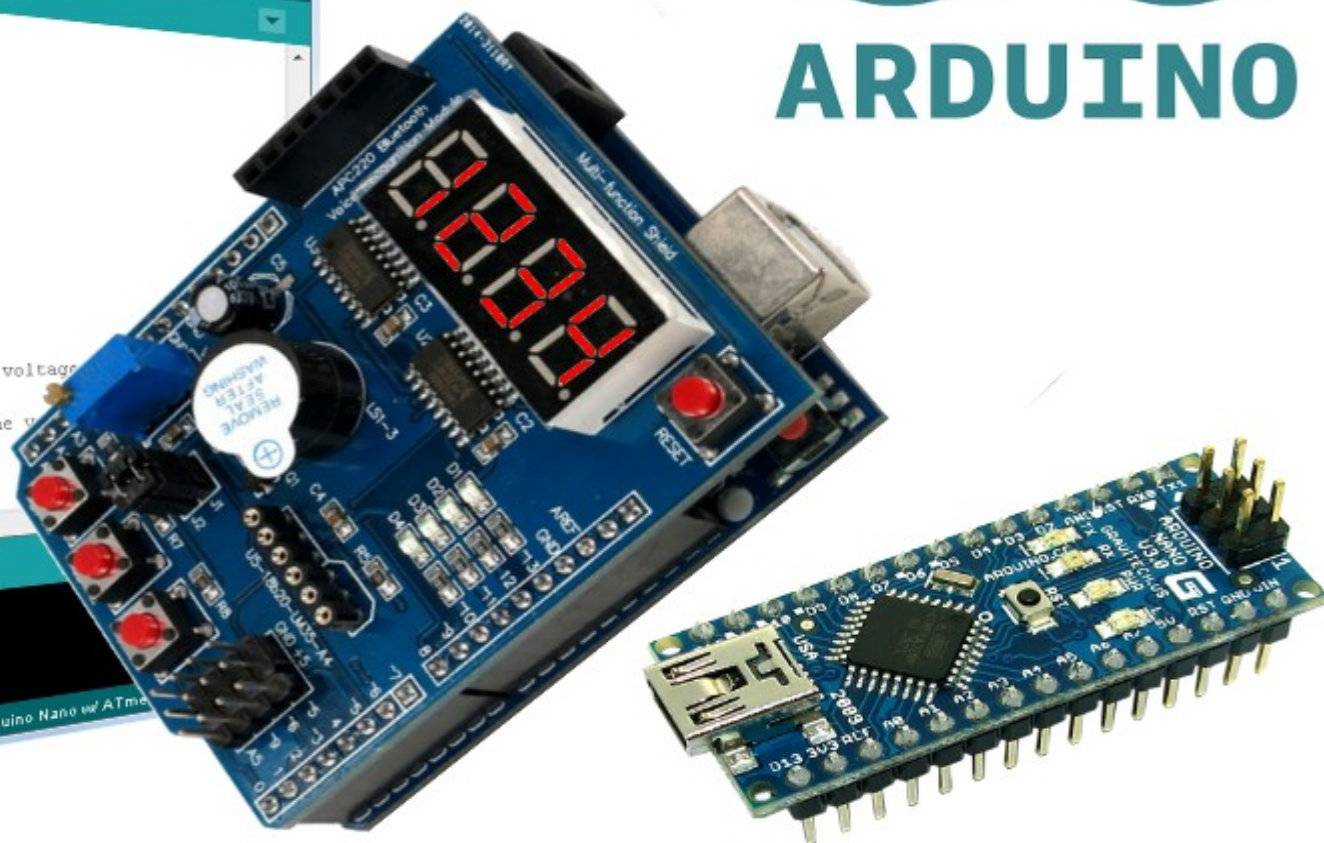
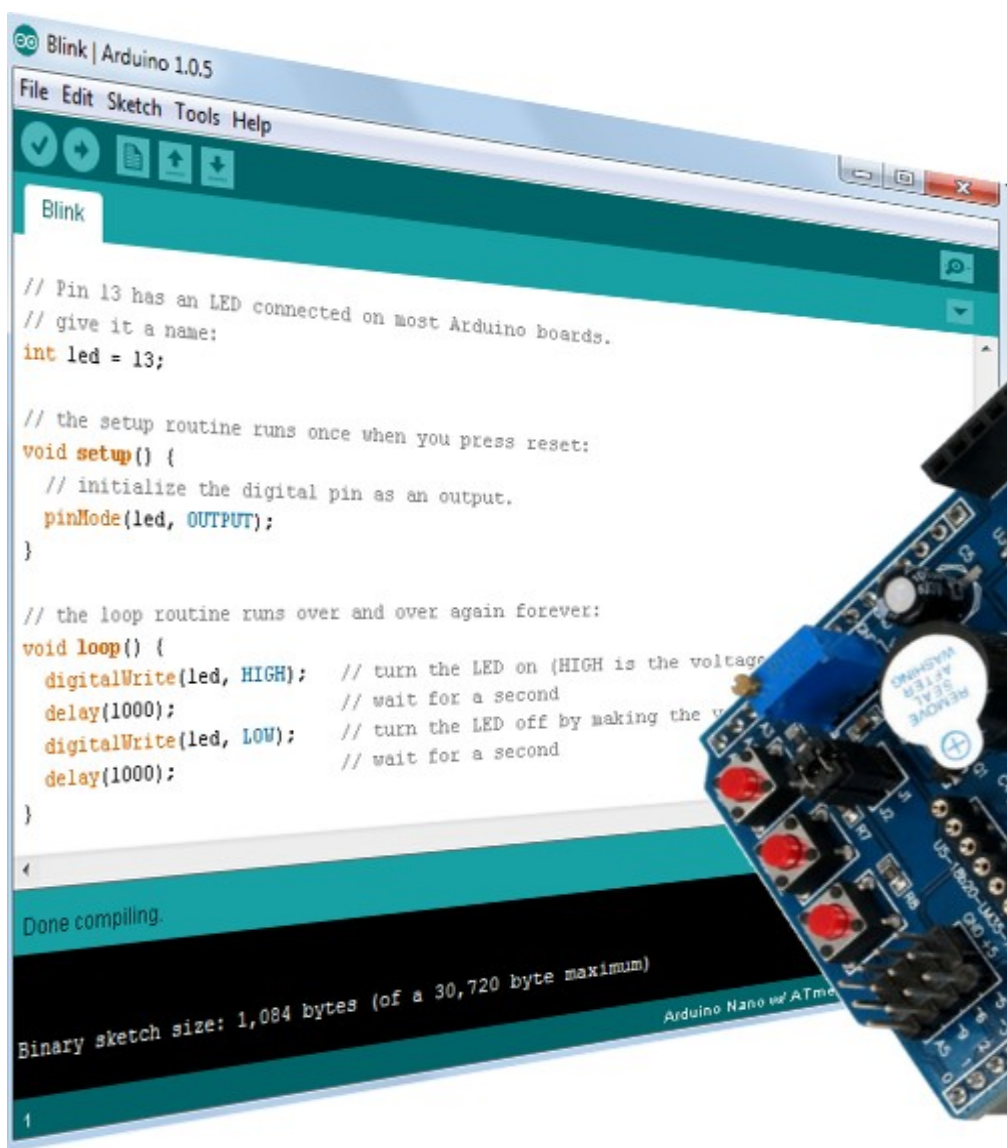


# Arduino tanfolyam kezdőknek és haladóknak



## 15. ESP8266 alapú MQTT szerver

# Felhasznált és ajánlott irodalom

---

- Leírások, dokumentáció:
  - ❖ HiveMQ : [MQTT Essentials](#)
  - ❖ OASIS: [MQTT Version 3.1.1 specification](#)
  - ❖ ESP8266 Community: [ESP8266 Arduino Core's documentation](#)
- Arduino programkönyvtárak:
  - ❖ Martin Ger: [uMQTTBroker library for ESP8266 Arduino](#)
  - ❖ Nick O'Leary: [PubSubClient - Arduino Client for MQTT](#)
  - ❖ Benoit Blanchon: [Mastering ArduinoJson 6](#)
- PC és mobil segédprogramok:
  - ❖ Thomas Nordquist: [MQTT Explorer](#)
  - ❖ Wireshark community: [Wireshark protocol analyzer](#)
  - ❖ Routix software: [MQTT Dash](#)

