wMQTT Client for M2M Node
(weightless MQTT Client)

Version 1

Istanbul, Turkey 29.03.2013

Table of Contents

Purpose of this document

This document had been prepared to help anyone who would like send MQTT Message to MQTT Server, using wMQTT Library. This document is IBM Internal Use only. Reason is, this document contains example which contains parameters to connect IBM Internal use IOC.

Purpose of wMQTT Client for M2M:

MQTT specifications had many features for light weight messaging between the devices or between devices and server. But, for the cases of M2M implementation (or just simply sending the data from sensor node to server) many of these features “many not” be necessary. Our starting point was to develop a library (in fact source code) which is “as light as possible” and neutral from the platform and connectivity between Sensor node and MQTT Server. Functions which are included in that package will be used to prepare “MQTT Messages” which will be used to send existing tcpip connectivity to server. All the functions prepare MQTT Messages which is based on MQTT V3 design specifications.

Library/Design Limitation:

Since that library had been developed “especially for M2M nodes”, many features of MQTT is not implemented.

- Message length is limited by 127 chars
- Only “Fire & Forget” type message had been implemented
- Only the functions which is necessary to send “data” from Node to server (i.e. connect Message, send Message, disconnect Message) had been implemented

Library Functions

MQTTConnect Message

This function, prepares a MQTT message string which can be used for "Connect to MQTT Server" using existing tcp/ip connectivity. This function expects a message buffer to be placed with required information for MQTT Server connection. This function will populate the given string with necessary data (i.e. Req Type, Protocol Name etc). Since, connect request message mostly contains fixed information (except client ID), supplying only ClientID is sufficient for this function.

Syntax:
void mqtt_connect_message(uint8_t * mqtt_message, char * client_id)

example:
```
mqtt_connect_message(mqttMessage, clientId);
```

where, mqTTMessage is the string that is going to be send to MQTT Server via tcp

where
byte mqtt_message [127*]; // prepared MQTT formatted *message* to be send to server
char client_id [20*] // arbitrary *client id* to be used at MQTT Connect message

Please notice that maximum MQTT message size is 127byte in wMQTT Client !

* sizes are given as example. Please use array size which is suitable for your application

MQTTPublish Message

Prepares a Publish MQTT Message (populates the string which is passed as parameter to function) which can be used to publish a MQTT Message to specified topic (ie. Parameter "topic") and given content (ie. Parameter passed as "message) .

Syntax:
void mqtt_publish_message(uint8_t * mqtt_message, char * topic, char * message)

example:
```
mqtt_publish_message(mqttMessage, topic, message);
```

where
byte mqttMessage[127*]; // prepared MQTT formatted *message* to be send to server
char topic[10*] // the name of the *topic* where message will be published
char message[80*] // the *message content* to be published at specified *topic*

* sizes are given as example. Please use array size which is suitable for your application

Please notice that maximum MQTT message size is 127byte in wMQTT Client !

MQTTDisconnect Message

Prepare a Disconnect Message (populates the string which is passed as paramteret to function) which will be used to disconnect from MQTT Server

Syntax:

void mqtt_disconnect_message(uint8_t * mqtt_message)

example:

```
mqtt_disconnect_message (mqttMessage);
```

where

byte mqttMessage[127*]; // prepared MQTT formatted *message* to be send to server

* size are given as example. Please use array size which is suitable for your application

Please notice that maximum MQTT message size is 127byte in wMQTT Client !

