

```
Командная строка - mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rpc/re...

C:\Program Files\Mosquitto>mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rp
c/request/+" -u "h0BcdhaZ1czSxMtkySrQ"
v1/devices/me/rpc/request/25 {"method":"setValueStatus","params":{"status":true}}
```

Рисунок 34 – Полученный запрос

```
Командная строка

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/rpc/response/25" -u "h0BcdhaZ1czSxMtkySr
Q" -m "{\"status\": false}"
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/rpc/response/25', ... (17 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT
```

Рисунок 35 – Отправка неверного ответа

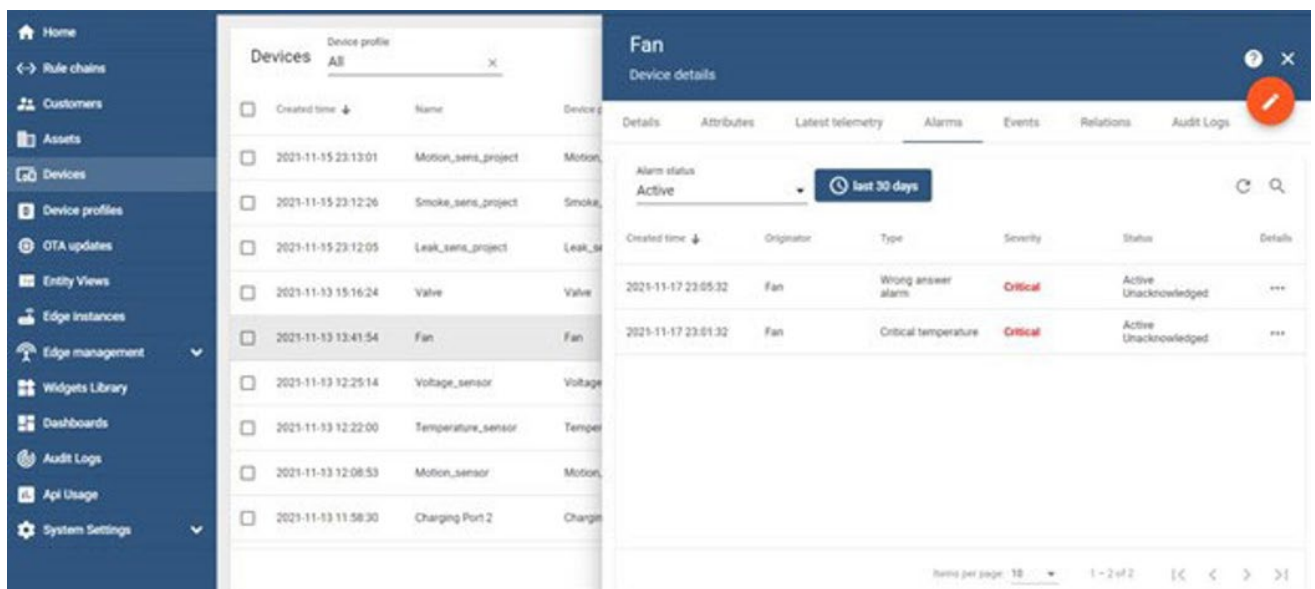


Рисунок 36 – Появление активной тревоги по неверному ответу

Часть 2, цепочка правил для шарового крана:

В цепочку правил была добавлена проверка на отсутствие состояния кнопок в приходящем сообщении, а также формирование тревоги при соответствии этому условию. Скрипт проверки представлен в листинге 7, скрипт формирования тревоги – в листинге 8.

Также была добавлена проверка правильности ответа и формирование тревоги, если ответ неправильный. Их скрипты представлены в листингах 9 и 10.

Листинг 7 – Проверка на отсутствие состояний кнопок

```
return (msg.hasOwnProperty("button1")) && (msg.hasOwnProperty("button2"));
```

Листинг 8 – Формирование тревоги при отсутствии состояний кнопок

```
var details = {};  
if (metadata.prevAlarmDetails) {  
  details = JSON.parse(metadata.prevAlarmDetails);  
  //remove prevAlarmDetails from metadata delete metadata.prevAlarmDetails;  
  //now metadata is the same as it comes IN this rule node  
}  
return details;
```

Листинг 9 – Проверка правильности ответа

```
let request_params = JSON.parse(metadata.ss_params); return msg.status ===  
request_params.status;
```

Листинг 10 – Формирование тревоги, если ответ неправильный

```
var details = {};  
var request_params = JSON.parse(metadata.ss_params); if (metadata.prevAlarmDetails)  
{  
  details = JSON.parse(metadata.prevAlarmDetails);  
  // Удаление поля prevAlarmDetails из метаданных  
  delete metadata.prevAlarmDetails;  
  // Теперь метаданные содержат только данные, которые были на входе  
}  
details.send_status = request_params.status; details.answer_status = msg.status;  
return details;
```

Итоговый вид цепочки правил представлен на рисунках 37 и 38.

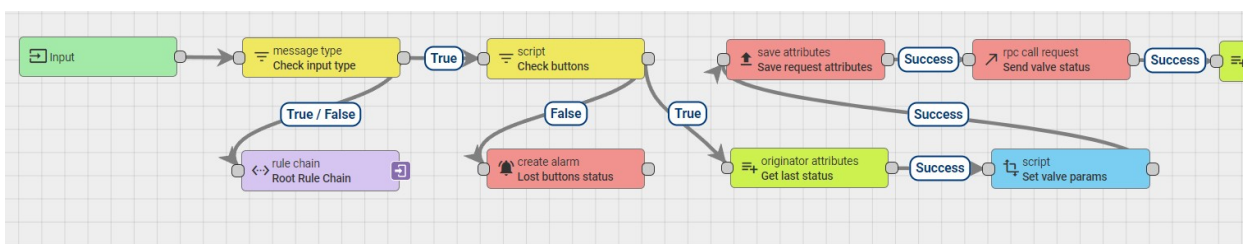


Рисунок 37 – Цепочка правил для крана (часть 1)

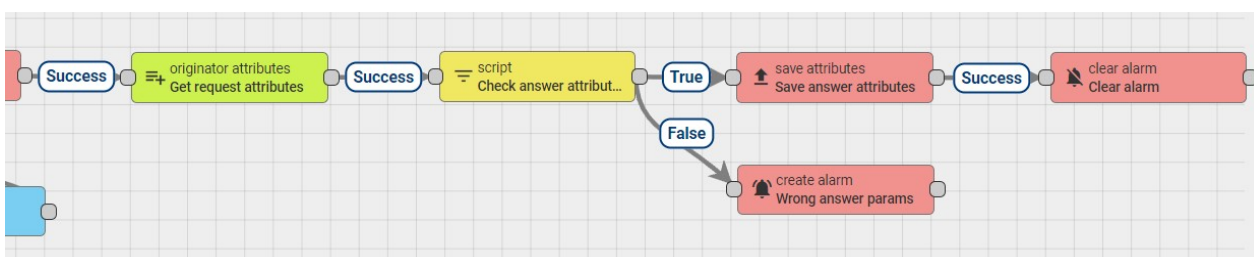


Рисунок 38 – Цепочка правил для крана (часть 2)

Протестируем работу созданной цепочки.

