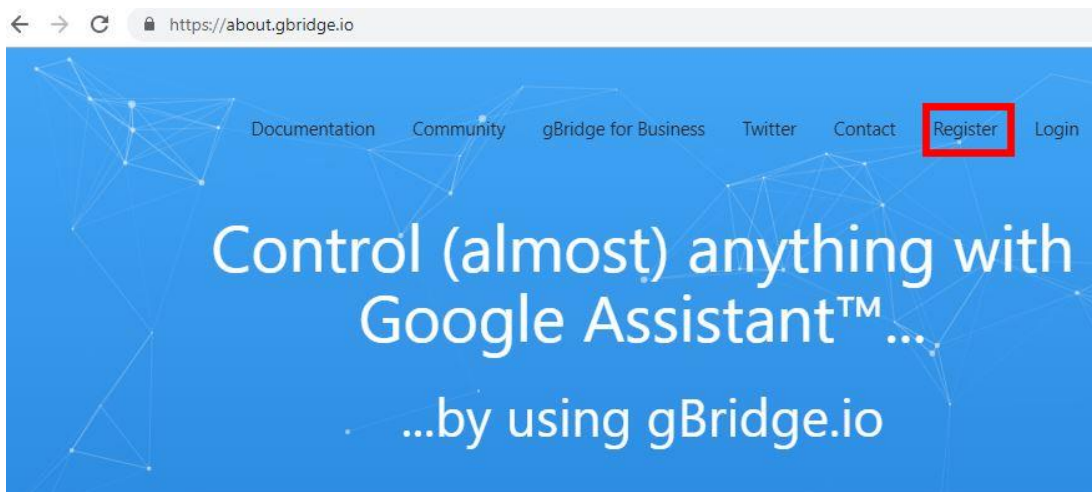


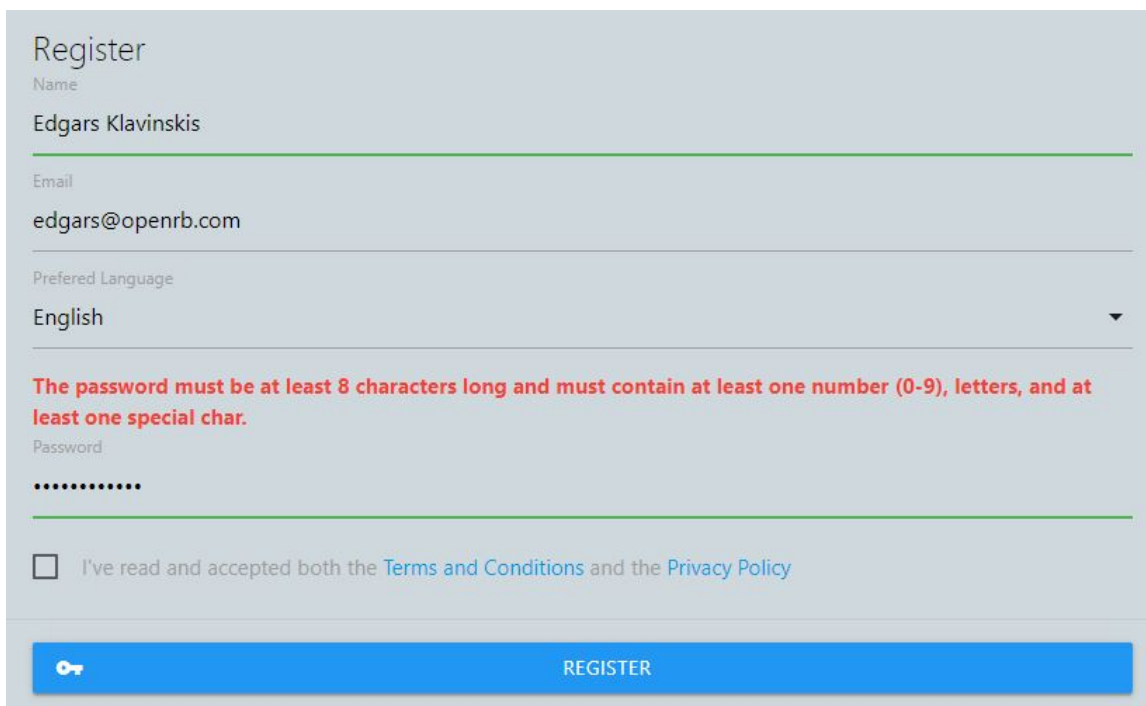
Embedded Systems SIA, VAT No LV40003411103  
47. Katolu str., Riga, LV 1003, LATVIA  
Phone: +371 67648888, fax: +371 67205036, e-mail: [sales@openrb.com](mailto:sales@openrb.com)