# A csatlakozások titkos adatai

- Az ESP8266 kliensként történő csatlakozásokhoz a személyes adatokat kiszerveztük egy **secrets.h** nevű fejléc állományba, amelyet a **Vázlatfüzet** (Sketchbook) mappa **libraries/secrets** almappájában helyeztünk el

```
#include <ESP8266WiFi.h>
#include "secrets.h"

void setup_wifi() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  Serial.print("Connecting to ");
  Serial.print(WIFI_SSID);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Connected! IP address: ");
  Serial.println(WiFi.localIP());
}
```

```
#define WIFI_SSID  "MY_SSID"
#define WIFI_PASS  "MY_PASSWORD"

String OPENWEATHERMAP_APPID =
"*****";

String THINGSPEAK_WRITE_APIKEY =
"xxxxxxxxxxxx";

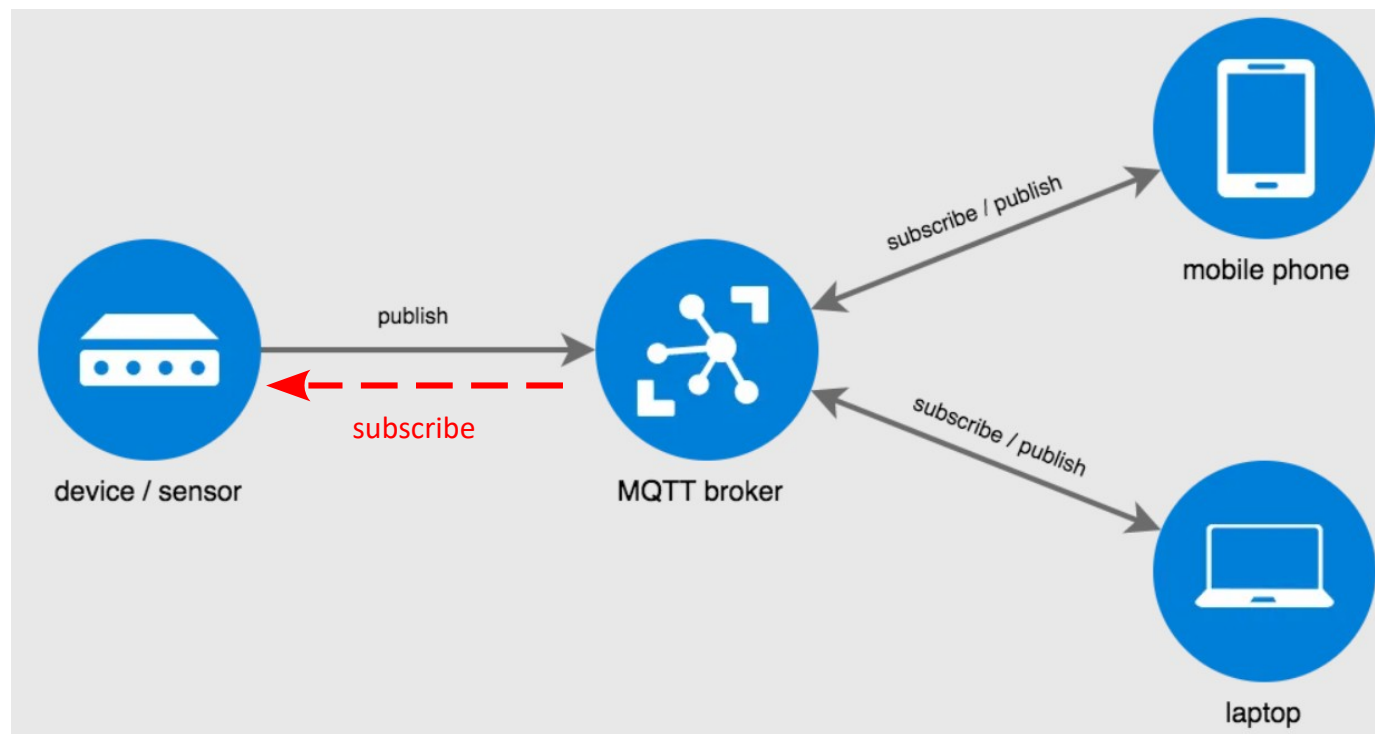
#define FLESPI_TOKEN
"Ws44Ev13*****rrQk0lbQGE1NuMoDnPNk1g"

#define HIVEMQ_USER "*****"
#define HIVEMQ_PASS "*****"

#define uMQTT_USER  "hobbi"
#define uMQTT_PASS  "megtestesules"
```

# MQTT alapok

- A **publish/subscribe** séma elkülöníti az adatküldő klienst (**publisher**) az üzeneteket fogadó kliens(ek)től (**subscriber**).
- A kapcsolatot egy harmadik komponens, az ún. bróker biztosítja
- Az **MQTT** kommunikáció **TCP/IP** alapú
- A **bróker** többnyire külső szolgáltató (pl. **HiveMQ**), vagy helyi, nagyobb teljesítményű eszköz (**PC**, esetleg **Raspberry Pi**)
- Most egy **ESP8266 NodeMCU** kártya felhasználásával készítünk brókert, (*olcsó és nem ráz...*), nyilván szerényebb igényeket célozva...



# Az uMQTTBroker programkönyvtár

---

- **Martin Ger** [uMQTTBroker](#) Arduino programkönyvtára lehetővé teszi, hogy egy **ESP8266** felhasználásával alakítsunk ki egy szerény képességű **MQTT brókert** a helyi hálózaton
- A bróker az alábbiakat támogatja:
  - ❖ MQTT v3.1 és v3.1.1 protokoll
  - ❖ több (~ 8) kliens egyidejű kapcsolódása
  - ❖ Megőrzött üzenetek
  - ❖ QoS level 0
  - ❖ username/password autentikáció
- A bróker az alábbiakat jelenleg nem támogatja:
  - ❖ QoS 1 és 2 szint
  - ❖ sok TCP(=MQTT) kliens
  - ❖ non-clear sessions
  - ❖ TLS

# uMQTTBroker C++ style API

```
class uMQTTBroker {
public: uMQTTBroker(uint16_t portno=1883, uint16_t max_subscriptions=30,
                 uint16_t max_retained_topics=30);

    void init();

//--- Callbacks on client actions -----
    virtual bool onConnect(IPAddress addr, uint16_t client_count);
    virtual void onDisconnect(IPAddress addr, String client_id);
    virtual bool onAuth(String username, String password, String client_id);
    virtual void onData(String topic, const char *data, uint32_t length);

//--- Infos on currently connected clients ----
    virtual uint16_t getClientCount();
    virtual bool getClientId(uint16_t index, String &client_id);
    virtual bool getClientAddr(uint16_t index, IPAddress& addr);

//--- Interaction with the local broker -----
    virtual bool publish(String topic, uint8_t* data, uint16_t data_length,
                        uint8_t qos=0, uint8_t retain=0);
    virtual bool publish(String topic, String data, uint8_t qos=0, uint8_t retain=0);
    virtual bool subscribe(String topic, uint8_t qos=0);
    virtual bool unsubscribe(String topic);

//--- Cleanup all clients on Wifi connection loss ---
    void cleanupClientConnections();
};
```

Jelszó  
ellenőrzése

Adat  
beérkezése

# ESP8266\_uMQTT\_Broker.ino

---

- Az első kísérlethez a [uMQTTBrokerSampleOOFull.ino](#) mintapéldából indulunk ki, amelyet kicsit átalakítottunk:
- A WiFi (STA) belépő kódot kiszerveztük a *secrets.h* állományba
- Az *onAuth()* visszahívási függvényben megfogalmazzuk a belépés érvényességének feltételét (sikeres belépéskor a függvénynek *true* értékkel kell visszatérnie), a belépés ellenőrzésére használt `uMQTT_USER` és `uMQTT_PASS` szintén a *secrets.h* állományban vannak definiálva
- Kísérleteinkben a routeren keresztül a helyi hálózatra kapcsolódtunk, ezért a STATION módot választottuk
- Az első próbák után a *loop()* függvényben célszerű kommentbe tenni a számláló publikálását
- Csak a kipróbálásnál célszerű használni a *printClients()* függvényt

# ESP8266\_uMQTT\_Broker.ino

```
#include <ESP8266WiFi.h>
#include "uMQTTBroker.h"
#include "secrets.h"
char ssid[] = WIFI_SSID;           //***** your network SSID (name)
char pass[] = WIFI_PASS;          //***** your network password
bool WiFiAP = false;              // Do yo want the ESP as AP?

void startWiFiClient() {
  Serial.println("Connecting to "+(String)ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: " + WiFi.localIP().toString());
}

void startWiFiAP() {
  WiFi.mode(WIFI_AP);
  WiFi.softAP(ssid, pass);
  Serial.println("AP started");
  Serial.println("IP address: " + WiFi.softAPIP().toString());
}
```

Kívánság szerint  
WiFi kliens (STA),  
vagy WiFi AP mód  
Inicializálása



# ESP8266\_uMQTT\_Broker.ino

```
class myMQTTBroker: public uMQTTBroker {
public:
    virtual bool onConnect(IPAddress addr, uint16_t client_count) {
        Serial.println(addr.toString()+" connected");
        return true;
    }

    virtual void onDisconnect(IPAddress addr, String client_id) {
        Serial.println(addr.toString()+" (" +client_id+") disconnected");
    }

    virtual bool onAuth(String username, String password, String client_id) {
        Serial.println("Username/Password/ClientId: "+username+"/"+password+"/"+client_id);
        return (username.equals(String(uMQTT_USER)) && password.equals(String(uMQTT_PASS)));
    }

    virtual void onData(String topic, const char *data, uint32_t length) {
        char data_str[length+1];
        os_memcpy(data_str, data, length);
        data_str[length] = '\0';
        Serial.println("received topic '"+topic+"' with data '"+(String)data_str+"'");
        // printClients();
    }

    // Sample for the usage of the client info methods
    virtual void printClients() { . . . }
};
```

# ESP8266\_uMQTT\_Broker.ino

```
myMQTTBroker myBroker;

void setup() {
  Serial.begin(115200);
  Serial.println();

  if (WiFiAP) startWiFiAP();           // Start WiFi
  else startWiFiClient();
  Serial.println("Starting MQTT broker"); // Start the broker

  myBroker.init();
  myBroker.subscribe("#");             // Subscribe to everything
}

int counter = 0;

void loop() {
  //--- Publish the counter value as String
  myBroker.publish("broker/counter", (String)counter++);
  delay(1000);                          // wait a second
}
```

# A bróker megérdemel egy fix IP címet!

- A bróker könnyű eléréséhez rendeljük fix IP címeket a helyi routeren a NodeMCU modulokhoz!

The image shows a TP-LINK router configuration page. On the left is a navigation menu with options like Status, Quick Setup, Network, Dual Band Selection, Wireless 2.4GHz, Wireless 5GHz, Guest Network, DHCP, - DHCP Settings, - DHCP Clients List, and - Address Reservation. The main content area is divided into two sections: 'DHCP Clients List' and 'Address Reservation'. The 'DHCP Clients List' table shows several clients, with 'ESP-DC2657' (labeled 'bróker') and 'ESP-1C5107' (labeled 'kliens') highlighted. The 'Address Reservation' table shows three reserved IP addresses, with arrows pointing from the 'bróker' and 'kliens' labels to the corresponding MAC addresses in this table.

