**Листинг main.cpp**

/\* WiFi Example

 \* Copyright (c) 2016 ARM Limited

 \*

 \* Licensed under the Apache License, Version 2.0 (the "License");

 \* you may not use this file except in compliance with the License.

 \* You may obtain a copy of the License at

 \*

 \*     http://www.apache.org/licenses/LICENSE-2.0

 \*

 \* Unless required by applicable law or agreed to in writing, software

 \* distributed under the License is distributed on an "AS IS" BASIS,

 \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

 \* See the License for the specific language governing permissions and

 \* limitations under the License.

 \*/

#define MQTTCLIENT\_QOS2 1

#include "mbed.h"

#include "MQTTmbed.h"

#include "MQTTClientMbedOs.h"

#include "Sht31.h"

WiFiInterface \*wifi;

int arrivedcount = 0;

const char \*sec2str(nsapi\_security\_t sec)

{

    switch (sec) {

        case NSAPI\_SECURITY\_NONE:

            return "None";

        case NSAPI\_SECURITY\_WEP:

            return "WEP";

        case NSAPI\_SECURITY\_WPA:

            return "WPA";

        case NSAPI\_SECURITY\_WPA2:

            return "WPA2";

        case NSAPI\_SECURITY\_WPA\_WPA2:

            return "WPA/WPA2";

        case NSAPI\_SECURITY\_UNKNOWN:

        default:

            return "Unknown";

    }

}

int scan\_demo(WiFiInterface \*wifi)

{

    WiFiAccessPoint \*ap;

    printf("Scan:\n");

    int count = wifi->scan(NULL,0);

    if (count <= 0) {

        printf("scan() failed with return value: %d\n", count);

        return 0;

    }

    /\* Limit number of network arbitrary to 15 \*/

    count = count < 15 ? count : 15;

    ap = new WiFiAccessPoint[count];

    count = wifi->scan(ap, count);

    if (count <= 0) {

        printf("scan() failed with return value: %d\n", count);

        return 0;

    }

    for (int i = 0; i < count; i++) {

        printf("Network: %s secured: %s BSSID: %hhX:%hhX:%hhX:%hhx:%hhx:%hhx RSSI: %hhd Ch: %hhd\n", ap[i].get\_ssid(),

               sec2str(ap[i].get\_security()), ap[i].get\_bssid()[0], ap[i].get\_bssid()[1], ap[i].get\_bssid()[2],

               ap[i].get\_bssid()[3], ap[i].get\_bssid()[4], ap[i].get\_bssid()[5], ap[i].get\_rssi(), ap[i].get\_channel());

    }

    printf("%d networks available.\n", count);

    delete[] ap;

    return count;

}

void http\_demo(NetworkInterface \*net)

{

    // Open a socket on the network interface, and create a TCP connection to mbed.org

    TCPSocket socket;

    socket.open(net);

    SocketAddress a;

    net->gethostbyname("ifconfig.io", &a);

    a.set\_port(80);

    socket.connect(a);

    // Send a simple http request

    char sbuffer[] = "GET / HTTP/1.1\r\nHost: ifconfig.io\r\n\r\n";

    int scount = socket.send(sbuffer, sizeof sbuffer);

    printf("sent %d [%.\*s]\n", scount, strstr(sbuffer, "\r\n") - sbuffer, sbuffer);

    // Recieve a simple http response and print out the response line

    char rbuffer[64];

    int rcount = socket.recv(rbuffer, sizeof rbuffer);

    printf("recv %d [%.\*s]\n", rcount, strstr(rbuffer, "\r\n") - rbuffer, rbuffer);

    // Close the socket to return its memory and bring down the network interface

    socket.close();

}

void messageArrived(MQTT::MessageData& md)

{

    MQTT::Message &message = md.message;

    printf("Message arrived: qos %d, retained %d, dup %d, packetid %d\r\n", message.qos, message.retained, message.dup, message.id);

    printf("Payload %.\*s\r\n", message.payloadlen, (char\*)message.payload);