## Google Home integration into LogicMachine

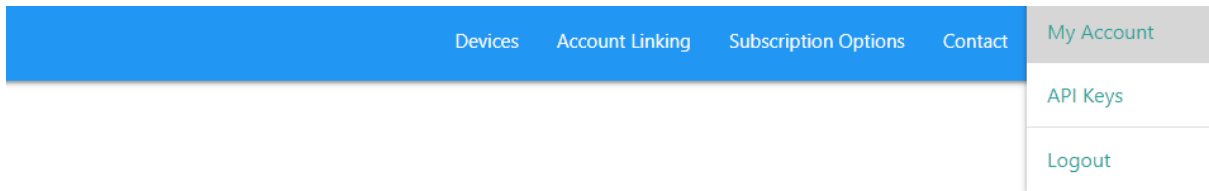
1. Register at gbridge.io website



2. Please note to use the correct password when registering – the password must be at least 8 characters long, contain at least one number (0-9), letters, and at least one special character

A screenshot of the registration form on the gbridge.io website. The form is titled "Register" and includes the following fields: Name (Edgars Klavinskis), Email (edgars@openrb.com), and Preferred Language (English). A red warning message states: "The password must be at least 8 characters long and must contain at least one number (0-9), letters, and at least one special char." The password field is currently filled with dots. At the bottom, there is a checkbox for "I've read and accepted both the Terms and Conditions and the Privacy Policy" and a blue "REGISTER" button.

3. In the upper right corner go to **My Account**.



4. Fill the MQTT password fields along with entering your gbridge account password, click the button **Change MQTT password**

## MQTT server

See [the documentation](#) for more information and alternatives.

- **MQTT server:** mqtt.gbridge.io
- **MQTT port:** 8883
- **MQTT protocol:** Version 3.1
- **TLS:** TLS V1.2 required.  
Certificate is signed by Let's Encrypt, so use the CAs of your system (for example under /etc/ssl/certs/ for Debian based systems).  
[Only download the CA files from here](#) if your system does not support Let's Encrypt CA natively.
- **Username:** gbridge-u97
- **Password:** (change below)

Account Password


---

New MQTT Password

---

New MQTT Password (confirm)

---

 CHANGE MQTT PASSWORD

5. In the **Devices** section click **+Device** button and fill the information about object/group address to control. Choose **name** which will be used to control, **device type** (Light, outlet, switch, scene, thermostat, fan, air conditioner, air purifier, sprinkler, door, blinds, shutter, dishwasher, dryer, vacuum, washer, camera) and **traits** (on/off, brightness, scene, temperature setting, fan speed, start/stop, open/close, camera stream, color setting/RGB, colors setting/JSON, color setting/temperature). Click **+Add** button when finished.

gBridge.io Devices Account Linking Subscription Options Contact Hi reproart!

## New Device

[← OVERVIEW](#)

Name  
Switch

---

Device Type  
Switch

Traits  
Select Supported Traits, On and Off

[+ ADD](#)

