RFM69-MQTT-Gateway

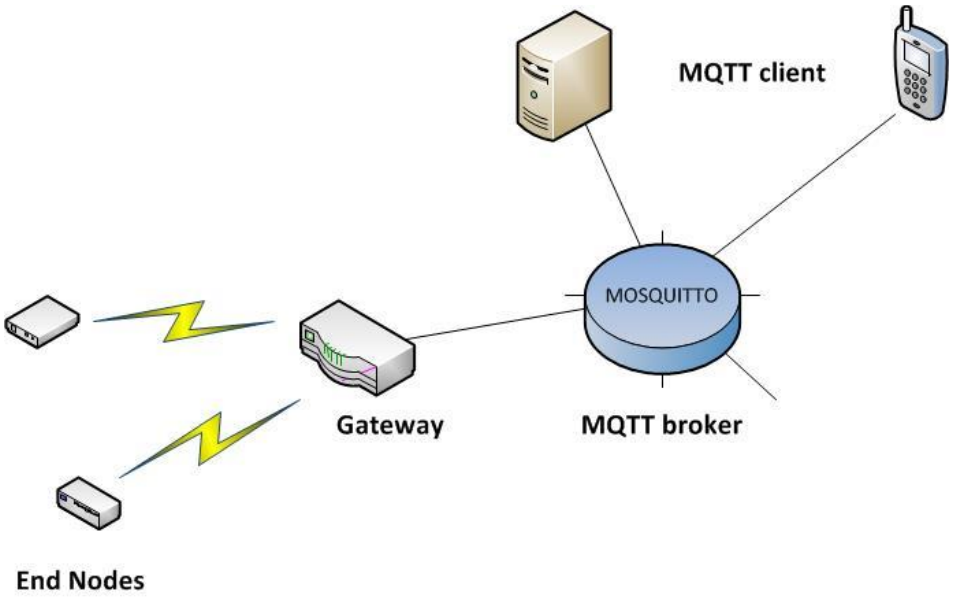
Introduction

The Internet of Things is now in the Hype / Expectation stage. Many companies are delivering solutions, expectations are high, and a bright future seems ahead. In reality there are numerous proprietary protocols and solutions around, many definitions and interpretations and no agreement on final solutions.

IOT will be mature once a standard way of interconnecting devices is around, irrespective of make or type of device. Suppliers seem to stick with partial solutions and are hesitant in providing products that work well in an interconnected world.

MQTT (Message Queue Telemetry Transport) is a likely candidate to act as a common protocol to connect the “things” in IOT. It is simple, lightweight and open source implementations of a message broker exist. In this way MQTT could be the catalyst needed to get the IOT moving.

This project implements a gateway between the MQTT protocol and wireless end nodes.



As shown in the picture the gateway acts as an interface between MQTT and the end node network.

Main requirements for the setup are:

- The gateway should be more or less transparent to node software. Different types or versions of node software should be able to work in a single network
- Traffic should be duplex (both directions) and encrypted
- High quality service is guarantees by using transmission retries, handshakes, etc.
- End nodes can be sensor nodes, actuator nodes, or a combination
- End nodes can be managed to a certain extend over the network
- Error conditions in the radio network are reported

Data format

A universal format for data exchange for several different end nodes has been derived, based on addressing a specific end node (nodeID) and a specific device on that end node (devID).

