mc Things™

Beaconing

White paper

Revision 0.8 (August 2016)

Contents

1. Beaconing

Beaconing is a function of the mc-Air™ wireless infrastructure. It is a low-power unencrypted message send to everybody that wants to hear it.

The major characteristics are:

- Beaconing the world that it alive and working
- Provide information to the outside world with the lowest power consumption possible
- Provides the receivers with the signal strength

Possible use cases are:

- Triangulation of the signal to find the device
- Indication if the device is in reach of a gateway
- Provide information, like temperature, etc. to other systems without draining the batteries

Beacons are send by default every 10 seconds. The energy consumption is extremely low and modules can run on a coin-cell for years. The beacon timing can be controlled with mcScript to limit the spectrum space in cases where there are a lot of modules in the same space. The beacon frequency can be increased if the device need to be discovered quickly.

The data in the beacons can be used to communicate data to gateways and from there to the Internet. The 4 bytes can be used to send for example temperature and other information without consuming much power. A beacon is more than 10 times more power efficient then making a connection to a gateway and sending a message.

Beacons will be received by all gateways that are in reach. The gateways relay the information to the LAN with UDP and optional to the cloud with MQTT.

1.1. Beacon Format

The format of the beacon is as follows:

Field	Offset	Type	Value	Remarks
ProtocolVersion	0	Byte	0x00	Version 0
MessageType	1	Byte	0x16	Beacon Type (22)
MessageVersion	2	Byte	0x01	Version 1
MessageSize	3	Byte	0x0D	Number of bytes to follow (13)
DeviceUID	4-7	Uint32		Unique Identifier Device
BeaconByte1	8	Byte		User defined data[1]
BeaconByte2	9	Byte		User defined data
BeaconByte3	10	Byte		User defined data
BeaconByte4	11	Byte		User defined data
RSSI	12	SByte		Relative Signal Strength Indicator
GatewayUID	13-16	Uint32		Unique Identifier Gateway

The unsigned 4 byte integers are placed in the buffer in the little endian format. The following example specifies a message in hex with the following information. The first line is the position in the message and the second line is the message in hex.

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
00 16 01 0D 05 00 01 00 10 11 12 13 A6 63 0A 01 00
```

The DeviceUID is 0x00010005, the beacon bytes are 0x10, 0x11, 0x12, 0x13, the RSSI is 0xA6 (-90) and GatewayUID is 0x00010A63.

[1] The default of the user defined data is the version number of the software in the device

2. Beacons over UDP

UDP is short for User Datagram Protocol and is one of the core members of the Internet protocol suite.

UDP uses a simple connectionless transmission model with a minimum of protocol mechanism. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network protocol. There is no guarantee of delivery, ordering, or duplicate protection. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram.

The UDP messages are send by the gateway. Any device in the local network can receive them. Simple setup an UdpSocket on port 25452. See a vb.net example below:

```
Dim UdpSocket As New UdpClient(25452)
While True
    Dim recvBytes As Byte() = Nothing
    Try
        Dim recvEndpoint As IPEndPoint = Nothing
        recvBytes = UdpSocket.Receive(recvEndpoint)
    Catch ex As Exception
        UdpSocket.Close()
        UdpSocket = New UdpClient(25452)
        Continue While
    End Try
    If recvBytes(1) = 22 Then
        '
        ' This test is required because the gateway send
        ' a number of different messages. 22 is Beacon
        '
        ' Handle the message by writing it in a database
        ' or communicate it to another thread.
        '
    End If
End While
```

2.1. UDP Beacon Format

The format of the beacon is as follows:

Field	Offset	Type	Value	Remarks
ProtocolVersion	0	Byte	0x00	Version 0
MessageType	1	Byte	0x16	Beacon Type (22)
MessageVersion	2	Byte	0x01	Version 1
MessageSize	3	Byte	0x0D	Number of bytes to follow (13)
DeviceUID	4-7	Uint32		Unique Identifier Device
BeaconByte1	8	Byte		User defined data[2]
BeaconByte2	9	Byte		User defined data
BeaconByte3	10	Byte		User defined data
BeaconByte4	11	Byte		User defined data
RSSI	12	SByte		Relative Signal Strength Indicator
GatewayUID	13-16	Uint32		Unique Identifier Gateway

The unsigned 4 byte integers are placed in the buffer in the little endian format. The following example specifies a message in hex with the following information. The first line is the position in the message and the second line is the message in hex.

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
00 16 01 0D 05 00 01 00 10 11 12 13 A6 63 0A 01 00
```

The DeviceUID is 0x00010005, the beacon bytes are 0x10, 0x11, 0x12, 0x13, the RSSI is 0xA6 (-90) and GatewayUID is 0x00010A63.

[2] The default of the user defined data is the version number of the software in the device

3. Beacons over MQTT

MQTT (Message Queue Telemetry Transport) is a publish-subscribe based "light weight" messaging protocol on top of the TCP/IP protocol for the Internet of Things. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited. The publish-subscribe messaging pattern requires a message broker. The broker is responsible for distributing messages to interested clients based on the topic of a message.

Every gateway can be configured for MQTT by defining a MQTT broker. The mc-Air methods Publish() and Subscribe() in mcScript communicate thru the mcGateway to the broker so they publish payloads to a topic and the broker will relay payloads to mcGateway and it will deliver the payload to the devices. If the gateway is in MQTT mode it will publish all beacons to the “mcThings/Beacons” topic.

Brokers can be in the cloud as services or run on a server in the company LAN or WAN. A small server can handle hundreds of thousands beacons per second because the protocol is very light.

The payload part of the MQTT message is beacon format and the topic is “mcThings/beacon/{GatewayUID}/{DeviceUID}”.

3.1. MQTT Beacon Format

The format of the beacon is as follows:

Field	Offset	Type	Value	Remarks
DeviceUID	0-3	Uint32		Unique Identifier Device
BeaconByte1	4	Byte		User defined data[3]
BeaconByte2	5	Byte		User defined data
BeaconByte3	6	Byte		User defined data
BeaconByte4	7	Byte		User defined data
RSSI	8	SByte		Relative Signal Strength Indicator
GatewayUID	9-12	Uint32		Unique Identifier Gateway

The unsigned 4 byte integers are placed in the buffer in the little endian format. The following example specifies a message in hex with the following information. The first line is the position in the message and the second line is the message in hex.

```
 0  1  2  3  4  5  6  7  8  9 10 11 12
05 00 01 00 10 11 12 13 A6 63 0A 01 00
```

The DeviceUID is 0x00010005, the beacon bytes are 0x10, 0x11, 0x12, 0x13, the RSSI is 0xA6 (-90) and GatewayUID is 0x00010A63.

The payload in this example will be delivered on the following topic:

mcThings/beacon/00010A63/00010005

[3] The default of the user defined data is the version number of the software in the device