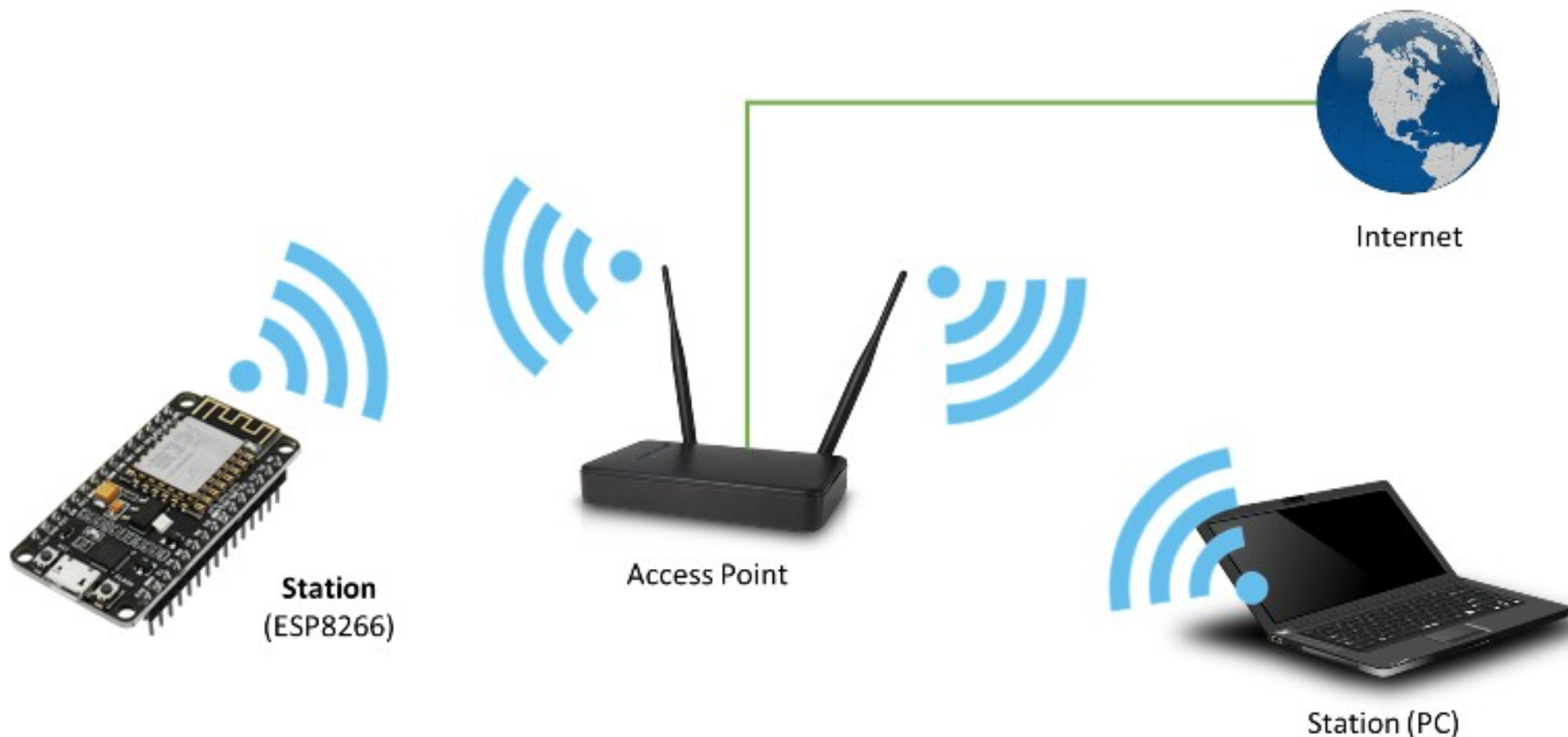
ID	Client Name	MAC Address	Assigned IP
1	Galaxy-Tab-S6-Lite	0A-93-5C-F7-BC-32	192.168.1.103
...			
6	<u>ESP-DC2657</u> <b>bróker</b>	18-FE-34-DC-26-57	192.168.1.104
7	NPIC6045D	B4-99-BA-C6-04-5D	192.168.1.200
8	<u>ESP-1C5107</u> <b>kliens</b>	A0-20-A6-1C-51-07	192.168.1.105

ID	MAC Address	Reserved IP Address
1	B4-99-BA-C6-04-5D	192.168.1.200
2	18-FE-34-DC-26-57	192.168.1.11
3	A0-20-A6-1C-51-07	192.168.1.12

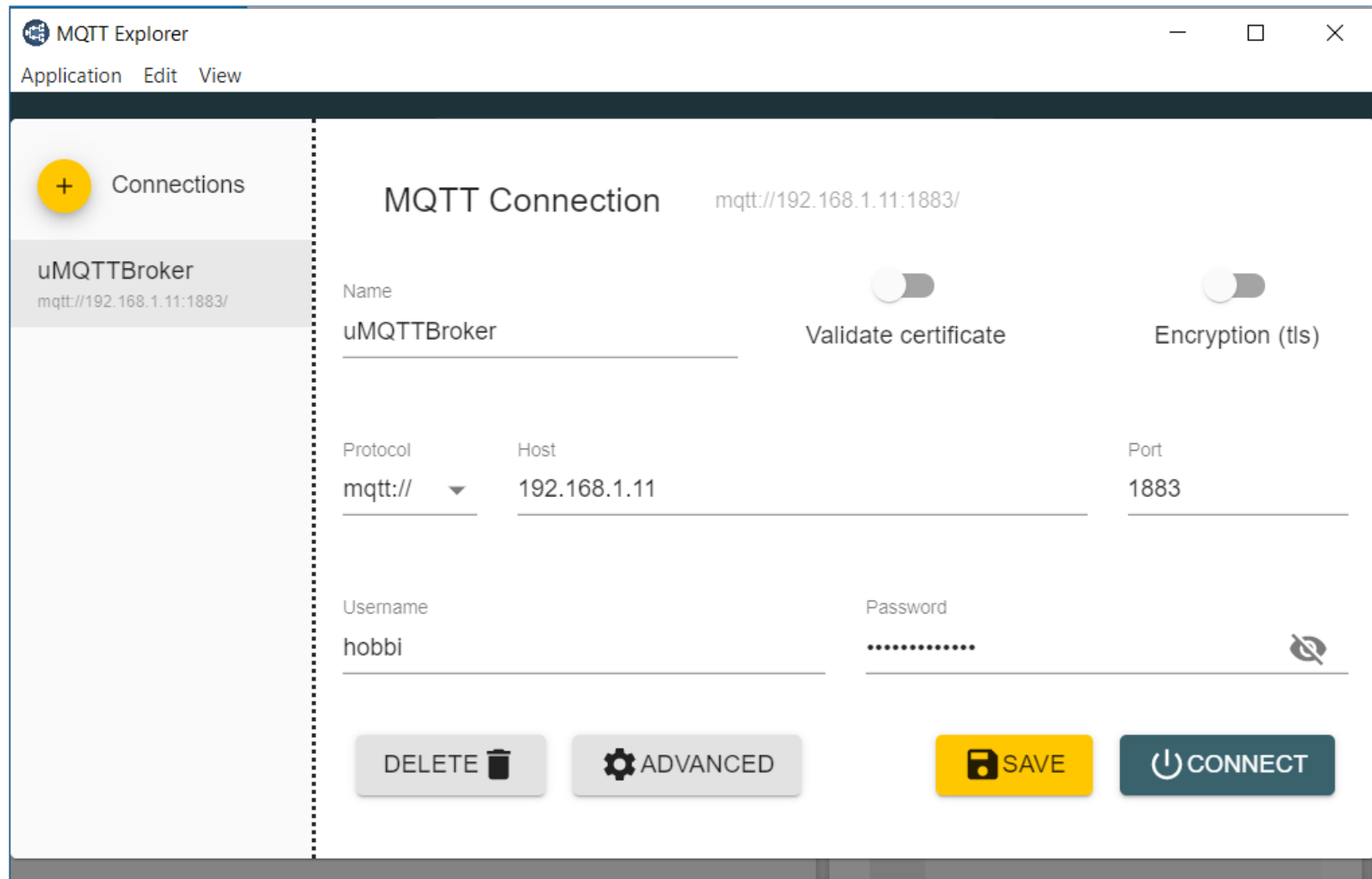
# Az MQTT üzenetek közelnézetből

- Az **ESP8266\_uMQTTBroker** működését az **MQTT Explorer** nevű számítógépes klienssel próbáljuk ki, s az üzenetcsomagokat a **Wireshark** program segítségével vizsgáljuk meg közelebbről



# MQTT Explorer (Windows alkalmazás)

- Kapcsolódjunk egy klienssel a helyi brókerhez!
- Az **Advanced** gombra kattintva iratkozzunk fel minden topikra (#)!



# A Connect esemény

Capturing from Wi-Fi (port 1883)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
9	6.621478	192.168.1.103	192.168.1.11	MQTT	112	Connect Command
10	6.643529	192.168.1.11	192.168.1.103	MQTT	58	Connect Ack
11	6.644784	192.168.1.103	192.168.1.11	MQTT	62	Subscribe Request (id=48053)
12	6.660680	192.168.1.11	192.168.1.103	MQTT	59	Subscribe Ack (id=48053)

> Frame 9: 112 bytes on wire (896 bits), 112 bytes captured (896 bits) on interface \Device\NPF\_{ACC1DC12...}

> Ethernet II, Src: LiteonTe\_46:45:bd (94:08:53:46:45:bd), Dst: Espressi\_dc:26:57 (18:fe:34:dc:26:57)

> Internet Protocol Version 4, Src: 192.168.1.103, Dst: 192.168.1.11

> Transmission Control Protocol, Src Port: 52071, Dst Port: 1883, Seq: 1, Ack: 1, Len: 58

> MQ Telemetry Transport Protocol, Connect Command

```
0000  18 fe 34 dc 26 57 94 08 53 46 45 bd 08 00 45 00  ..4.&W..SFE...E.
0010  00 62 8c 3f 40 00 80 06 ea 93 c0 a8 01 67 c0 a8  .b.?@... ..g..
0020  01 0b cb 67 07 5b 7b 83 ff 54 00 02 17 f8 50 18  ...g.[{. .T...P.
0030  fa f0 3e 8e 00 00 10 38 00 04 4d 51 54 54 04 c2  ..>...8 ..MQTT..
0040  00 3c 00 16 6d 71 74 74 2d 65 78 70 6c 6f 72 65  .<..mqtt -explore
0050  72 2d 65 30 30 64 36 63 62 37 00 05 68 6f 62 62  r-e00d6c b7..hobb
0060  69 00 0d 6d 65 67 74 65 73 74 65 73 75 6c 65 73  i..megte stesules
```

MQ Telemetry Transport Protocol (mqtt), 58 byte(s) | Packets: 31 · Displayed: 31 (100.0%) | Profile: Default

# A Connect esemény

- A Wireshark hajlandó számunkra dekódolni az üzenetet, így bájtokra lebontva és szövegesen is láthatjuk az üzenetet

The screenshot displays the details of an MQTT Connect Command in Wireshark. The details pane shows the following information:

- MQ Telemetry Transport Protocol, Connect Command
- Header Flags: 0x10, Message Type: Connect Command
- Msg Len: 56
- Protocol Name Length: 4
- Protocol Name: MQTT
- Version: MQTT v3.1.1 (4)
- Connect Flags: 0xc2, User Name Flag, Password Flag, QoS Level: At most once delivery (Fire and Forge)
- Keep Alive: 60
- Client ID Length: 22
- Client ID: mqtt-explorer-e00d6cb7
- User Name Length: 5
- User Name: hobbi
- Password Length: 13
- Password: megtestesules

The hex dump pane shows the raw data of the message, with the ASCII column displaying the decoded text:

```
0000 18 fe 34 dc 26 57 94 08 53 46 45 bd 08 00 45 00  ..4-&W.. SFE...E-
0010 00 62 8c 3f 40 00 80 06 ea 93 c0 a8 01 67 c0 a8  -b·?@... ..g..
0020 01 0b cb 67 07 5b 7b 83 ff 54 00 02 17 f8 50 18  ...g·[{- ·T....P·
0030 fa f0 3e 8e 00 00 10 38 00 04 d4 51 54 54 04 c2  ..>...·8 ..MQTT..
0040 00 3c 00 16 6d 71 74 74 2d 65 78 70 6c 6f 72 65  ·<..mqtt -explore
0050 72 2d 65 30 30 64 36 63 62 37 00 05 68 6f 62 62  r-e00d6c b7··hobb
0060 69 00 0d 6d 65 67 74 65 73 74 65 73 75 6c 65 73  i··megte stesules
```

MQ Telemetry Transport Protocol (mqtt), 58 byte(s) | Packets: 43 · Displayed: 43 (100.0%) | Profile: Default

# Subscribe kérés

- Az alábbiakban a beérkezett Subscribe Request látható – melyben feliratkozunk mindenre IS...

