ESP8266 meshed network exploration results as of 19-03-2019.

The “ESP8266MQTTMesh” library provides the possibility to have a radio mesh created automatically. This network is created at the radio link level, not at the WiFi level.
Each node (esp module) is given a unique id. For esp8266 modules this is the unique chip id. This chip id is an integer value. This id is used to create the radio mesh.

The individual nodes are not aware of the structure of the network.

One of the nodes of the mesh establishes a WiFi connection with Mosquitto.
The data exchange mesh is created at the level of Mosquitto. The structure of the network resides on Mosquitto.
This is done by a clever way of concatenating topics and subtopics and relaying messages if required. The id is part of the topic which is used to create the meshed network at MQTT level.

You can't subscribe to an arbitrary topic. The default topic that is subscribed to is “esp8266-in/” though this can be changed during initialisation.
The specified callback will be called whenever a topic of “esp8266-in/” is received (or if “esp8266-in/broadcast” is received) is the node's chipid in hex (not zero padded) I.e. if your chipid was 123456 the expected topic would be “esp8266-in/1e240/”.

By a small adaptation of the library this chip-id can be set in the software of the node to the rocnet node-id. The rocnet node-id is than used instead of the standard chip-id.

After performing some preliminary tests the result is as follows:
the outgoing topic can be set to “rocnet/rocnet node-id/. . . ”
the incoming topic can be set to “rocnet/rocnet node-id/. . .”

the incoming topic can also be set to “rocnet/broadcast/....”

Because the information is inserted by means of a pre-compiler statement, it is not possible to change this value after compilation. Each node requires separate compilation of the software.

The data which are transferred via the mesh are of the type char or string. Byte data are not supported. Rocnet can handle “HEXA”. This protocol facilitates the transfer of byte data using ASCII characters. The only drawback is that each byte requires two ASCII characters for transmission. This is not a problem because of the very small messages used with the rocnet protocol.

Some live testing without Rocrail have to be performed to confirm these findings.

Test setup

The mesh contains 2 switch decoders and 2 sensor decoders. Outside the mesh one rocrail “emulator” is used. The software for those component is tailor made for testing purposes.

Each switch decoder has an LED. Each sensor decoder has a push button.
The emulator will have two pushbuttons and two LEDs.

One test will be to see if it is possible to turn the LEDs on both decoders on and of with the push buttons of the emulator.
The second test will be to see if it is possible to switch the LEDs of the emulator on and of with the pushbuttons of the switch decoders.

New advice popped up.

Introduction of mesh – rocrail gateway.

The library is not changed.

The mesh is not able to transmit byte type of data. Ronet (RN) requires byte type of data. In RN the "HEXA" format is foreseen to convey byte data by means of ASCII characters. This will be used in a general sense.

In RN some bytes represent the originator id and some the destination id of data. Furthermore the data are segregated by means of topics.

In the mesh outgoing topics have a common first part of the topic. This part can be set by the user. (e.g. "mesh_out")
Incoming topics also have a common first part of the topic. This part can be set by the user. (e.g. "mesh_in")

Next part of the topic used by the mesh is the chip-id of the decoder. This is the source id of the message. This id is the meshId.

This meshId needs to be in the topic of the incoming message also. A second option to send a message to a mesh node would be to use mesh broadcasting.

The aim for RN is to minimise the number of messages being transmitted into the mesh. Therefore broadcasting is not used.

For RN a mapping is required.
Messages coming from the mesh are dealt with by using the RN topic structure. Only the two first parts of the topic need to be removed.

Messages going to mesh decoders require mapping from receiverId to meshId. The receiverId in RN is a two byte value.

<u>Process</u>

RN is a byte oriented protocol with a defined structure. SenderId consists of two bytes always at the same place in the message structure. ReceiverId consists of two bytes always at the same place in the message structure.
For reception of RN messages via MQTT in RN a defined topic structure is used.

HEXA is used to transform the message content from byte to ASCII.

<u>Gateway</u>
Messages from mesh decoder to Rocrail (RR):
First two parts of topic are stripped off. The content is reverted from HEXA to byte.
The second part of the topic is the sender meshId. In the RN message the RN senderId is contained. Both values are stored together. This is the mapping of meshId to RN receiverId.

Messages from RR to mesh decoder:
At the beginning of the topic the mesh incoming topic is appended ("mesh_in")
Second part of the topic is the meshId of the mesh decoder. This value is looked up in an array.

The meshId is the second part of the topic. The rest of the RN topic is appended.

The message content is converted into HEXA.

Mesh decoder
Messages from mesh decoder to RR:
The allocated RN topic is used.
The message content is converted to HEXA.

Messages from RR to mesh decoder:
The message content is reverted from HEXA to byte.
Based on the topic the message is processed according to decoder type.

Implementation
Decoders have a common software package. Their function is established by configuration. Part of the configuration procedure is the transfer of meshId from decoder to gateway. A specific message is used for this.

Boot sequence
The gateway is booted at first. Than the mesh is powered. Each decoder is calling in at the gateway. A specific message is exchanged between decoder and gateway. When all decoders are checked in normal operation starts.