6. You can see now the MQTT topic for this device. You can edit it aswell by clicking edit button



gBridge.io Devices Account Linking Subscription Options Contact

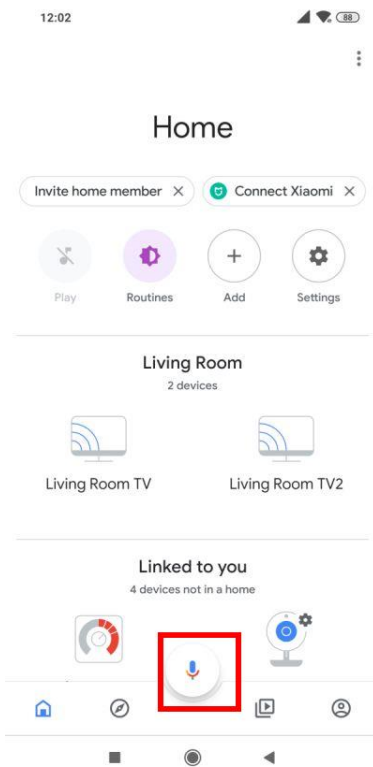
## All Devices

Switch: **switch**

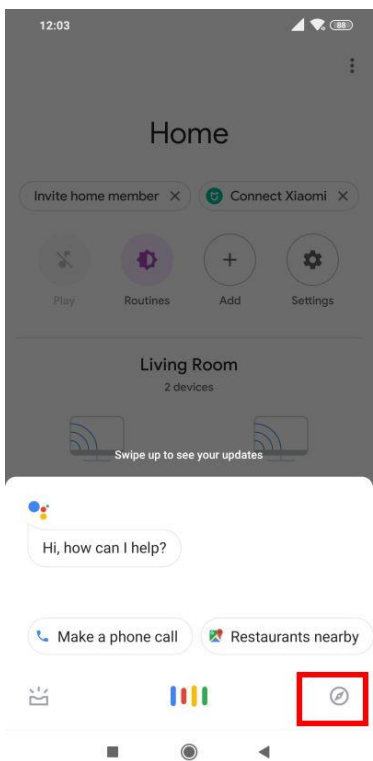
**Features and MQTT-Topics:**  
On and Off

gBridge/u97/d2697/onoff	gBridge/u97/d2697/onoff/set
-------------------------	-----------------------------

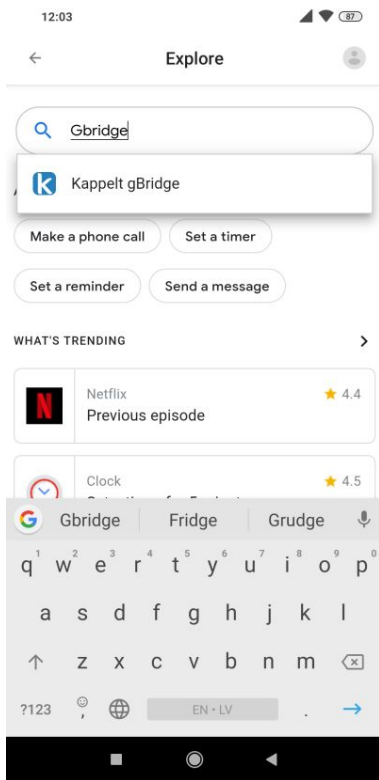
7. In the Google Home app on your phone, click the Mic button.