No.	Time	Source	Destination	Protocol	Length	Info
9	6.621478	192.168.1.103	192.168.1.11	MQTT	112	Connect Command
10	6.643529	192.168.1.11	192.168.1.103	MQTT	58	Connect Ack
11	6.644784	192.168.1.103	192.168.1.11	MQTT	62	Subscribe Request (id=48053) [#]
12	6.660680	192.168.1.11	192.168.1.103	MQTT	59	Subscribe Ack (id=48053)

> Frame 11: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface \Device\NPF\_{ACC1DC12-A348-44E4-A...}

> Ethernet II, Src: LiteonTe\_46:45:bd (94:08:53:46:45:bd), Dst: Espressi\_dc:26:57 (18:fe:34:dc:26:57)

> Internet Protocol Version 4, Src: 192.168.1.103, Dst: 192.168.1.11

> Transmission Control Protocol, Src Port: 52071, Dst Port: 1883, Seq: 59, Ack: 5, Len: 8

MQ Telemetry Transport Protocol, Subscribe Request

- > Header Flags: 0x82, Message Type: Subscribe Request
- Msg Len: 6
- Message Identifier: 48053
- Topic Length: 1
- Topic: #
- Requested QoS: At most once delivery (Fire and Forget) (0)

MQ Telemetry Transport Protocol, Subscribe Ack

- > Header Flags: 0x90, Message Type: Subscribe Ack
- Msg Len: 3
- Message Identifier: 48053
- Granted QoS: At most once delivery (Fire and Forget) (0)

90 03 bb b5 00

0000	18 fe 34 dc 26 57 94 08 53 46 45 bd 08 00 45 00	..4.&W.. SFE...E.
0010	00 30 8c 40 40 00 80 06 ea c4 c0 a8 01 67 c0 a8	..0.@@... ..g..
0020	01 0b cb 67 07 5b 7b 83 ff 8e 00 02 17 fc 50 18	...g.[{- .....P.
0030	fa ec 6a 84 00 00 82 06 bb b5 00 01 23 00	..j... ..#.

A nyugtázás



# Adatküldés (Publish esemény)

The screenshot shows the MQTT Explorer interface. On the left, a tree view shows the hierarchy: 192.168.1.11 > cspista > home > temp = 24.7, humidity = 26. A red text overlay reads: "Publikáljunk adatot a 'cspista/home/temp' témakörbe, az adat legyen '24.7'". A red arrow points from this text to the 'Publish' panel on the right. The 'Publish' panel is highlighted with a red border and contains the following elements: a 'Topic' field with 'cspista/home/temp', a 'Value' field with '24.7', and a 'PUBLISH' button. Below the 'Publish' panel, two line graphs are visible: 'cspista/home/humidity' and 'cspista/home/temp'. The 'cspista/home/temp' graph shows a yellow line fluctuating around 26, with a red circle highlighting a data point at approximately 24.7.

# Publish esemény

Capturing from Wi-Fi (port 1883)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.103	192.168.1.11	MQTT	79	Publish Message [cspista/home/temp]
2	0.074443	192.168.1.11	192.168.1.103	MQTT	79	Publish Message [cspista/home/temp]
3	0.118420	192.168.1.103	192.168.1.11	TCP	54	59446 → 1883 [ACK] Seq=26 Ack=26 Wi

> Frame 1: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface \Device\NPF\_{ACC1DC12-A348-44E4-AC  
 > Ethernet II, Src: LiteonTe\_46:45:bd (94:08:53:46:45:bd), Dst: Espressi\_dc:26:57 (18:fe:34:dc:26:57)  
 > Internet Protocol Version 4, Src: 192.168.1.103, Dst: 192.168.1.11  
 > Transmission Control Protocol, Src Port: 59446, Dst Port: 1883, Seq: 1, Ack: 1, Len: 25  
 √ MQ Telemetry Transport Protocol, Publish Message  
 > [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet?)]  
 > Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)  
 Msg Len: 23  
 Topic Length: 17  
 Topic: cspista/home/temp  
 Message: 32342e37

```

0000  18 fe 34 dc 26 57 94 08 53 46 45 bd 08 00 45 00  ..4.&W.. SFE...E.
0010  00 41 c8 ad 40 00 80 06 ae 46 c0 a8 01 67 c0 a8  .A..@... .F...g..
0020  01 0b e8 36 07 5b 45 c1 d1 df 00 00 61 8d 50 18  ...6.[E. ....a.P.
0030  f6 e3 ad 8c 00 00 30 17 00 11 63 73 70 69 73 74  .....0. ..cspist
0040  61 2f 68 6f 6d 65 2f 74 65 6d 70 32 34 2e 37  a/home/t emp24.7
  
```

# A publish üzenet formátuma (QoS=0)

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (3)			DUP flag		QoS level		RETAIN
	0	0	1	1	X	X	X	X
byte 2	Remaining Length							

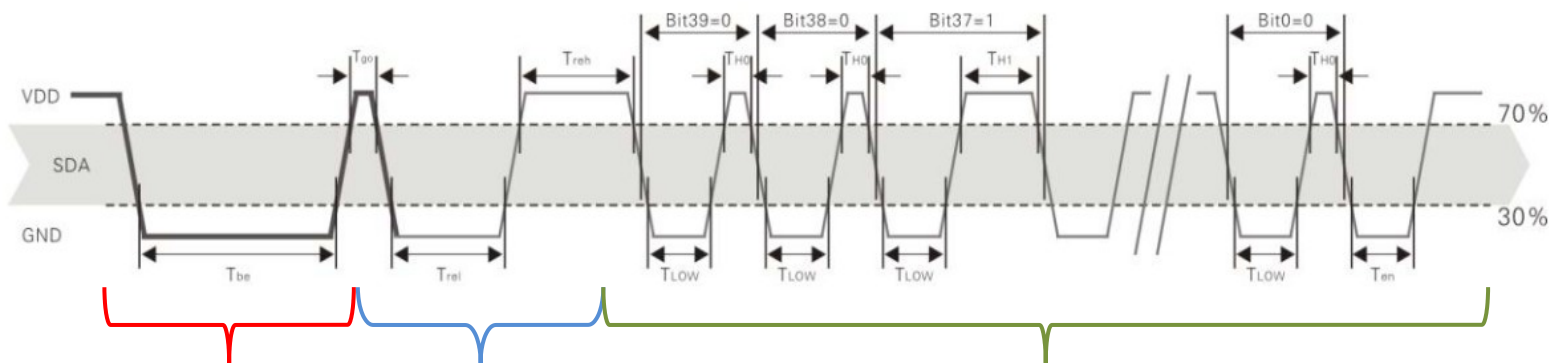
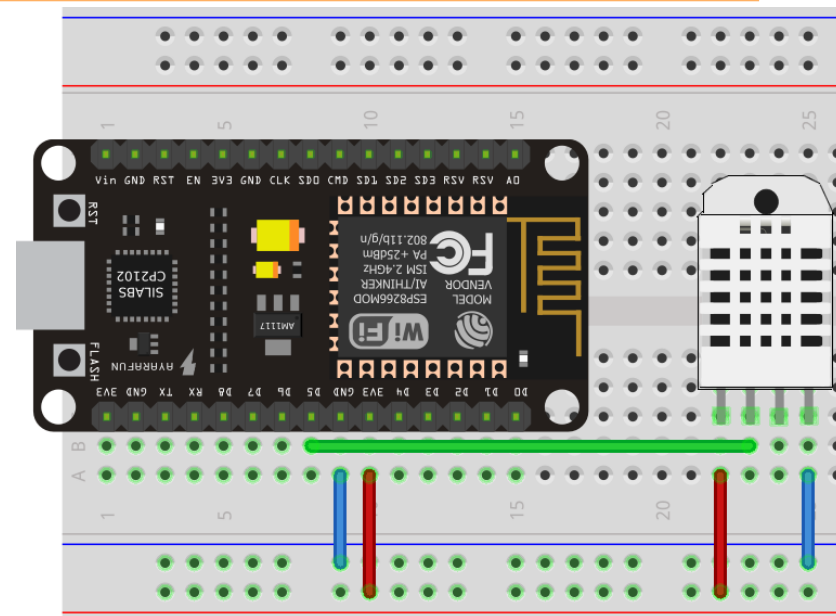
	Description	7	6	5	4	3	2	1	0
Topic Name									

byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
byte 3	'a' (0x61)	0	1	1	0	0	0	0	1
byte 4	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 5	'b' (0x62)	0	1	1	0	0	0	1	0

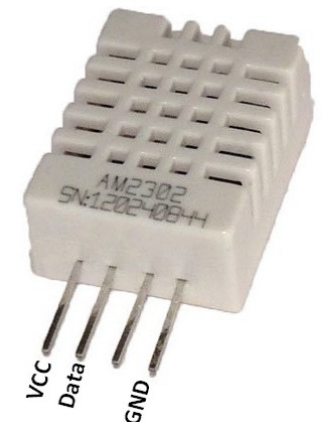
The Payload contains the Application Message that is being published. The content and format of the data is application specific. The length of the payload can be calculated by subtracting the length of the variable header from the Remaining Length field that is in the Fixed Header. It is valid for a PUBLISH Packet to contain a zero length payload.