Изначально активных тревог нет. Отправим на устройство данные, которые не содержат состояния кнопок, затем отправим данные, содержащие состояния обеих кнопок, но вернём неверный ответ на запрос. В итоге, во вкладке тревог устройства у должны появиться активные тревоги типа «Lost buttons status» и «Wrong answer params».

Весь процесс тестирования цепочки изображён на рисунках 39 – 45.

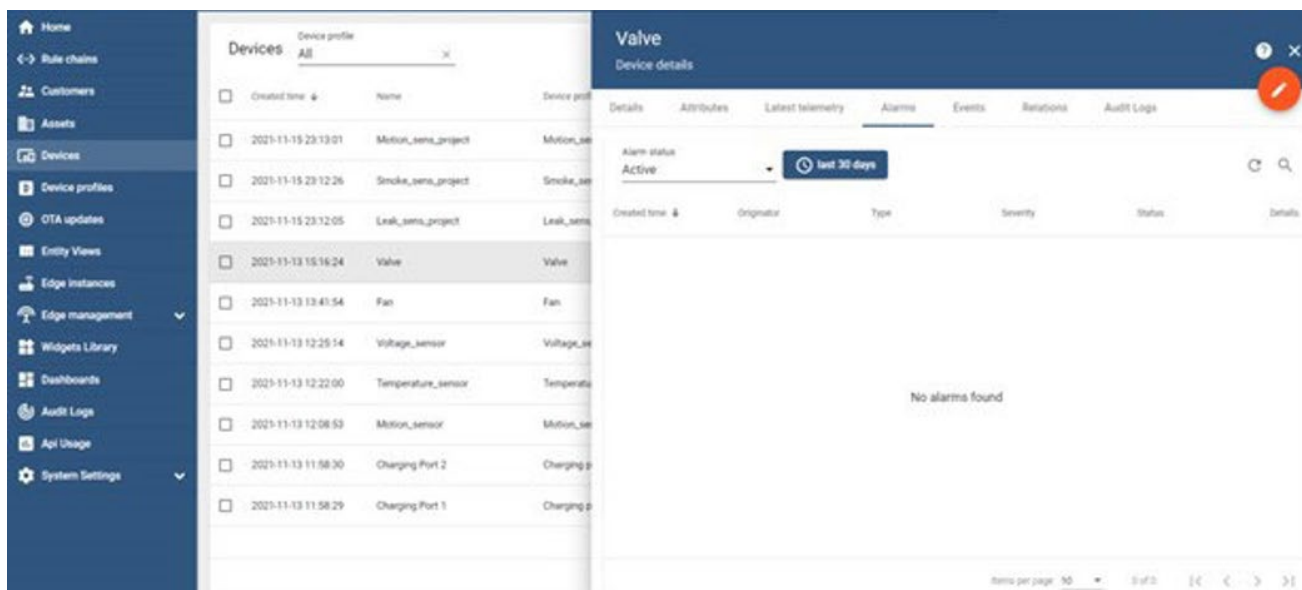


Рисунок 39 – Вкладка тревог перед тестированием



Рисунок 40 – Отправка сообщения без состояния кнопок

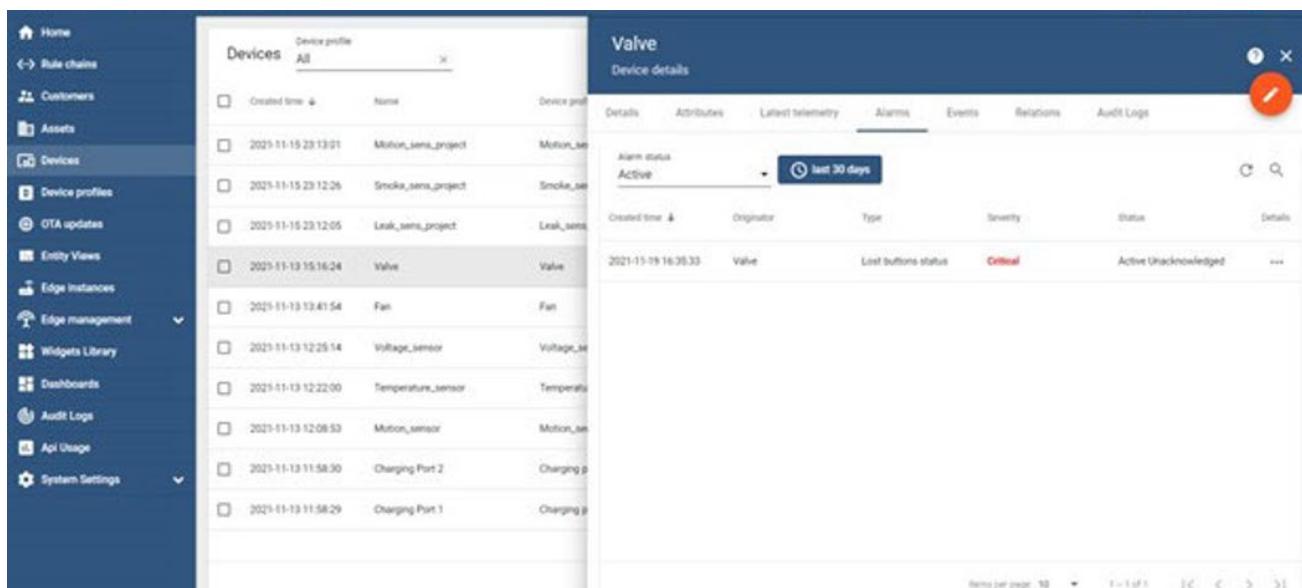


Рисунок 41 – Появление активной тревоги из-за отсутствия состояний кнопок


```
Командная строка

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "GLG4fcKpAH409QL06NF9" -m '{"button1": true, "button2": true}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (30 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT
```

Рисунок 42 – Отправка сообщения с состоянием кнопок

```
Командная строка - mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/...

v1/devices/me/rpc/request/86 {"method":"setValueStatus","params":{"status":false,
"button1":true,"button2":true}}
```

Рисунок 43 – Полученный запрос

```
Командная строка

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/rpc/response/86" -u "GLG4fcKpAH409QL06NF9" -m '{"status": true}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/rpc/response/86', ... (16 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT

C:\Program Files\Mosquitto>
```