Example Source Code (for Arduino 1.0 or above)

```
/*
  Very, very primitive. MQTT Test program to send data to a MQTT Server

  V00 / 09th March 2013, omer sever, omers@tr.ibm.com
  initial version w/o valid IOC parameters

  V01 / 10th March 2013, ates yurdakul, atesh@puaga.com
  modified for IOC parameters
  disabled GPS for simplicity
  sending "static" information for simplicity
*/

#include <mqtt.h>

/* #include <SoftwareSerial.h>
#include "TinyGPS.h"
TinyGPS gps;
*/

// GPS parser for 406a
#define BUFFSIZ 90 // plenty big
char buffer[BUFFSIZ];
char buffidx;
char *parseptr, *initptr;
uint8_t hour, minute, second, year, month, date;
uint32_t latitude, longitude;
uint8_t groundspeed, trackangle;
char latdir, longdir;
char status;

int i = 1;
char atCommand[50];
byte mqttMessage[127];
int mqttMessageLength = 0;

String gsmStr = "                                        ";
String gprsStr = "                                       ";
char mqttSubject[50];
int index = 0;
byte data1;
boolean smsReady = false;
boolean smsSent = false;
boolean gprsReady = false;
boolean mqttSent = false;

int sentCount = 0;

// SoftwareSerial gpss(1, 3);

void setup() {

  pinMode(13, OUTPUT);
  pinMode(14, OUTPUT);      // GSM

  digitalWrite(0, HIGH);



  Serial.begin(9600);
  Serial1.begin(19200);
//  gpss.begin(9600);

  Serial.println("Hello");

  digitalWrite(14, HIGH);   // GSM ON
  delay(1000);
  digitalWrite(14, LOW);   // GSM ON
  delay(1000);

}
```

```
void loop(){

  Serial.println("Checking if GPRS is ready");
  gprsReady = isGPRSReady();

  if (gprsReady == true){
    Serial.println("GPRS Ready");
    // Change the IP and Topic.
    /* The arguments here are:
       clientID, IP, Port, Topic, Message
    */
    sendMQTTMessage("nodeTest", "test.mosquitto.org", "1883", "MFNodeTopic", "A Test data from WeightlessMQTT Client");
  }

  delay(10000);

}

uint32_t parsedecimal(char *str) {
 uint32_t d = 0;

 while (str[0] != 0) {
   if ((str[0] > '9') || (str[0] < '0'))
     return d;
   d *= 10;
   d += str[0] - '0';
   str++;
 }
 return d;
}

void readline() {
 /*char c;

 buffidx = 0; // start at begninning
 while (1) {
   c = gpss.read();
   if (c == -1)
     continue;
   Serial.print(c);
   if (c == '\n')
     continue;
   if ((buffidx == BUFFSIZ-1) || (c == '\r')) {
     buffer[buffidx] = 0;
     return;
   }
   buffer[buffidx++]= c;
 }*/
}

boolean isGPRSReady(){

 Serial1.println("AT+CGATT?");
 index = 0;
 while (Serial1.available()){
   data1 = (char)Serial1.read();
   Serial.write(data1);
   gprsStr[index++] = data1;
 }

 Serial.println("Check OK");
 Serial.print("gprs str = ");
 Serial.println(gprsStr);
 if (gprsStr.indexOf("+CGATT: 1") > -1){
   Serial.println("GPRS OK");
   return true;
 }
 else {
   Serial.println("GPRS NOT OK");
   return false;
 }
}
```

```
void sendMQTTMessage(char* clientId, char* brokerUrl, char* brokerPort, char* topic, char* message){
  Serial1.println("AT"); // Sends AT command to wake up cell phone
  Serial.println("AT");
  delay(1000); // Wait a second

  digitalWrite(13, HIGH);

  Serial1.println("AT+CSTT=\"internet\""); // Puts phone into GPRS mode
  Serial.println("AT+CSTT=\"internet\"");
  delay(2000); // Wait a second

  Serial1.println("AT+CIICR");
  Serial.println("AT+CIICR");
  delay(2000);

  Serial1.println("AT+CIFSR");
  Serial.println("AT+CIFSR");
  delay(2000);

  strcpy(atCommand, "AT+CIPSTART=\"TCP\",\"");
  strcat(atCommand, brokerUrl);
  strcat(atCommand, "\",\"");
  strcat(atCommand, brokerPort);
  strcat(atCommand, "\"");
  Serial1.println(atCommand);
  Serial.println(atCommand);
  //  Serial.println("AT+CIPSTART=\"TCP\",\"mqttdashboard.com\",\"1883\"");
  delay(2000);

  Serial1.println("AT+CIPSEND");
  Serial.println("AT+CIPSEND");
  delay(2000);

  mqttMessageLength = 16 + strlen(clientId);
  Serial.println(mqttMessageLength);
  mqtt_connect_message(mqttMessage, clientId);

  for (int j = 0; j < mqttMessageLength; j++) {
    Serial1.write(mqttMessage[j]); // Message contents
    Serial.write(mqttMessage[j]); // Message contents
  }

  Serial1.write(byte(26)); // (signals end of message)
  Serial.print("Sent");
  delay(10000);

  Serial1.println("AT+CIPSEND");
  Serial.println("AT+CIPSEND");
  delay(2000);


  mqttMessageLength = 4 + strlen(topic) + strlen(message);
  Serial.println(mqttMessageLength);
  mqtt_publish_message(mqttMessage, topic, message);
  for (int k = 0; k < mqttMessageLength; k++) {
    Serial1.write(mqttMessage[k]);
    Serial.write((byte)mqttMessage[k]);
  }

  Serial1.write(byte(26)); // (signals end of message)
  Serial.print("Sent"); // Message contents
  delay(5000);

  Serial1.println("AT+CIPCLOSE");
  Serial.println("AT+CIPCLOSE");
  delay(2000);


}
```

Credits:

Initial *"language neutral"* design by Reha Yurdakul, rehay@tr.ibm.com
Arduino Implementation by Ates Yurdakul, atesh@puaga.com
Final implementation by Omer Sever, omers@tr.ibm.com