# ESP8266\_uMQTT\_DHT22.ino

- Az előző előadásban bemutatott `ESP8266_HiveMQ_DHT22` kliens alkalmazást módosítsuk úgy, hogy a helyi brókerhez csatlakozzon!
- **Emlékeztető:** DHT22 szenzorral hőmérsékletet és rel. páratartalmat mérünk
- Programkönyvtár: [Adafruit DHT](#)
- Adatlap: [sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf](http://sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf)



Host indítójel      Szenzor nyugtázó jel      40 bitnyi adat  
Összesen tehát 85 időzítést tartalmaz egy-egy tranzakció ...



# Emlékeztető: PubSubClient programkönyvtár

- **PubSubClient** – kliens programkönyvtár MQTT kommunikációra
- **PubSubClient** (*server*, *port*, [*callback*], *client*, [*stream*]) – constructor  
*server* – pl. mqtt.flespi.io vagy más bróker, *port* – általában 1883  
*client* – egy WiFiClient objektum  
**Megjegyzés:** *server* és *port* a **setServer()** függvénnyel is megadhatók!
- **connect** (*clientId*, [*username*, *password*], [*willTopic*, *willQoS*, *willRetain*, *willMessage*], [*cleanSession*]) – kapcsolódása bróker szerverhez  
*clientId* - egy egyedi azonosító
- **setCallback** (*callback*) – visszahívási függvény regisztrálása  
`void callback(const char[] topic, byte* payload, unsigned int length);`
- **publish**(*topic*, *payload*, [*length*], [*retained*]) – adatküldés a brókernek  
*topic* – az adott témakör, *payload* – a témakörben küldött új adat
- **subscribe** (*topic*, [*qos*]) – feliratkozás a megadott témakörre  
*topic* – az adott témakör, *qos* – csak 0 vagy 1 lehet (Qos2 nem támogatott)
- **loop()** – rendszeresen meg kell hívni a beérkező üzenetek fogadásához

PubSubClient API dokumentáció: <https://pubsubclient.knolleary.net/api>

# ESP8266\_uMQTT\_DHT22.ino

---

- A mért adatok publikálásához két témakör használunk:
  - ❖ `cspista/home/temp`
  - ❖ `cspista/home/humidity`
- Az adatoknak csak a mérőszámát írjuk ki, hogy numerikus adatként lehessen kezelni (egyébként °C és % lenne a „mértékegység”)
- Kapcsolódáskor vagy újrakapcsolódáskor a `cspista/home` témakörbe küldünk egy `"DHT22 connected"` üzenetet
- A feliratkozásra (és a `GPIO2`-re kötött LED vezérlésére) használt témakör neve:
  - ❖ `cspista/home/led`

# ESP8266\_uMQTT\_DHT22.ino

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
#include "secrets.h"
const char* mqtt_server = "broker.hivemq.com";
#define DHTPIN          14           // GPIO14 (D5) pin for DHT sensor
DHT dht(DHTPIN, DHT22);
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
char msg[50];
int value = 0;
int nClient = 0;

void setup() {
  pinMode(BUILTIN_LED, OUTPUT); // Initialize the BUILTIN_LED pin as an output
  dht.begin();
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void callback(char* topic, byte* payload, unsigned int length) {
  if ((char)payload[0] == '1') {
    digitalWrite(BUILTIN_LED, LOW); // Turn the LED on
  } else digitalWrite(BUILTIN_LED, HIGH); // Turn the LED off
}
```

Lecseréljük erre: 192.168.1.11

```
void setup_wifi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
}
```

# ESP8266\_uMQTT\_DHT22.ino

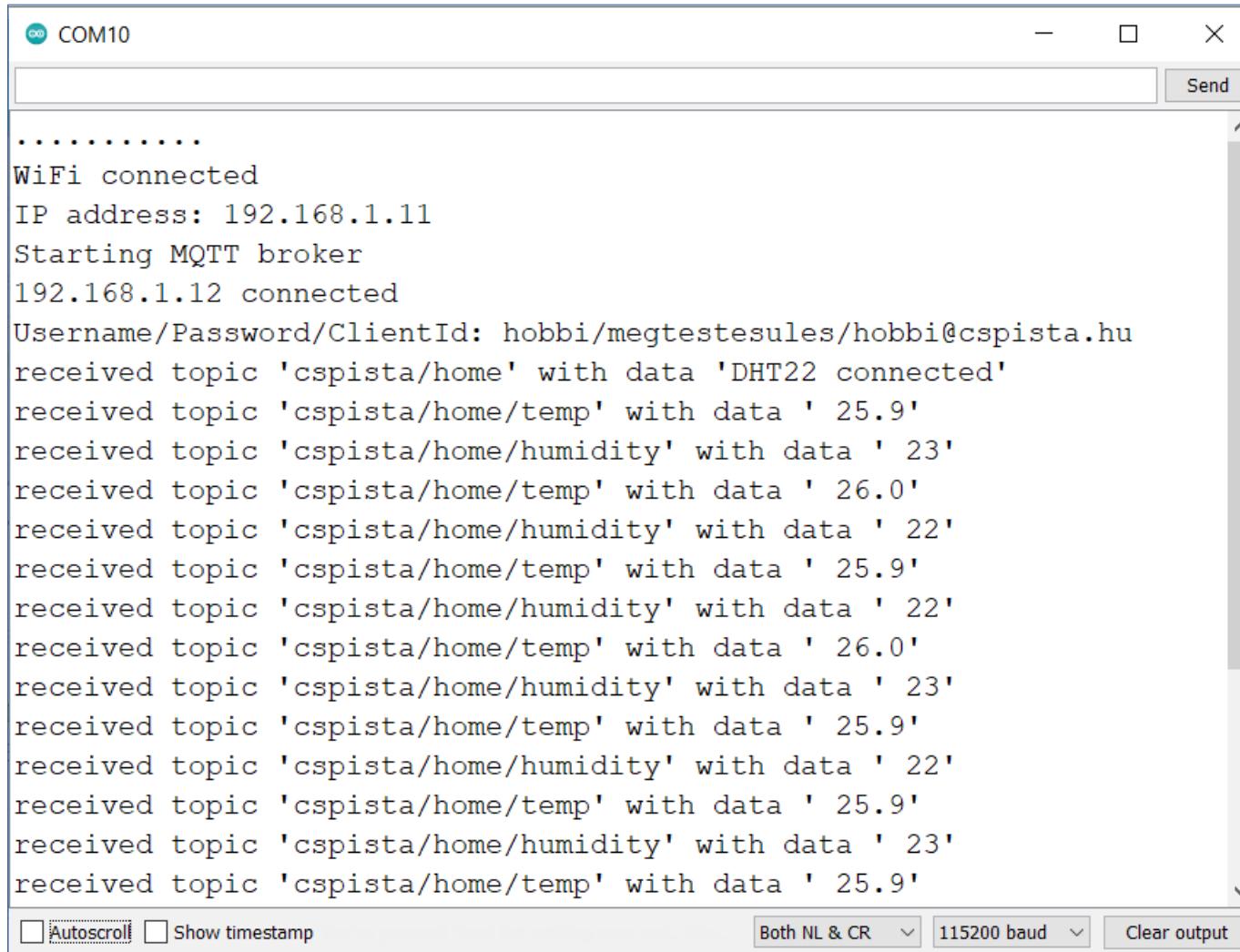
```
void reconnect() {  
  while (!client.connected()) {  
    char clientId[] = "hobbi@cspista.hu";  
    if (client.connect(clientId, uMQTT_USER, uMQTT_PASS )) { // Attempt to connect  
      client.publish("cspista/home", "DHT22 connected"); // Once connected, publish...  
      client.subscribe("cspista/home/led"); // ... and resubscribe  
    } else delay(5000); // Wait 5 seconds before retrying  
  }  
}  
  
void loop() {  
  if (!client.connected()) reconnect();  
  client.loop();  
  unsigned long now = millis();  
  if (now - lastMsg > 5000) {  
    lastMsg = now;  
    float t = dht.readTemperature();  
    if (!isnan(t)) {  
      sprintf(msg, "%5.1f", t);  
      client.publish("cspista/home/temp", msg);  
    }  
    float h = dht.readHumidity();  
    if (!isnan(h)) {  
      sprintf(msg, "%3.0f", h);  
      client.publish("cspista/home/humidity", msg);  
    }  
  }  
}
```





# ESP8266\_uMQTT\_Broker.ino

- A bróker és a kliens együttes futtatásakor ezt látjuk a bróker terminál kimenetén



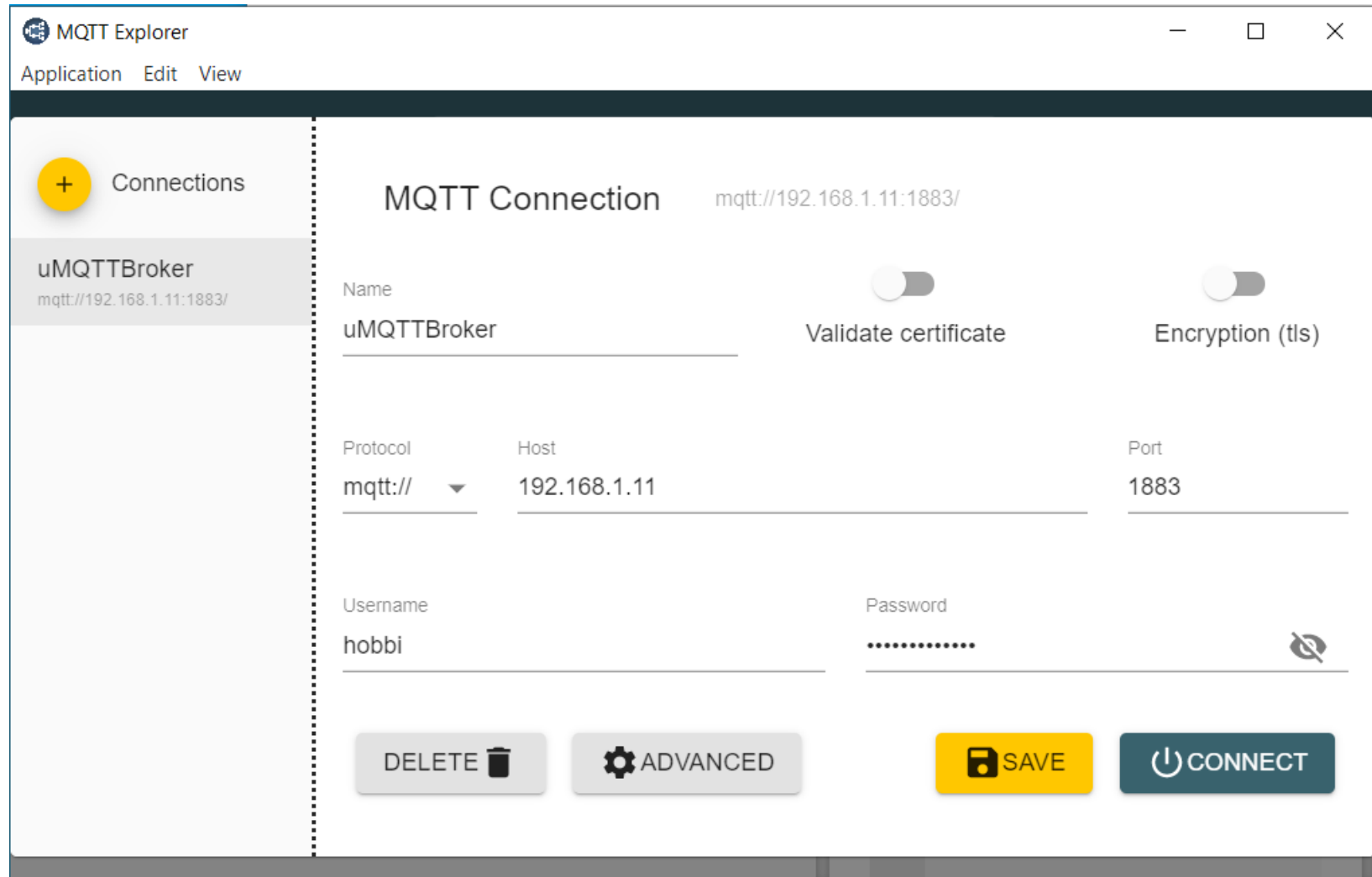
The screenshot shows a serial terminal window titled 'COM10'. The output text is as follows:

```
.....  
WiFi connected  
IP address: 192.168.1.11  
Starting MQTT broker  
192.168.1.12 connected  
Username/Password/ClientId: hobbi/megtestesules/hobbi@cspista.hu  
received topic 'cspista/home' with data 'DHT22 connected'  
received topic 'cspista/home/temp' with data ' 25.9'  
received topic 'cspista/home/humidity' with data ' 23'  
received topic 'cspista/home/temp' with data ' 26.0'  
received topic 'cspista/home/humidity' with data ' 22'  
received topic 'cspista/home/temp' with data ' 25.9'  
received topic 'cspista/home/humidity' with data ' 22'  
received topic 'cspista/home/temp' with data ' 26.0'  
received topic 'cspista/home/humidity' with data ' 23'  
received topic 'cspista/home/temp' with data ' 25.9'  
received topic 'cspista/home/humidity' with data ' 22'  
received topic 'cspista/home/temp' with data ' 25.9'  
received topic 'cspista/home/humidity' with data ' 23'  
received topic 'cspista/home/temp' with data ' 25.9'
```

At the bottom of the terminal window, there are several controls: a checkbox for 'Autoscroll' (checked), a checkbox for 'Show timestamp' (unchecked), a dropdown menu for 'Both NL & CR' (selected), a dropdown menu for '115200 baud' (selected), and a 'Clear output' button.

# MQTT Explorer (Windows alkalmazás)

- Kapcsolódjunk egy újabb klienssel a helyi brókerhez!
- Az **Advanced** gombra kattintva iratkozzunk fel minden topikra (#)!





Search...



DISCONNECT



▼ 192.168.1.11

▼ cspista

▼ home = connected

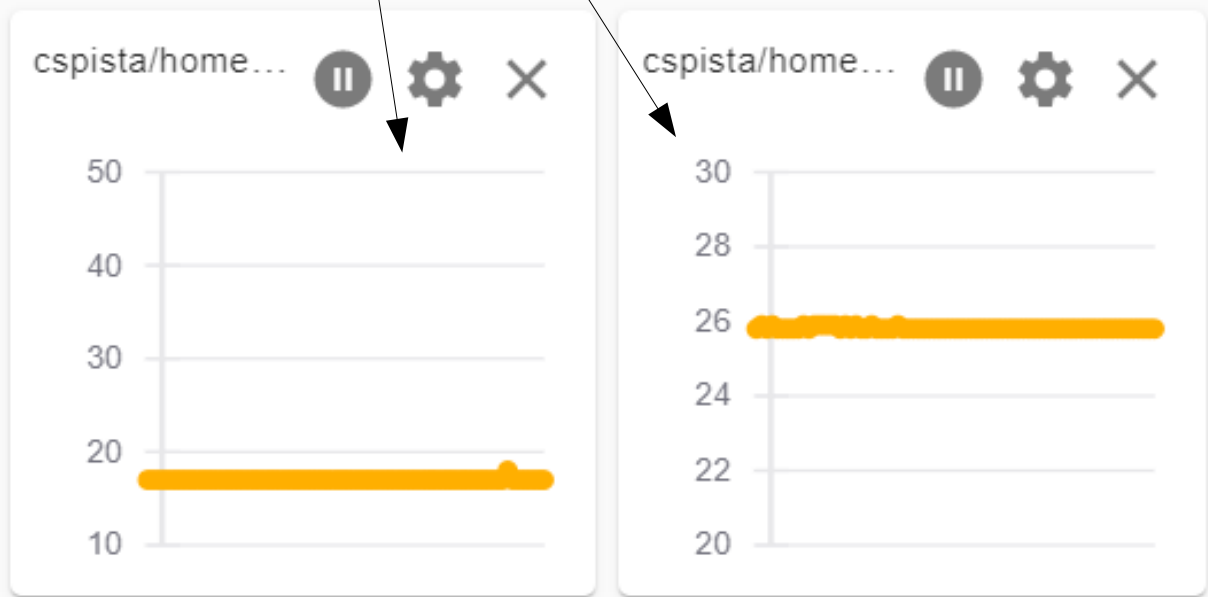
temp = 25.8

humidity = 17

led = 1

▼ SYS

uptime = 1245



Publish

Topic

cspista/home

raw

xml

json



PUBLISH

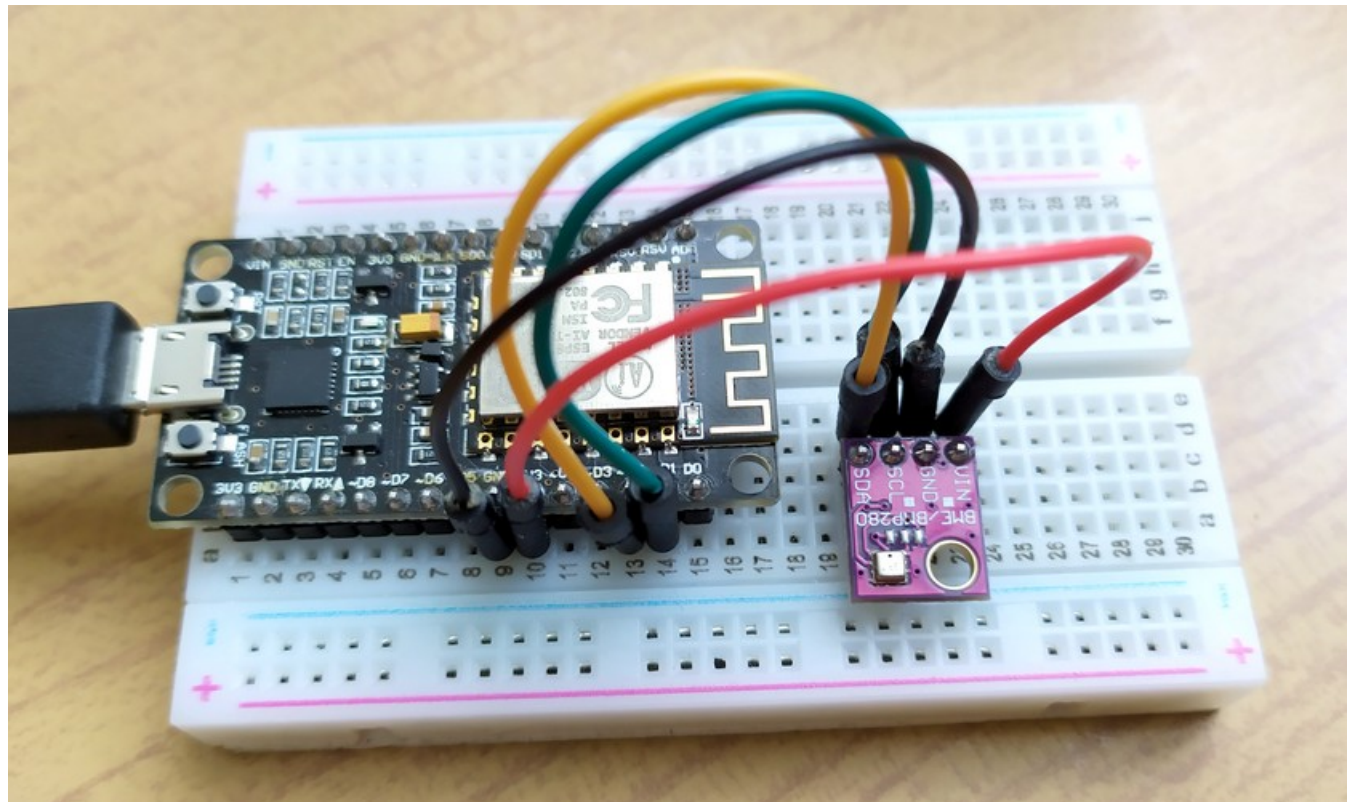
1245

# Adatküldés JSON formátumban

- Kibővítjük a rendszert egy **BME280** szenzorral (hőmérséklet, nyomás és relatív páratartalom mérése), s a mérési adatokat egyetlen topikban, JSON formátumba csomagolva publikáljuk
- Technikai okokból nem új klienst vetünk be, hanem a bróker szoftverét bővítjük ki **mérés** és **publish** funkciókkal

- Bekötés:

- ❖ SDA – GPIO4 (D2)
- ❖ SCL – GPIO5 (D1)
- ❖ GND – GND
- ❖ VIN – 3V3



# ESP8266\_uMQTT\_Broker\_BMP280.ino