Рисунок 44 – Отправка неверного ответа на запрос

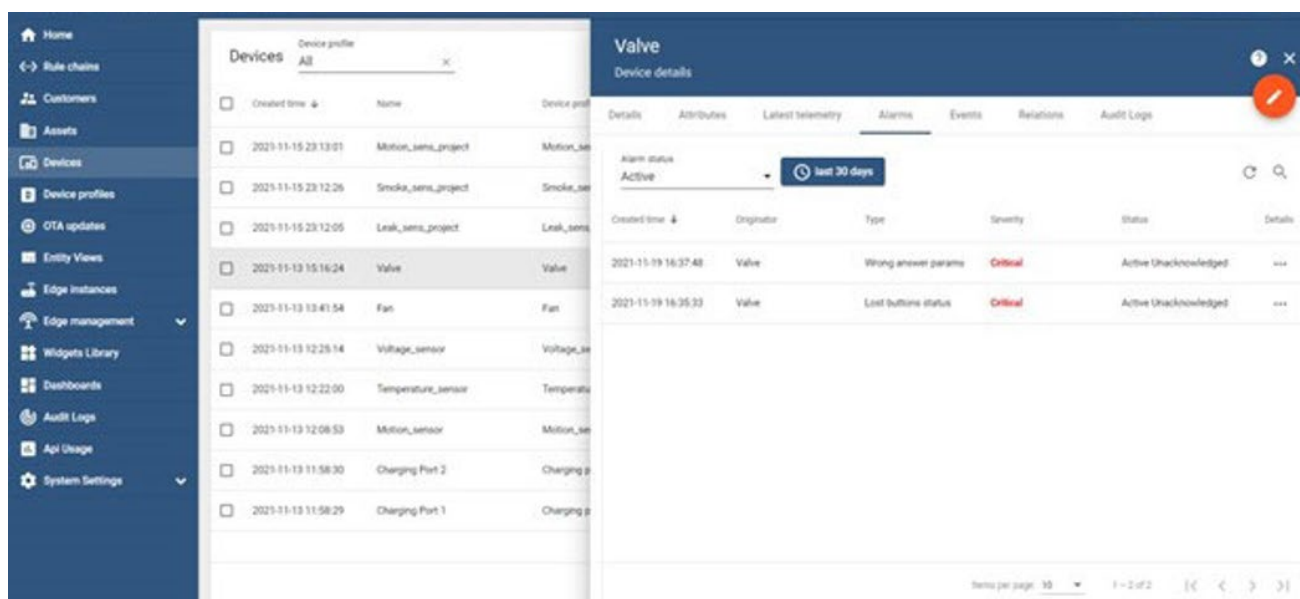


Рисунок 45 – Появление активной тревоги по неверному ответу

Практическая работа №12

Часть 1, вентилятор:

Обновлённая цепочка правил представлена на рисунках 46 и 47.

Параметры узлов пересылки сообщений представлены на рисунках 48 и 49.

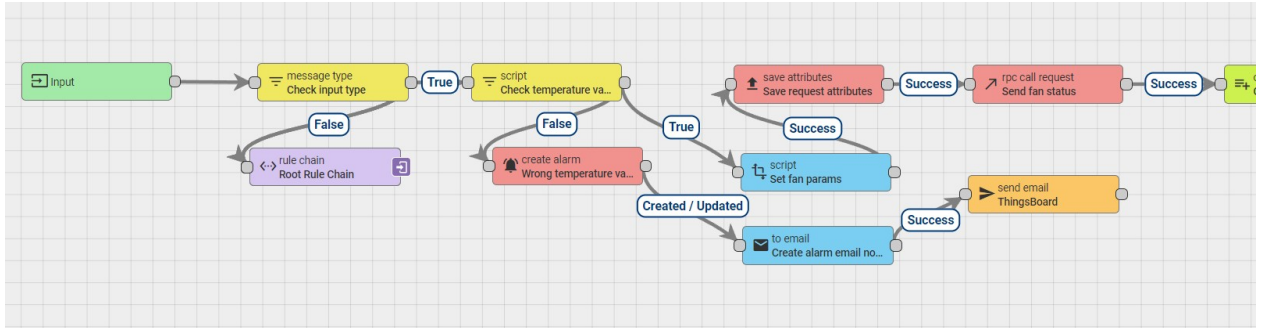


Рисунок 46 – Обновлённая цепочка правил (часть 1)

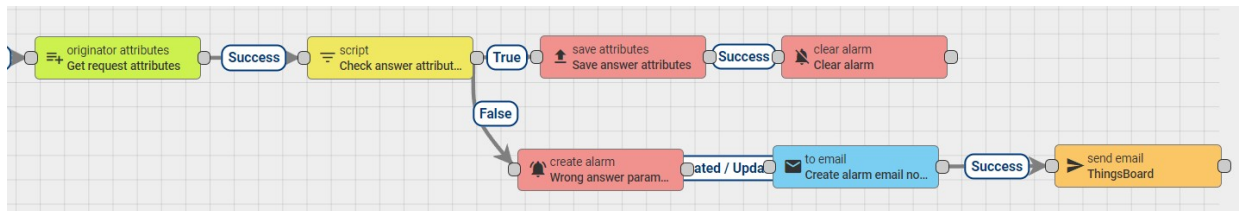


Рисунок 47 – Обновлённая цепочка правил (часть 2)