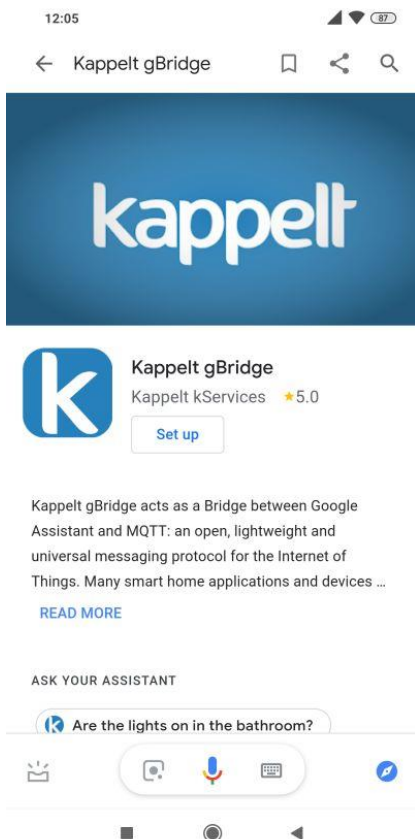
8. Then click Explore button



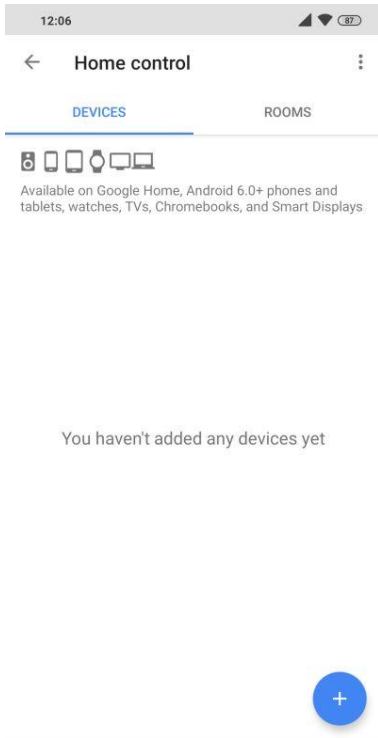
9. In the Search field enter Gbridge



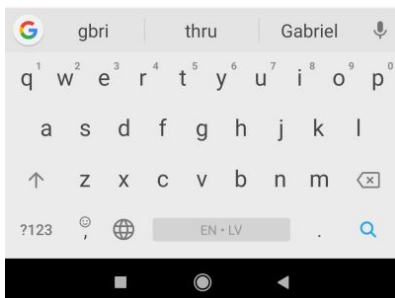
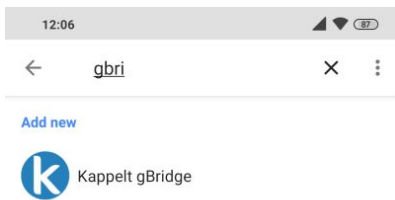
10. Click **Set up** button.



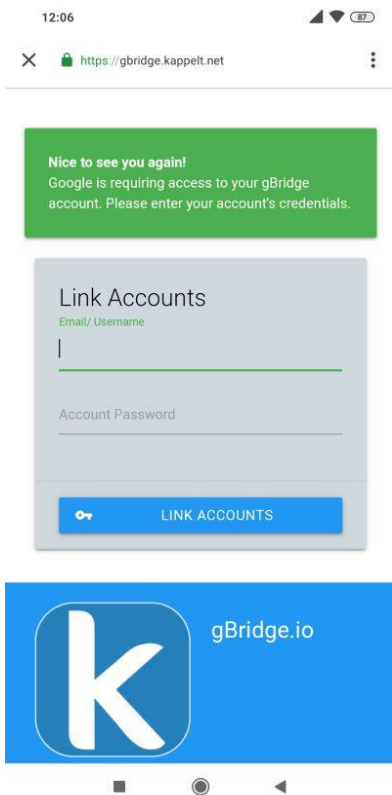
## 11. Click + button



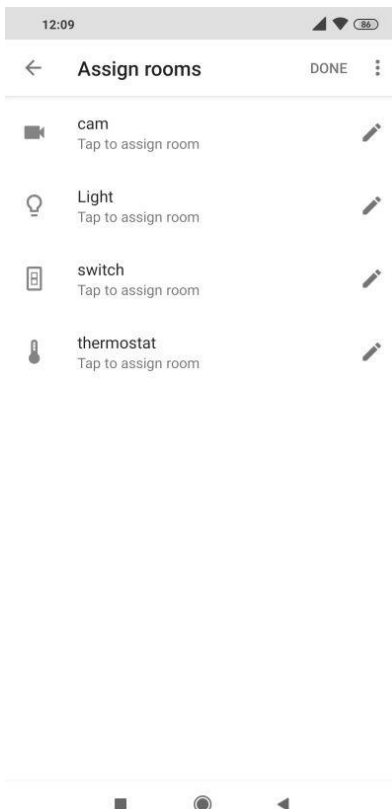
## 12. Search for Gbridge



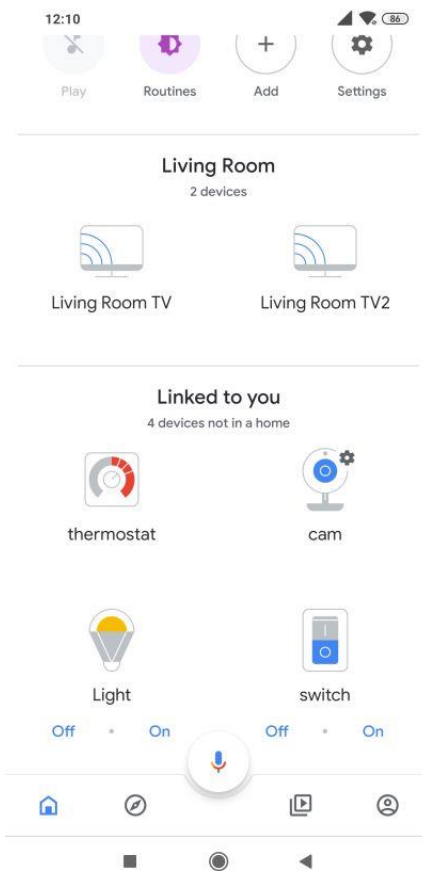
13. Login with your account login name and password (created in 2<sup>nd</sup> point above)