The data exchange format is independent of software implementation or end node capability. This ensures a standard gateway setup and the possibility to mix different end nodes and software versions within a single network. At this moment the following device groups have been defined:

0 – 15:	system devices
16 – 31:	Binary output devices (LED, relay, etc)
32 – 39:	Integer output devices (dimmer, pwm-module, etc)
40 – 47:	Binary input devices (switch, PIR-module, etc)
48 – 63:	Decimal number input devices (temperature, humidity, voltage)
64 – 71:	Integer number input
72:	String transfer device
73 – 90:	Special devices (not implemented in the gateway)
90 – 99:	Error messages

The system devices are:

00	Uptime:	Time (in minutes) since start-up of the node
01	Tx interval:	Flag/periodic transmission interval in seconds
02	RSSI:	Signal strength
03	Version:	Software version end node
04	Voltage:	Battery Voltage
05	ACK:	Flag for acknowledge messages to be sent or not
06	toggle:	Flag for toggle action for button on end node
07	timer:	Flag/Interval for timer action on end node
08	btnpress:	Flag for button press message to be sent

Error messages are:

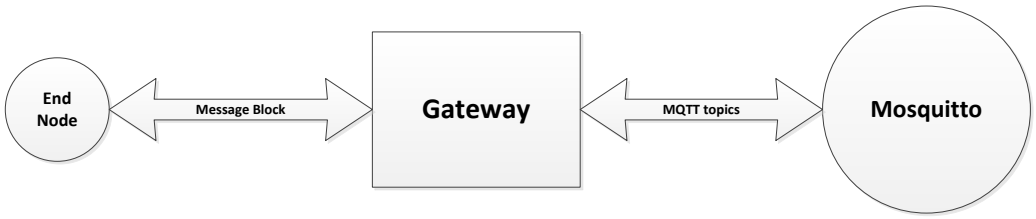
90	link error:	the radio link has failed
91	syntax error:	the MQTT message contains a syntax error
92	invalid device:	the device addressed does not exist (not defined in end-node)
99:	wakeup:	message sent on end node startup

The syntax error message contains an error-code:

1:	syntax error in MQTT topic
2:	empty MQTT message (no payload)
3:	syntax error in MQTT message
4:	invalid device (not defined in gateway)

Data exchange

The setup of the gateway is shown below:



Gateway and end nodes exchange a fixed data block of max 66 bytes in length.
It contains the following data fields:

-	nodeID	integer	indicates the nodeID of the end node
-	devID	integer	indicates the device on the end node
-	cmd	integer	indicated whether to SET or READ a parameter value
-	intVAL	long	Long Integer payload
-	fltVAL	float	Real payload
-	payload	Array[32]	Character payload

These values are always exchanged; depending on the function addressed values have different meanings: when reading temperature, the value is in fltVAL; when querying TX interval the value is in intVAL. According to device type the gateway will pick up the proper value.

The format of the MQTT ***topic*** is :

home/rfm_gw/direction/nodeID/deviceID

Where:

Direction: nb (northbound) for messages from end node to broker
sb (southbound) for messages from broker to end node.
Without this distinction the end node would receive its own messages.

nodeID: the ID of the end node addressed. Each node has an unique ID that was fixed during compile time. The Gateway has ID 01.

deviceID: the id according to the list one page earlier.

The MQTT ***message*** (payload) will be different according to device and direction:

Southbound: The topic will contain commands for the end node: ON, OFF, READ or values for parameters to be set (for example: number of seconds).

Northbound: The topic will contain the value of the parameter corresponding to device ID.

The gateway will subscribe to southbound messages for all end nodes in the network.
The subscription topic uses a wildcard and is:

home/rfm_gw/sb/#

All messages with the subscription will be parsed by the gateway. According to the nodeID the proper end node will be addressed, according to device ID the proper device will be addressed, and according to message content the proper command will be carried out.

Examples:

topic: home/rfm_gw/sb/node02/dev16 ;
message: ON — will switch on the LED on node 2

topic: home/rfm_gw/sb/node02/dev16 ;
message: READ — will read the LED-status on node 2

topic: home/rfm_gw/sb/node02/dev01
message: 300 — will set the transmission interval to 5 minutes

topic: home/rfm_gw/sb/node02/dev01
message: 0 — will disable periodic transmissions

topic: home/rfm_gw/sb/node02/dev48
message: READ — will read temperature

topic: home/rfm_gw/sb/node02/dev02
message: READ — will read radio signal strength

Gateway

The gateway links the MQTT-broker to the end-nodes over the radio network. Main functions are:

- route messages to the proper end-node
- translate between MQTT and radio format and vice versa
- Ensure high quality delivery and notification of errors
- Check proper syntax of messages

The device groups as defined earlier are implemented in the gateway and will be handled according to specific properties of each device. The gateway will not differentiate between devices within a device group; that is up to the end-node.