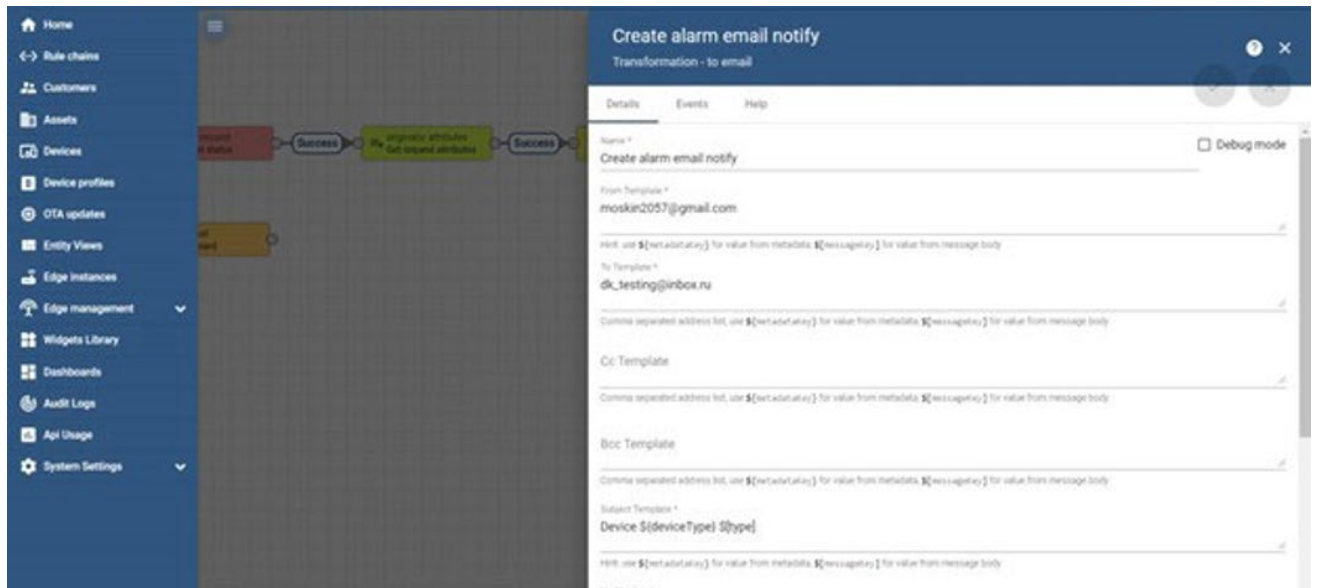


Рисунок 48 – Узел создания сообщения

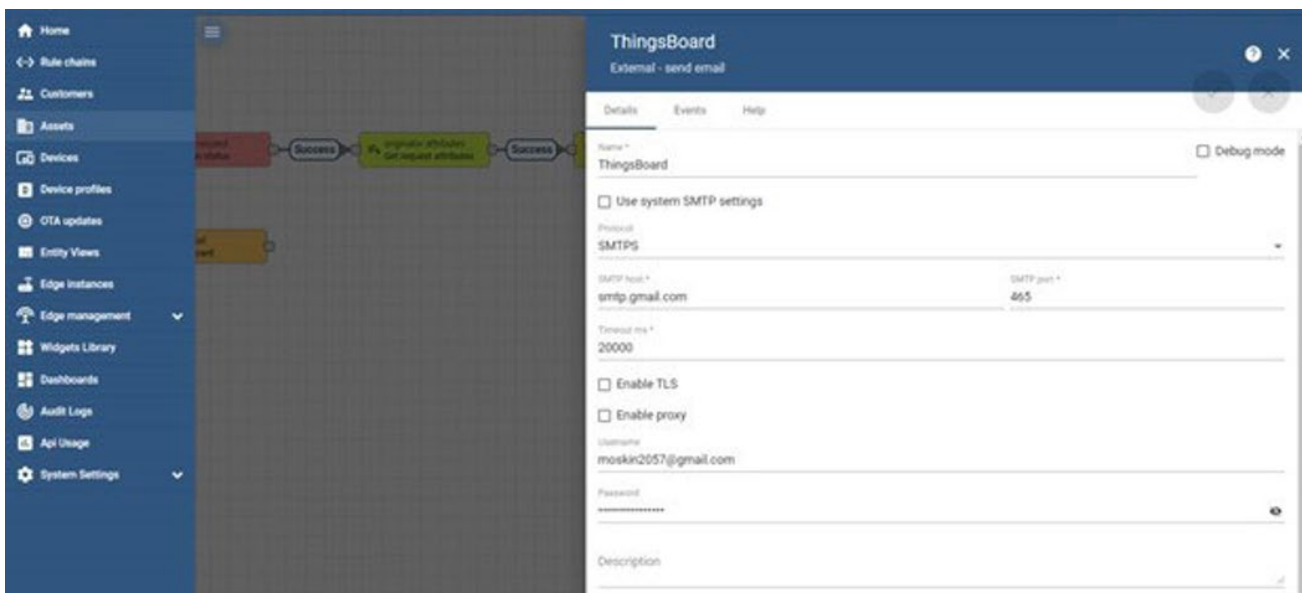


Рисунок 49 – Узел отправки сообщения

Протестируем обновлённую цепочку правил.

Изначально активных тревог нет. Отправим на устройство данные, находящиеся вне указанного выше диапазона, а затем отправим данные, попадающие в диапазон, но вернём неверный ответ на запрос. В итоге, во вкладке тревог устройства у должны появиться активные тревоги типа «Critical temperature» и «Wrong answer alarm», а также два письма на указанной выше электронной почте.

Весь процесс тестирования цепочки изображён на рисунках 50 – 56.

```

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "h0BcdhaZ1czSxMtkySrQ" -m '{"temperature": 91}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (17 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT
  
```

Рисунок 50 – Отправка температуры вне диапазона

```

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "h0BcdhaZ1czSxMtkySrQ" -m '{"temperature": 30}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (17 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT
  
```

Рисунок 51 – Отправка температуры в заданном диапазоне

```
Командная строка - mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rpc/re...  
C:\Program Files>cd mosquitto  
C:\Program Files\Mosquitto>mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rp  
c/request/+" -u "h0BcdhaZ1czSxMtkySrQ"  
v1/devices/me/rpc/request/27 {"method":"setValueStatus","params":{"status":true}}
```

Рисунок 52 – Получение запроса

```
Командная строка  
C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/rpc/response/27" -u "h0Bcdha  
Z1czSxMtkySrQ" -m "{\"status\": false}"  
Client (null) sending CONNECT  
Client (null) received CONNACK (0)  
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/rpc/response/27', ... (17 bytes))  
Client (null) received PUBACK (Mid: 1, RC:0)  
Client (null) sending DISCONNECT
```

Рисунок 53 – Неверный ответ на запрос

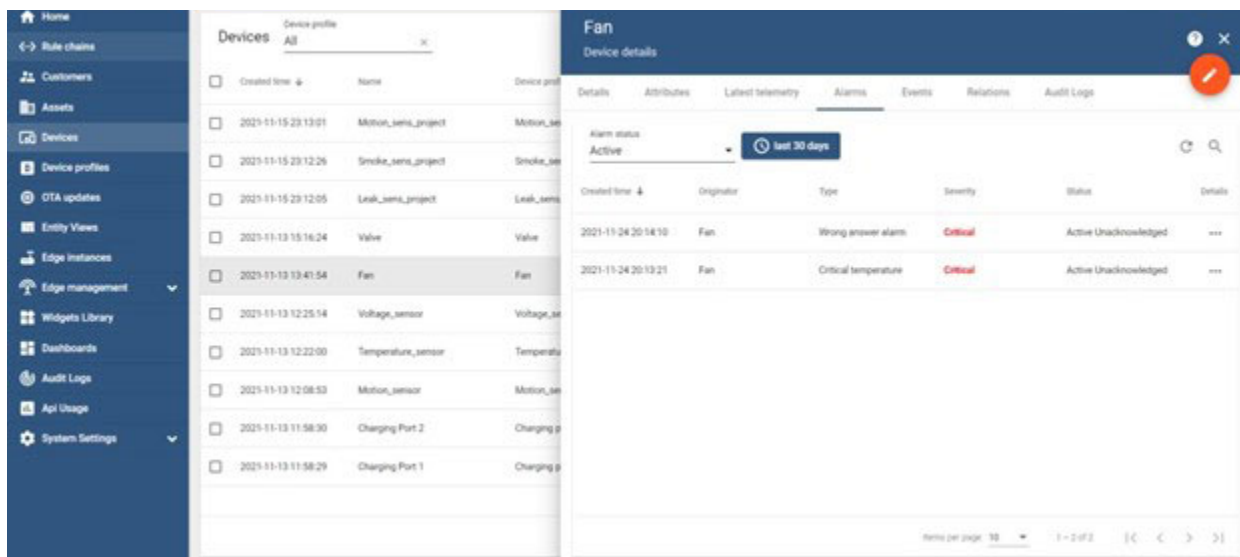


Рисунок 54 – Появление тревог

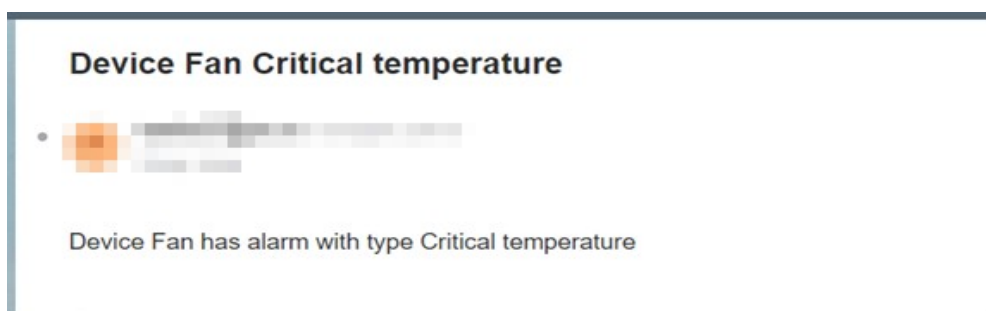


Рисунок 55 – Письмо о тревоге по температуре

Device Fan Wrong answer alarm



Device Fan has alarm with type Wrong answer alarm

Рисунок 56 – Письмо о тревоге из-за неверного ответа

Часть 2, шаровой кран:

Обновлённая цепочка правил представлена на рисунках 57 и 58.