14. You will then see Devices which you have added in the web interface for the gbridge.io



15. Now you can control these Devices from Google Home manually or via voice e.g. "Switch ON"



16. You can test that the communication goes well from Windows/Apple computer by using MQTTBox program: [http://workswithweb.com/html/mqttbox/installing\\_apps.html#install\\_on\\_windows](http://workswithweb.com/html/mqttbox/installing_apps.html#install_on_windows)  
Fill the fields like in the below example picture, change mqtt **username** and **password** to the ones you created in the 4<sup>th</sup> point above. Click **Save**.



MQTTBox

MQTT CLIENT SETTINGS Client Settings Help

**MQTT Client Name**

**MQTT Client Id**

**Append timestamp to MQTT client id?**  
 Yes  No

**Broker is MQTT v3.1.1 compliant?**  
 No  Yes

**Protocol**

**Host**

**Clean Session?**  
 Yes  No

**Auto connect on app launch?**  
 Yes  No

**SSL / TLS Version**

**SSL / TLS Certificate Type**

**Username**

**Password**

**Reschedule Pings?**  
 Yes  No

**Queue outgoing QoS zero messages?**  
 Yes  No

**Reconnect Period (milliseconds)**

**Connect Timeout (milliseconds)**

**KeepAlive (seconds)**

**Will - Topic**

**Will - QoS**

**Will - Retain**  
 No  Yes

**Will - Payload**

17. Make sure the status for the connection is **Connected**:

MQTTBox

MQTTBox Edit Help

Connected

mqtt.gbridge.io - mqttts://mqtt.gbridge.io

**Topic to publish**

**QoS**

**Retain**

**Payload Type**  
  
 e.g: {'hello':'world'}

**Payload**

**Topic to subscribe**

**QoS**

18. In the Topic to subscribe enter the topic name from the **Device** section, click **Subscribe**

## All Devices

Switch: **switch**

Features and MQTT-Topics:

On and Off

[gBridge/u97/d2697/onoff](#)

[gBridge/u97/d2697/onoff/set](#)