    ++arrivedcount;

}

void mqtt\_demo(NetworkInterface \*net)

{

    float version = 0.6;

    char\* topic = "mbed-sample";

    TCPSocket socket;

    MQTTClient client(&socket);

    SocketAddress a;

    char\* hostname = "192.168.1.66";

    net->gethostbyname(hostname, &a);

    int port = 1883;

    a.set\_port(port);

    printf("Connecting to %s:%d\r\n", hostname, port);

    socket.open(net);

    printf("Opened socket\n\r");

    int rc = socket.connect(a);

    if (rc != 0)

        printf("rc from TCP connect is %d\r\n", rc);

    printf("Connected socket\n\r");

    MQTTPacket\_connectData data = MQTTPacket\_connectData\_initializer;

    data.MQTTVersion = 3;

    data.clientID.cstring = "mbed-sample";

    data.username.cstring = "1234";

    data.password.cstring = "1234";

    if ((rc = client.connect(data)) != 0)

        printf("rc from MQTT connect is %d\r\n", rc);

    if ((rc = client.subscribe(topic, MQTT::QOS2, messageArrived)) != 0)

        printf("rc from MQTT subscribe is %d\r\n", rc);

    MQTT::Message message;

    // QoS 0

    char buf[100];

    sprintf(buf, "Hello World!  QoS 0 message from app version %f\r\n", version);

    message.qos = MQTT::QOS0;

    message.retained = false;

    message.dup = false;

    message.payload = (void\*)buf;

    message.payloadlen = strlen(buf)+1;

    rc = client.publish(topic, message);

    while (arrivedcount < 1)

        client.yield(100);

    // QoS 1

    sprintf(buf, "Hello World!  QoS 1 message from app version %f\r\n", version);

    message.qos = MQTT::QOS1;

    message.payloadlen = strlen(buf)+1;

    rc = client.publish(topic, message);

    while (arrivedcount < 2)

        client.yield(100);

    while (arrivedcount < 3)

        client.yield(100);

    if ((rc = client.unsubscribe(topic)) != 0)

        printf("rc from unsubscribe was %d\r\n", rc);

    if ((rc = client.disconnect()) != 0)

        printf("rc from disconnect was %d\r\n", rc);

    socket.close();

    printf("Version %.2f: finish %d msgs\r\n", version, arrivedcount);

    return;

}

int main()

{

    printf("WiFi example\n");

#ifdef MBED\_MAJOR\_VERSION

    printf("Mbed OS version %d.%d.%d\n\n", MBED\_MAJOR\_VERSION, MBED\_MINOR\_VERSION, MBED\_PATCH\_VERSION);

#endif

    wifi = WiFiInterface::get\_default\_instance();

    if (!wifi) {

        printf("ERROR: No WiFiInterface found.\n");

        return -1;

    }

    int count = scan\_demo(wifi);

    if (count == 0) {

        printf("No WIFI APs found - can't continue further.\n");

        return -1;

    }

    printf("\nConnecting to %s...\n", MBED\_CONF\_APP\_WIFI\_SSID);

    int ret = wifi->connect(MBED\_CONF\_APP\_WIFI\_SSID, MBED\_CONF\_APP\_WIFI\_PASSWORD, NSAPI\_SECURITY\_WPA\_WPA2);

    if (ret != 0) {

        printf("\nConnection error: %d\n", ret);

        return -1;

    }

    printf("Success\n\n");

    printf("MAC: %s\n", wifi->get\_mac\_address());

    printf("IP: %s\n", wifi->get\_ip\_address());

    printf("Netmask: %s\n", wifi->get\_netmask());

    printf("Gateway: %s\n", wifi->get\_gateway());

    printf("RSSI: %d\n\n", wifi->get\_rssi());

    http\_demo(wifi);

    mqtt\_demo(wifi);

    wifi->disconnect();

    printf("\nDone\n");

}