Параметры узлов пересылки сообщений представлены на рисунках 59 и 60.

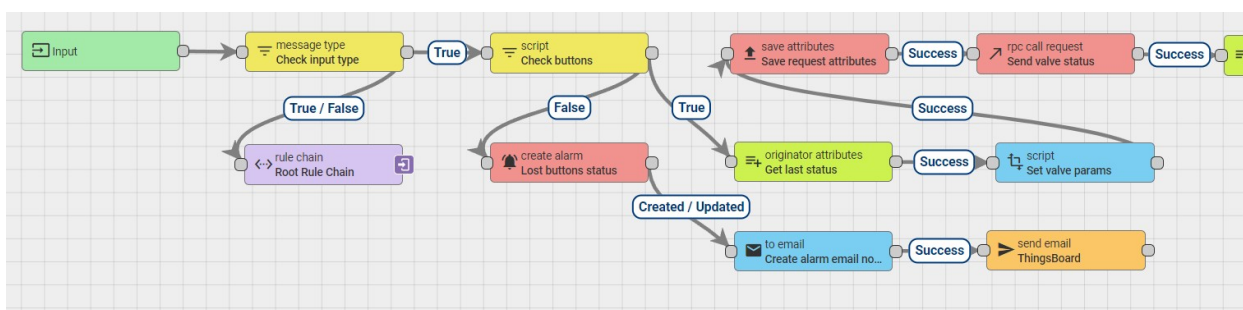


Рисунок 57 – Обновлённая цепочка правил (часть 1)

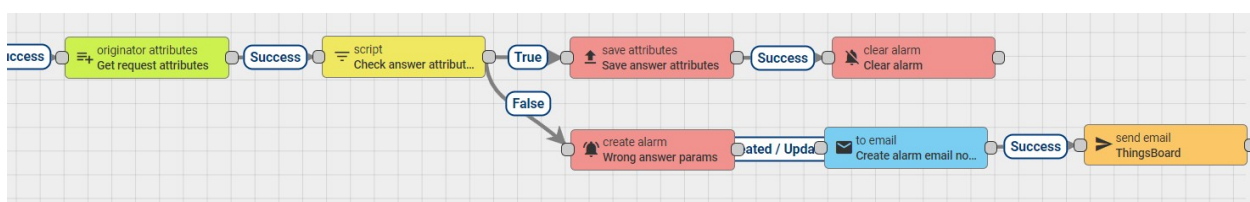


Рисунок 58 – Обновлённая цепочка правил (часть 2)

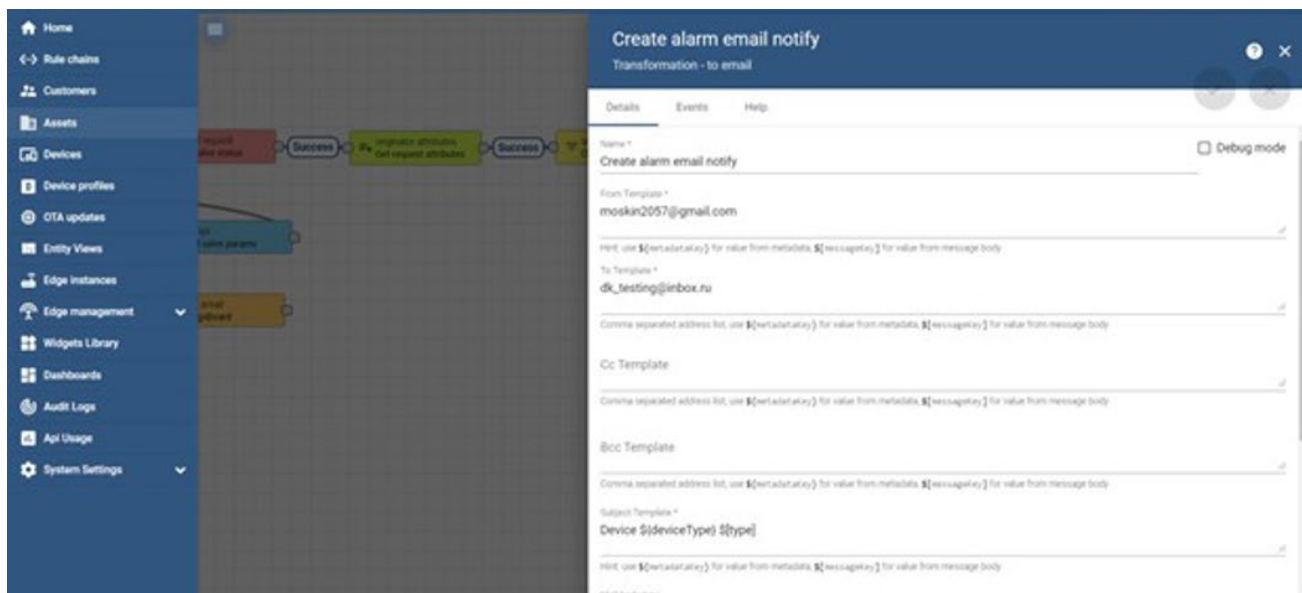
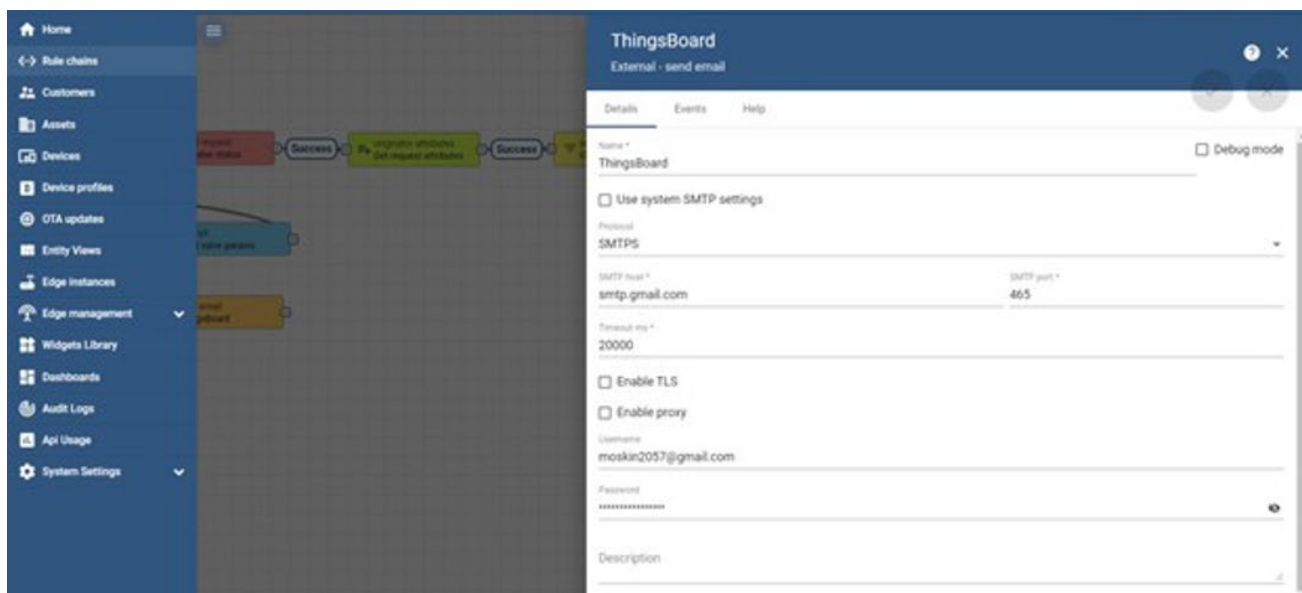