```
#include <ESP8266WiFi.h>
#include "uMQTTBroker.h"
#include "secrets.h"

#include <ArduinoJson.h>
StaticJsonDocument<200> doc;

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#define SEALEVELPRESSURE_HPA (1013.25)
Adafruit_BME280 bme;

bool WiFiAP = false; // Do yo want the ESP as AP?
myMQTTBroker myBroker; // myMQTTBroker definiója a 9. oldalon!

void setup() {
  Serial.begin(115200);
  Serial.println();
  bme.begin(0x76);
  if (WiFiAP) startWiFiAP(); // Start WiFi
  else startWiFiClient();
  Serial.println("Starting MQTT broker"); // Start the broker
  myBroker.init();
  myBroker.subscribe("#"); // Subscribe to everything
}
```

# ESP8266\_uMQTT\_Broker.ino

```
void loop() {
  //--- Publish the counter value as String -----
  // myBroker.publish("broker / counter", (String)counter++);
  //--- wait a second
  delay(1000);
  //--- BME280 sensor data -----
  doc["temperature"] = String(bme.readTemperature(),1);
  doc["humidity"] = (int)bme.readHumidity();
  doc["pressure"] = String(((bme.readPressure() + 1440) / 100.0F),1);
  String payload = doc.as<String>(); // Serialization
  myBroker.publish("cspista/home/bme280", payload);
  delay(4000);
}
```

- Arra ügyeljünk, hogy a **serializeJson()** adatfolyamként kezeli a *String* objektumokat, azaz hozzáfűz... Például:

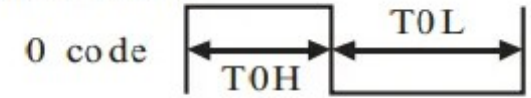
```
String output = "JSON = ";
serializeJson(doc, output);
```

- Esetünkben most egyszerűbbnek tűnt a típuskonverzió, de nem ez az egyetlen út

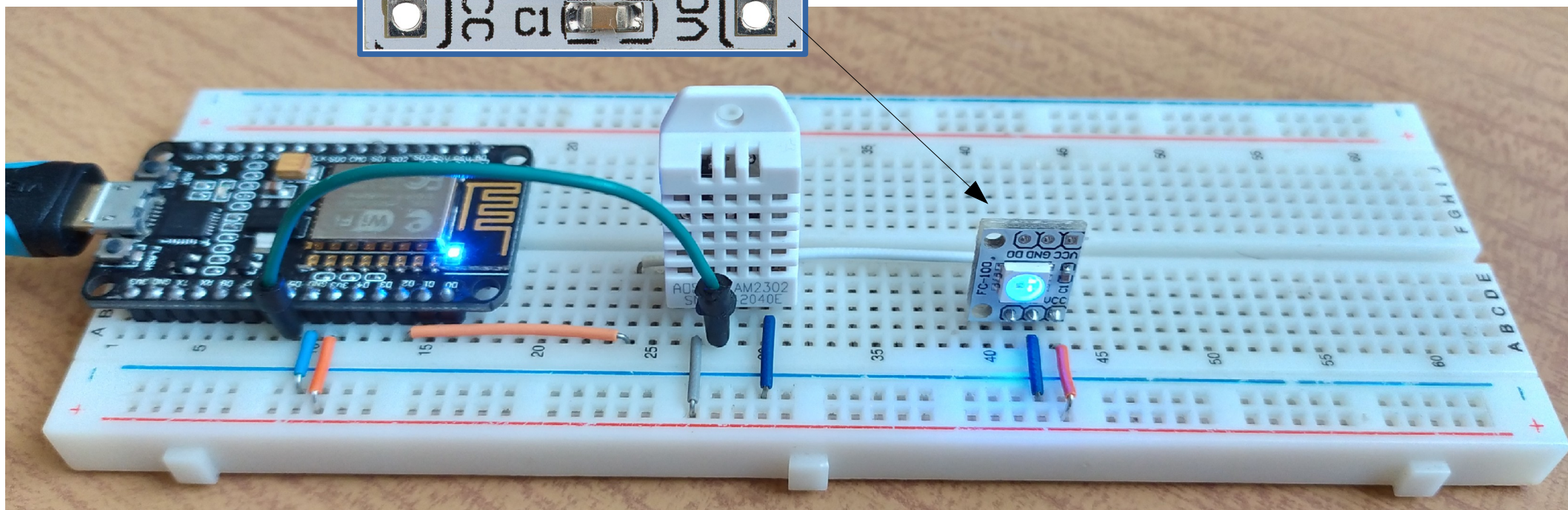
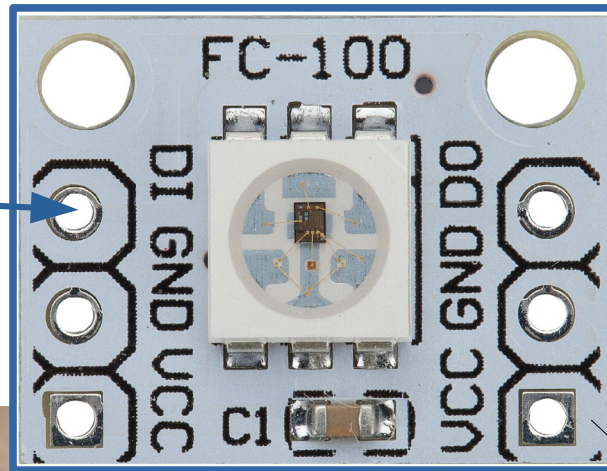
# Bővítsük a kliensünket is: Neopixel

- A DHT22-vel ellátott MQTT klienst bővítjük egy **WS2812** (Neopixel) digitálisan vezérelhető RGB LED-del!

Sequence chart:



GPIO4 (D2)



# ESP8266\_uMQTT\_neopixel.ino

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Adafruit_NeoPixel.h>
#include "DHT.h"
#include "secrets.h"
const char* mqtt_server = "192.168.1.11";
#define DHTPIN          14      // GPIO14 (D5) pin for DHT sensor
#define NEOPIXEL_PIN    4       // GPIO4 (D2) pin for Neopixel
#define NUMPIXELS       1       // NeoPixel string size (csak 1 db-ot használunk!)
Adafruit_NeoPixel pixels(NUMPIXELS, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);

DHT dht(DHTPIN, DHT22);
WiFiClient espClient;
PubSubClient client(espClient);

unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;
int nClient = 0;

String led_topic = "cspista/home/led";           // Így könnyebb lesz összehasonlítani!
String rgb_topic = "cspista/home/rgb";
```



# ESP8266\_uMQTT\_neopixel.ino

```
void setup() {
  pinMode(BUILTIN_LED, OUTPUT); // Initialize the BUILTIN_LED pin as an output
  Serial.begin(115200);
  pixels.begin(); // INITIALIZE NeoPixel strip object
  pixels.clear(); // Set all pixel colors to 'off'
  pixels.setPixelColor(0, pixels.Color(0, 150, 0));
  pixels.show(); // Send the updated pixel colors to the hardware.
  dht.begin(); // Initialize the DHT22 sensor
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void reconnect() {
  while (!client.connected()) { // Loop until we're reconnected
    char clientId[] = "hobbi@cspista.hu";
    if (client.connect(clientId, uMQTT_USER, uMQTT_PASS )) { // Attempt to connect
      client.publish("cspista/home", "DHT22 connected"); // Once connected, publish...
      client.subscribe(led_topic.c_str()); // ... and resubscribe
      client.subscribe(rgb_topic.c_str());
    } else delay(5000); // Wait 5 seconds before retrying
  }
}
```

# ESP8266\_uMQTT\_neopixel.ino

```
void callback(char* topic, byte* payload, unsigned int length) {
  if (led_topic.equals(String(topic))) {
    // Switch on the LED if an 1 was received as first character
    if ((char)payload[0] == '1') {
      digitalWrite(BUILTIN_LED, LOW); // Turn the LED on

    } else {
      digitalWrite(BUILTIN_LED, HIGH); // Turn the LED off
    }
  }
  else if (rgb_topic.equals(String(topic))) {
    long color = strtol((char*)(payload+1), NULL, 16);
    pixels.setPixelColor(0, color);
    pixels.show();
  }
}

// Az itt nem részletezett loop() és a setup_wifi() függvények ugyanazok,
// mint korábban...
```

- Az *rgb\_topic*-ba érkező adat #RRGGBB formátumú hexadecimális szám (pl. #203A4C), amit a # karakter elhagyásával *long* típusúvá konvertálunk



192.168.1.11

cspista

home

bme280 = {"temperature": "27.0", "humidity": 36, "pressure": "1010.6"}

temp = 27.5

humidity = 25

led = 1

rgb = #9D1919

cspista/home/temp



cspista/home/humidity



pressure

cspista/home/bme280



humidity

cspista/home/bme280



Topic



cspista / home / bme280

Value



QoS: 0

04/29/2021

12:45:41 PM

```
{
  "temperature": "27.0",
  "humidity": 36,
  "pressure": "1010.6"
}
```