MQTTBox

MQTTBox Edit Help

Menu



Connected

Add publisher

Add subscriber



mqtt.gbridge.io - mqtt://mqtt.gbridge.io

Topic to publish

QoS

0 - Almost Once

Retain

Payload Type

Strings / JSON / XML / Characters

e.g. {'hello':'world'}

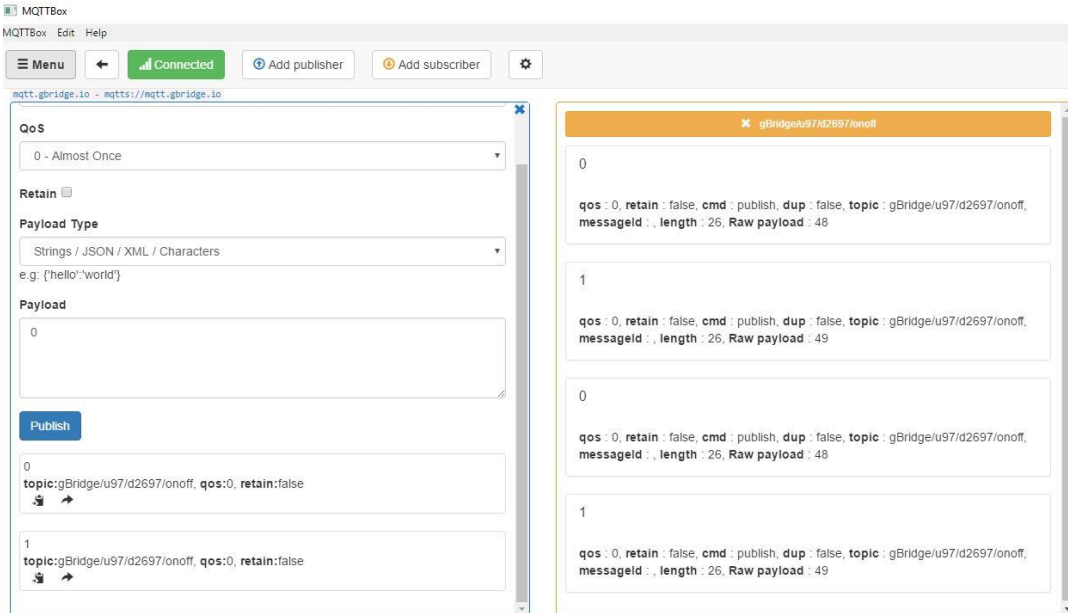
Topic to subscribe

QoS

0 - Almost Once

Subscribe

19. In the Google Home app change this Device status On/Off, you should see communication in the Subscribe window:



20. In the LogicMachine update these two packages (System Config → System → Packages :

New power CPU:

[http://dl.openrb.com/pkg/libmosquitto\\_1.6.3-1\\_imx6.ipk](http://dl.openrb.com/pkg/libmosquitto_1.6.3-1_imx6.ipk)

[http://dl.openrb.com/pkg/luamosquitto\\_0.3-4\\_imx6.ipk](http://dl.openrb.com/pkg/luamosquitto_0.3-4_imx6.ipk)

[http://dl.openrb.com/pkg/luasocket\\_2.0.2-34\\_imx6.ipk](http://dl.openrb.com/pkg/luasocket_2.0.2-34_imx6.ipk)

Old CPU:

[http://dl.openrb.com/pkg/luasocket\\_2.0.2-35\\_mxs.ipk](http://dl.openrb.com/pkg/luasocket_2.0.2-35_mxs.ipk)

[http://dl.openrb.com/pkg/libmosquitto\\_1.6.3-1\\_mxs.ipk](http://dl.openrb.com/pkg/libmosquitto_1.6.3-1_mxs.ipk)

[http://dl.openrb.com/pkg/luamosquitto\\_0.3-5\\_mxs.ipk](http://dl.openrb.com/pkg/luamosquitto_0.3-5_mxs.ipk)

21. Resident script, sleep time = 0.

Edit these variables:

username - your gbridge MQTT username

password - your gbridge MQTT password

controlmap - topic -> address map for control (from MQTT)

statusmap - address -> topic map for statuses (to MQTT)

Thermostat mode mapping to/from numeric values:

off = 0

on = 1

heat = 2

cool = 3

```

    auto = 4
    fan-only = 5
    purifier = 6
    eco = 7
    dry = 8

-- script begins here

if not mclient then
  username = 'gbridge-user'
  password = 'password'

  -- control, topic -> address map
  controlmap = {
    ['gBridge/user/device/onoff'] = '32/1/1',
    ['gBridge/user/device/onoff'] = '32/1/2',
    ['gBridge/user/device/tempset-setpoint'] = '32/1/3',
    ['gBridge/user/device/tempset-mode'] = '32/1/6',
  }

  -- status, address -> topic map
  statusmap = {
    ['32/1/1'] = 'gBridge/user/device/onoff/set',
    ['32/1/2'] = 'gBridge/user/device/onoff/set',
    ['32/1/3'] = 'gBridge/user/device/tempset-setpoint/set',
    ['32/1/4'] = 'gBridge/user/device/tempset-ambient/set',
    ['32/1/5'] = 'gBridge/user/device/tempset-humidity/set',
    ['32/1/6'] = 'gBridge/user/device/tempset-mode/set',
  }

  -- thermostat mode -> number map and vice versa
  modemap = {
    ['off'] = 0,
    ['on'] = 1,
    ['heat'] = 2,
    ['cool'] = 3,
    ['auto'] = 4,
    ['fan-only'] = 5,
    ['purifier'] = 6,
    ['eco'] = 7,
    ['dry'] = 8,
    [0] = 'off',
    [1] = 'on',
    [2] = 'heat',
    [3] = 'cool',
    [4] = 'auto',
    [5] = 'fan-only',
    [6] = 'purifier',
    [7] = 'eco',
    [8] = 'dry',
  }

  socket = require('socket')

  host = 'mqtt.gbridge.io'
  port = 8883

  datatypes = {}
  values = {}

  for addr, _ in pairs(statusmap) do
    obj = grp.find(addr)

    if obj then
      datatypes[ addr ] = obj.datatype
      values[ addr ] = obj.value
    else
      log('object not found ' .. tostring(addr))
    end
  end
end