Рисунок 59 – Узел создания сообщения



Протестируем обновлённую цепочку правил.

Рисунок 60 – Узел отправки сообщения

Изначально активных тревог нет. Отправим на устройство данные, которые не содержат состояния кнопок, затем отправим данные, содержащие состояния обеих кнопок, но вернём неверный ответ на запрос. В итоге, во вкладке тревог устройства у додлжны появиться активные тревоги типа «Lost buttons status» и «Wrong answer params», а также два письма на указанной выше электронной почте.

Весь процесс тестирования цепочки изображён на рисунках 61 – 67.

```
Командная строка

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "GLG4fcKpAH409QL06NF9" -m '{"amg": true, "amh": true}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (22 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT
```

Рисунок 61 – Отправка сообщения без состояния кнопок

```
Командная строка

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "GLG4fcKpAH409QL06NF9" -m '{"button1": true, "button2": true}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (30 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT
```

Рисунок 62 – Отправка сообщения с состоянием кнопок

```
Командная строка - mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rpc/re...
v1/devices/me/rpc/request/27 {"method":"setValueStatus","params":{"status":true}}
^C
C:\Program Files\Mosquitto>mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rp
c/request/+" -u "GLG4fcKpAH409QL06NF9"
v1/devices/me/rpc/request/87 {"method":"setValueStatus","params":{"status":false,"button1
":true,"button2":true}}
```

Рисунок 63 – Получение запроса

```
Командная строка

C:\Program Files\Mosquitto>mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/rpc/response/87" -u "GLG4fcKpAH409QL06NF9" -m '{"status": true}'
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/rpc/response/87', ... (16 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT
```

Рисунок 64 – Отправка неверного ответа

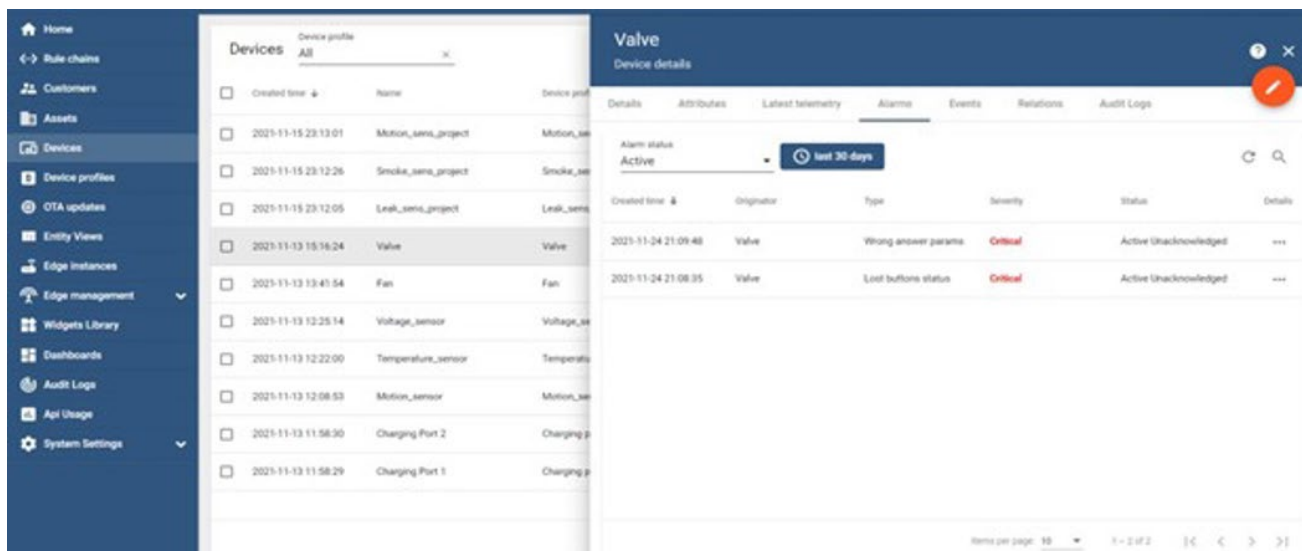


Рисунок 65 – Появление тревог

Device Valve Lost buttons status



Device Valve has alarm with type Lost buttons status

Рисунок 66 – Сообщение о тревоге из-за отсутствия состояний кнопок

Device Valve Wrong answer params



Device Valve has alarm with type Wrong answer params

Рисунок 67 – Сообщение о тревоге из-за неверного ответа

Дополнительное задание к ПР №9

Часть 1, выбор IoT-платформы:

В качестве IoT-платформы была выбрана платформа ThingsBoard Demo, поскольку обладает необходимым функционалом в бесплатной версии, удобна в использовании и подключении к устройствам.

Часть 2, отправка данных с физического устройства в выбранную IoT-платформу по протоколу MQTT:

Данные на IoT-платформу передаются с программного эмулятора трех датчиков: дыма, движения и протечки. Для каждого из них на платформе было создано своё виртуальное устройство (рис. 68).

Протокол MQTT был выбран, так как удобен в использовании и различные реализации во многих языках программирования.

Идентификация получающего устройства будет осуществляться по уникальному, заранее установленному `client_id`.

<input type="checkbox"/>	Created time ↓	Name	Device profile
<input type="checkbox"/>	2021-11-15 23:13:01	Motion_sens_project	Motion_sens_project
<input type="checkbox"/>	2021-11-15 23:12:26	Smoke_sens_project	Smoke_sens_project
<input type="checkbox"/>	2021-11-15 23:12:05	Leak_sens_project	Leak_sens_project

Рисунок 68 – Добавление виртуальных устройств

Далее приведён листинг (листинг 11) подключения к устройствам и отправки на них показаний. Каждое сообщение содержит в себе id устройства, name устройства и несколько измеряемых параметров.