The gateway always has node number 1 and has two system devices: Uptime (0) and Version (3).

For Example, the following message will return the Gateway Uptime:

- Topic: home/rfm_gw/sb/node01/dev00;
- message: READ

NOTE: Syntax checking is done in the gateway. Any error messages will be originated in the gateway; te receive them make sure to subscribe to MQTT topic: ***home/rfm_gw/nb/node01/#***

End node

The end node is the link between the physical world and the IOT network.

The end node:

- handles the radio link to the gateway and handles handshakes, retransmissions, etc.
- has the capability to periodically send sensor data (push model)
- can be queried and asked to provide sensor data (pull model)
- has some intelligence for local control of the actuator (toggle, timer)
- limits the number of messages to prevent overloading the gateway

On start-up a default configuration is loaded. This configuration is provided on compile time and offers possibility to:

- send sensor data periodically or on demand
- set length of periodic transmission interval
- determine type of sensor data to be sent
- set operation mode of the local button (toggle, timer of send a message)
- acknowledge all “SET” operations with a status message

During operation the configuration can be changed by setting device values:

home/rfm_gw/sb/nodexx/dev01:	number of seconds for the TX interval value ‘0’ disables periodic transmission
home/rfm_gw/sb/nodexx/dev05:	ON enables acknowledge messages OFF disables them
home/rfm_gw/sb/nodexx/dev06:	ON allows toggling output1 with the button OFF disables toggling
home/rfm_gw/sb/nodexx/dev07:	number of seconds enables timer on output1 value ‘0’ disables timer after buttonpress
home/rfm_gw/sb/nodexx/dev10:	ON switches the output1 ON OFF switches the output OFF

For all these topics a READ message can be sent, generating a status message for the specific device.

Setup of the Gateway

NOTE: in order for the Ethernet and RFM to work together two changes are needed:
- the Slave Select signal of the RFM module should be connected to D8 of the Arduino
- the interrupt handling in the Ethernet library (W5100.h) needs to be changed
(see http://harizanov.com/2012/04/rfm12b-and-arduino-ethernet-with-wiznet5100-chip/)

On compile the following parameters should be changed:

Ethernet:

- MAC address of gateway Ethernet; a unique MAC address should be used
- Mqtt_server: the IP address of the message broker
- Ip: gateway IP address in case DHCP fails

Radio:

- NODEID: the nodeID for the gateway is 1;
- NetworkID: a common value for all network nodes should be used
- FREQUENCY: corresponding to the RFM module used
- IS_RFM69HW: corresponding to the RFM module used
- ENCRYPTKEY: a unique shared key should be used for all nodes.

Debug:
DEBUG can be uncommented to allow debugging the gateway. In this mode the serial port is used to indicate activity. Serial input can be used to emulate MQTT messages and test the radio network.

Setup of the DHT end node

On compile the following parameters should be changed:

Radio:
NODEID: each end node should have a unique nodeID

- NetworkID: a common value for all network nodes should be used
- FREQUENCY: corresponding to the RFM module used
- IS_RFM69HW: corresponding to the RFM module used
- ENCRYPTKEY: a unique shared key should be used for all nodes.

Debug:
DEBUG can be uncommented to allow debugging the gateway. In this mode the serial port is used to indicate activity.

DHTType:
I used DHT11 on 3.3 Volt. The initialisation needs an extra parameter at this voltage. Adjust accordingly.

Startup values:
The default startup values can be adjusted if needed.