```

```

        statusmap[ addr ] = nil
    end
end

function istempset(topic)
    return topic:find('tempset-mode', 1, true)
end

function publish(topic, value)
    local data

    if istempset(topic) then
        data = modemap[ tonumber(value) or 0 ]
    else
        data = tostring(value)
    end

    mclient:publish(topic, data)
end

mclient = require('mosquitto').new()

mclient.ON_CONNECT = function(status, rc, msg)
    connected = status

    if status then
        log('mqtt connected')

        for topic, _ in pairs(controlmap) do
            mclient:subscribe(topic)
        end

        for addr, topic in pairs(statusmap) do
            local value = values[ addr ]
            if value ~= nil then
                publish(topic, value)
            end
        end
    else
        log('mqtt on_connect failed ' .. tostring(msg))

        mclient:disconnect()
    end
end

mclient.ON_MESSAGE = function(mid, topic, data)
    local addr = controlmap[ topic ]

    if addr then
        local value

        if istempset(topic) then
            value = modemap[ data ]
        else
            value = tonumber(data)
        end

        if value ~= nil then
            grp.write(addr, value)
        end
    end
end

mclient.ON_DISCONNECT = function(status, rc, msg)
    if connected then
        connected = false
        log('mqtt disconnected ' .. tostring(msg))
    end

    mclientfd = nil
end

```

```

function mconnect()
    local status, rc, msg, fd

    status, rc, msg = mclient:connect(host, port)

    if not status then
        log('mqtt connect failed ' .. tostring(msg))
    end

    fd = mclient:socket()
    if fd then
        mclientfd = fd
    end
end

function busevent(event)
    local addr = event.dst
    local topic = statusmap[ addr ]

    if topic and connected then
        local value = busdatatype.decode(event.datahex, datatypes[ addr ])
        values[ addr ] = value
        publish(topic, value)
    end
end

lb = require('localbus').new(10)

lb:sethandler('groupwrite', busevent)
lb:sethandler('groupresponse', busevent)

busfd = socket.fdmaskset(lb:getfd(), 'r')

mclient:tls_insecure_set(true)
mclient:login_set(username, password or '')

mconnect()

timer = require('timerfd').new(5)
timerfd = socket.fdmaskset(timer:getfd(), 'r')
end

-- mqtt connected
if mclientfd then
    mask = mclient:want_write() and 'rw' or 'r'
    mclientfdset = socket.fdmaskset(mclientfd, mask)
    res, busstat, timerstat, mclientstat =
        socket.selectfds(10, busfd, timerfd, mclientfdset)
-- mqtt not connected
else
    res, busstat, timerstat =
        socket.selectfds(10, busfd, timerfd)
end

if mclientstat then
    if socket.fdmaskread(mclientstat) then
        mclient:loop_read()
    end

    if socket.fdmaskwrite(mclientstat) then
        mclient:loop_write()
    end
end

if busstat then
    lb:step()
end

if timerstat then
    timer:read() -- clear armed timer

    if mclientfd then

```

```
mclient:loop_misc()
else
  mconnect()
end
end
```

22. Source: <https://github.com/kservices/gBridge>

23. To increase device limit in free version, you can use objects like Temperature or Scene which has several sub-objects. Each of these sub-objects can be used to control separate object on LM

## Thermostat: **temp**



### Features and MQTT-Topics:

On and Off

gBridge/u97/d7328/onoff

gBridge/u97/d7328/onoff/set

Temperature Setting -  
Mode

gBridge/u97/d7328/tempset-mode

gBridge/u97/d7328/tempset-mode/set

Temperature Setting -  
Setpoint

gBridge/u97/d7328/tempset-setpoint

gBridge/u97/d7328/tempset-setpoint/set

Temperature Setting -  
Ambient

gBridge/u97/d7328/tempset-ambient/set

Temperature Setting -  
Humidity

gBridge/u97/d7328/tempset-humidity/set