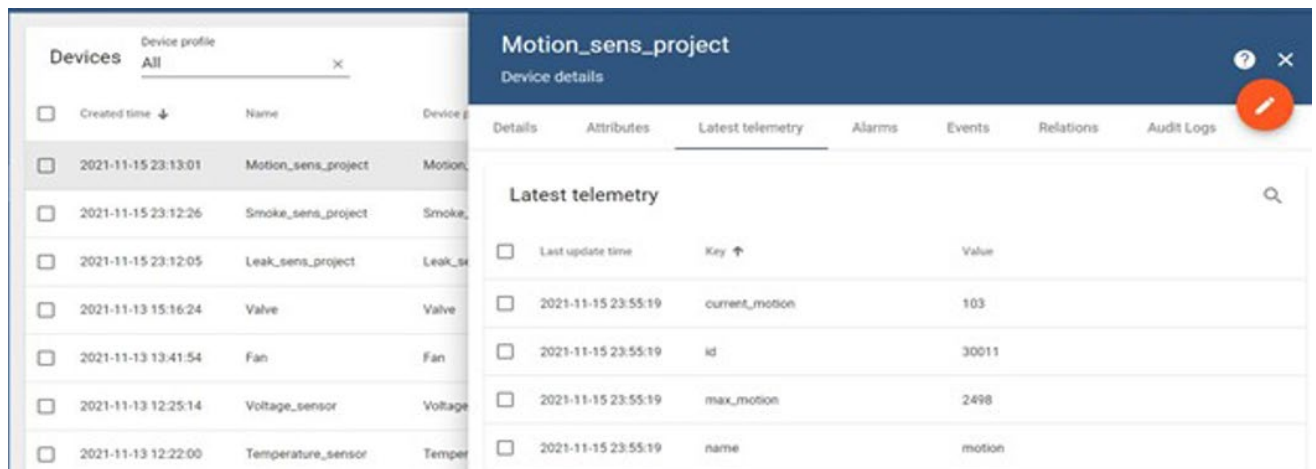
Листинг 11 – Подключение и отправка сообщений

```
import paho.mqtt.publish as publish
import time
import values_generator_thingsboard

while True:
    msgs = [
        ("v1/devices/me/telemetry", values_generator_thingsboard.rand_smoke(), 0, True),
        ("v1/devices/me/telemetry", values_generator_thingsboard.rand_leak(), 0, True),
        ("v1/devices/me/telemetry", values_generator_thingsboard.rand_motion(), 0, True)
    ]
    publish.multiple(msgs[0], hostname='demo.thingsboard.io', port=1883,
        client_id='my_smoke_sens_1', transport='tcp')
    publish.multiple(msgs[1], hostname='demo.thingsboard.io', port=1883,
        client_id='my_leak_sens_1', transport='tcp')
    publish.multiple(msgs[2], hostname='demo.thingsboard.io', port=1883,
        client_id='my_motion_sens_1', transport='tcp')

    time.sleep(1)
```

Данные, которые получают виртуальные устройства с эмулятора датчиков (в частности, последнее полученное сообщение), изображены на рисунках 69 – 71.



The screenshot shows the ThingsBoard web interface. On the left, a 'Devices' table lists several virtual devices. On the right, the 'Motion_sens_project' device details are shown, with the 'Latest telemetry' tab selected, displaying a table of recent sensor readings.

Created time	Name	Device type
2021-11-15 23:13:01	Motion_sens_project	Motion
2021-11-15 23:12:26	Smoke_sens_project	Smoke
2021-11-15 23:12:05	Leak_sens_project	Leak
2021-11-13 15:16:24	Valve	Valve
2021-11-13 13:41:54	Fan	Fan
2021-11-13 12:25:14	Voltage_sensor	Voltage
2021-11-13 12:22:00	Temperature_sensor	Temper

Motion_sens_project		
Device details		
Latest telemetry		
Last update time	Key	Value
2021-11-15 23:55:19	current_motion	103
2021-11-15 23:55:19	id	30011
2021-11-15 23:55:19	max_motion	2498
2021-11-15 23:55:19	name	motion

Рисунок 69 – Данные, получаемые виртуальным устройством от датчика движения

Smoke_sens_project			
Device details			
Latest telemetry			
Last update time	Key	Value	
2021-11-15 23:55:19	density	1.58	
2021-11-15 23:55:19	id	10011	
2021-11-15 23:55:19	max_temperature	34.37	
2021-11-15 23:55:19	name	smoke	

Рисунок 70 – Данные, получаемые виртуальным устройством от датчика дыма

Leak_sens_project			
Device details			
Latest telemetry			
Last update time	Key	Value	
2021-11-15 23:55:19	density	false	
2021-11-15 23:55:19	id	20011	
2021-11-15 23:55:19	name	leak	

Рисунок 71 – Данные, получаемые виртуальным устройством от датчика протечки

Дополнительное задание к ПР №10

Часть 1, описание взаимодействия пользователя с интерфейсом:

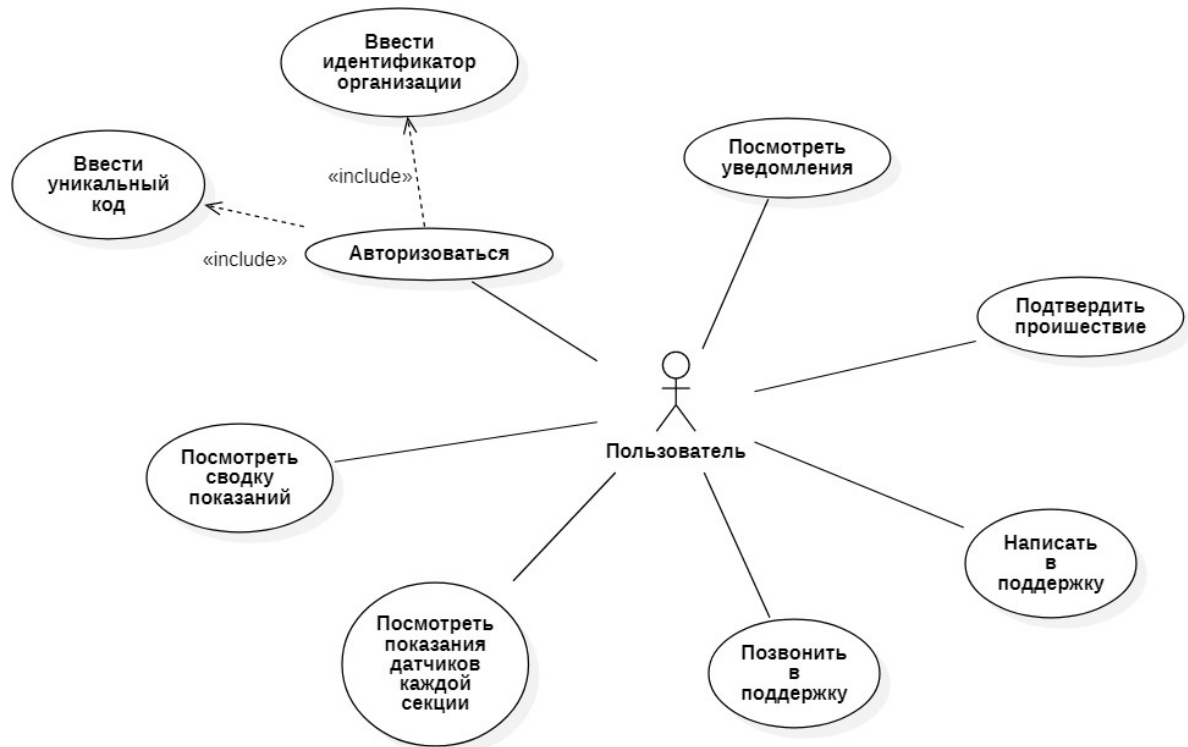


Рисунок 72 – Диаграмма взаимодействия пользователя с интерфейсом

Часть 2, требования с интерфейсу пользователя:

Интерфейс будет реализован в виде мобильного приложения. Пользователь сможет получать информацию о чрезвычайных ситуациях или предпосылках к их возникновению с помощью уведомлений. Также пользователь может посмотреть сводку показаний по всем секциям. Из сводки показаний пользователь может перейти к просмотру графиков показаний датчиков каждой секции.

Перед использованием приложения пользователь должен будет авторизоваться, используя идентификатор организации и уникальный код. В случае возникновения каких-либо неполадок пользователь может связаться с технической поддержкой.

Часть 3, макет интерфейса:

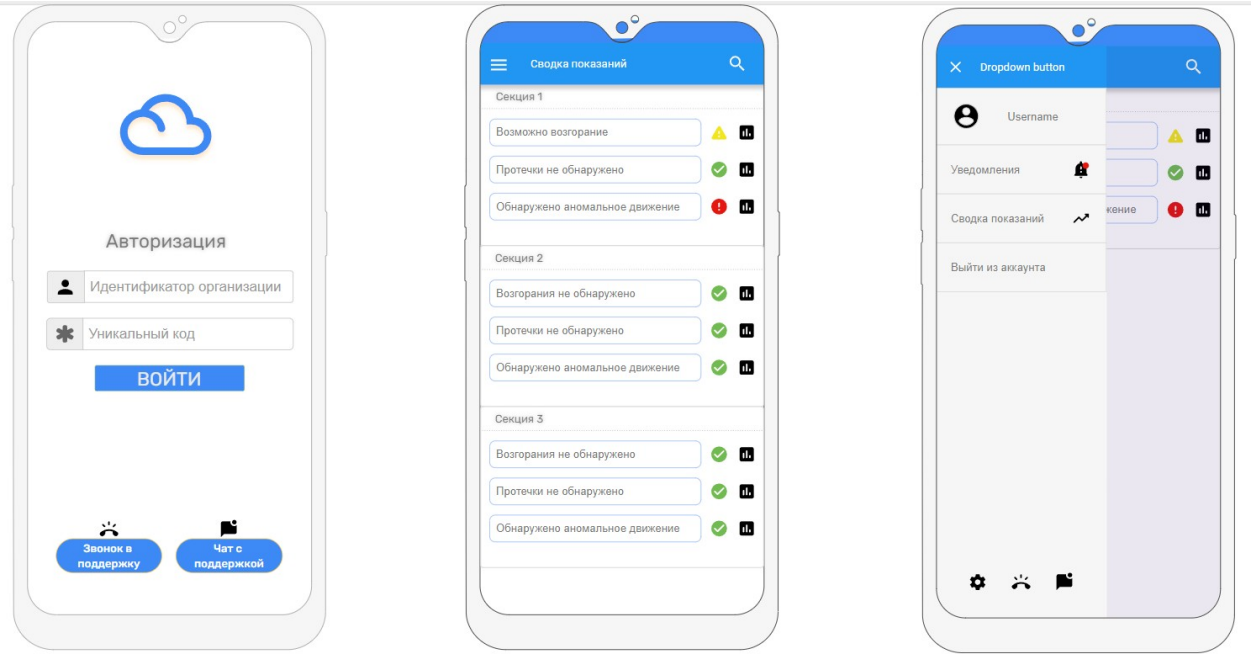


Рисунок 73 – Макет интерфейса (часть 1)

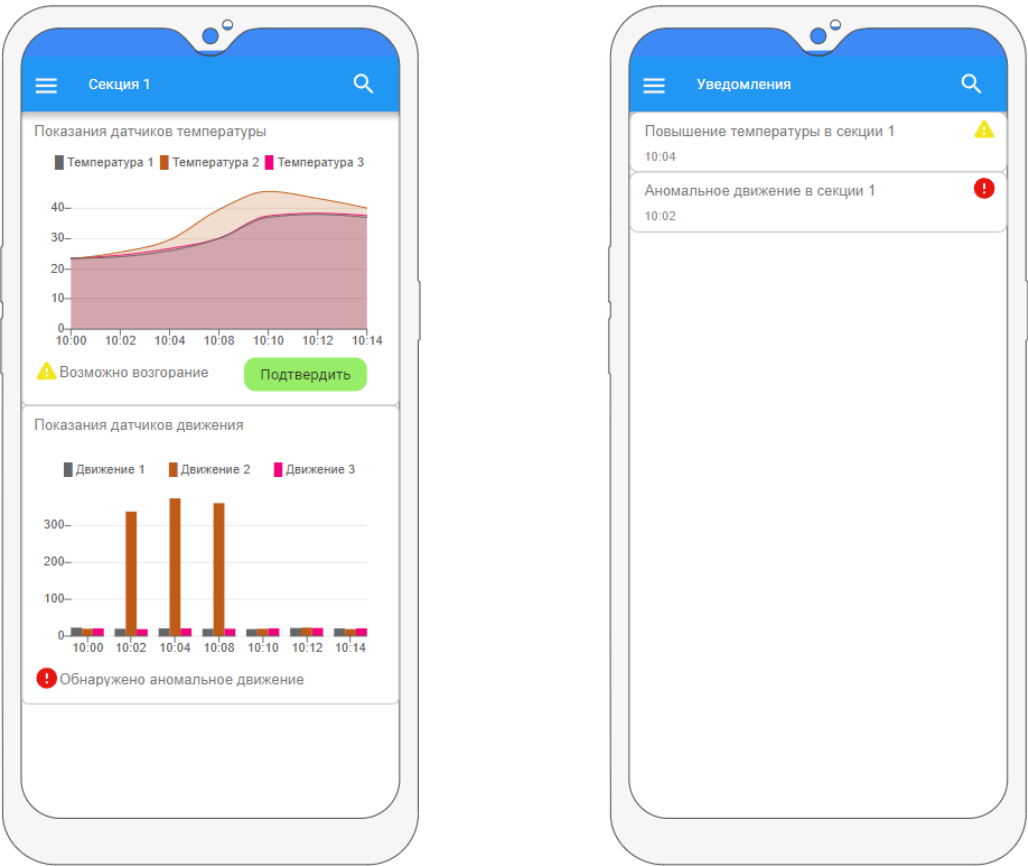


Рисунок 74 – Макет интерфейса (часть 2)

ЗАКЛЮЧЕНИЕ

В процессе выполнения данной практической работы были получены теоретические и практические знания об облачных платформах, хранилищах и вычислениях и об их положении в концепции Интернета вещей. Было смоделировано поведение виртуальных устройств на платформе ThingsBoard. Выполнена передача тестовых данных в каждое из созданных устройств. Получены навыки разработки скриптов при помощи цепочек правил ThingsBoard.