History 99+

Publish

Topic

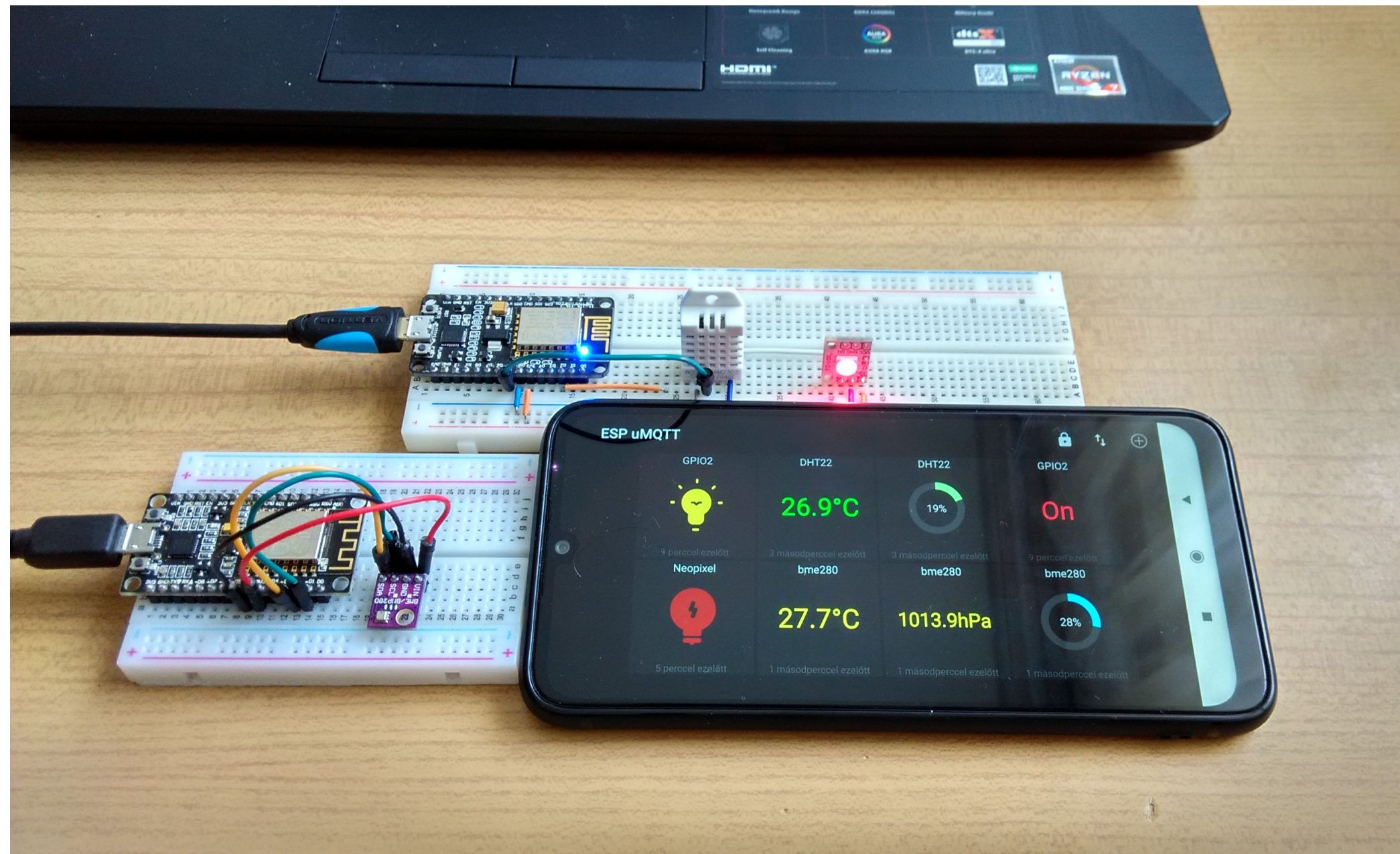
cspista/home/bme280

raw xml json

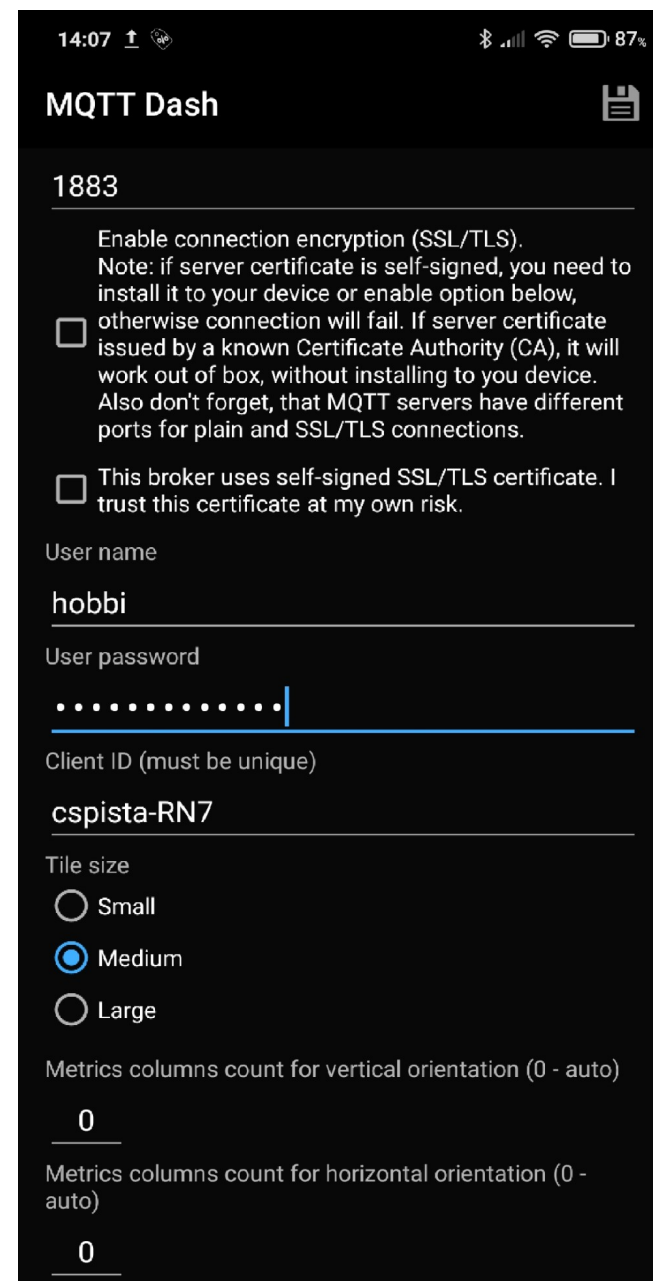
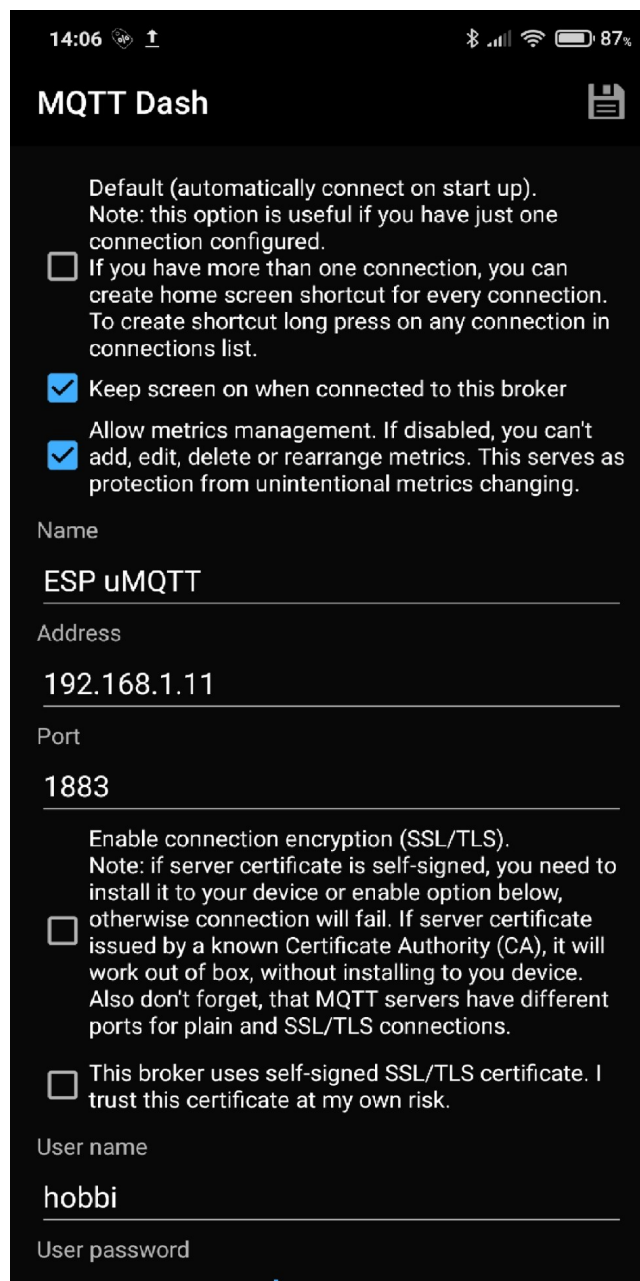
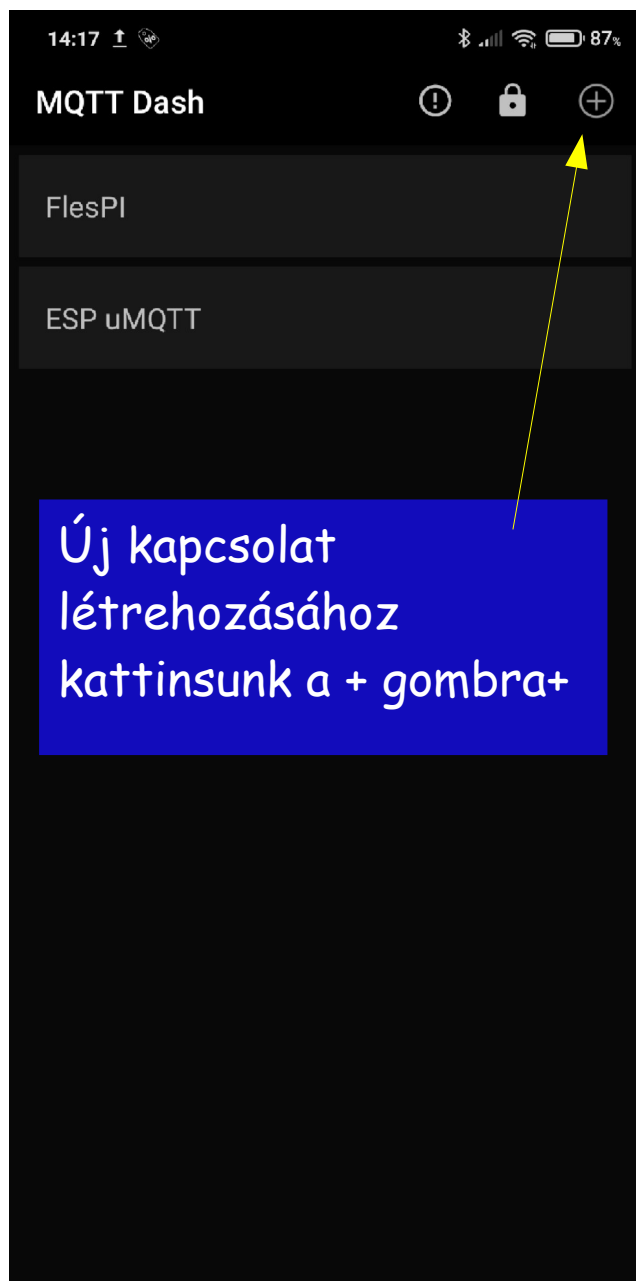


PUBLISH

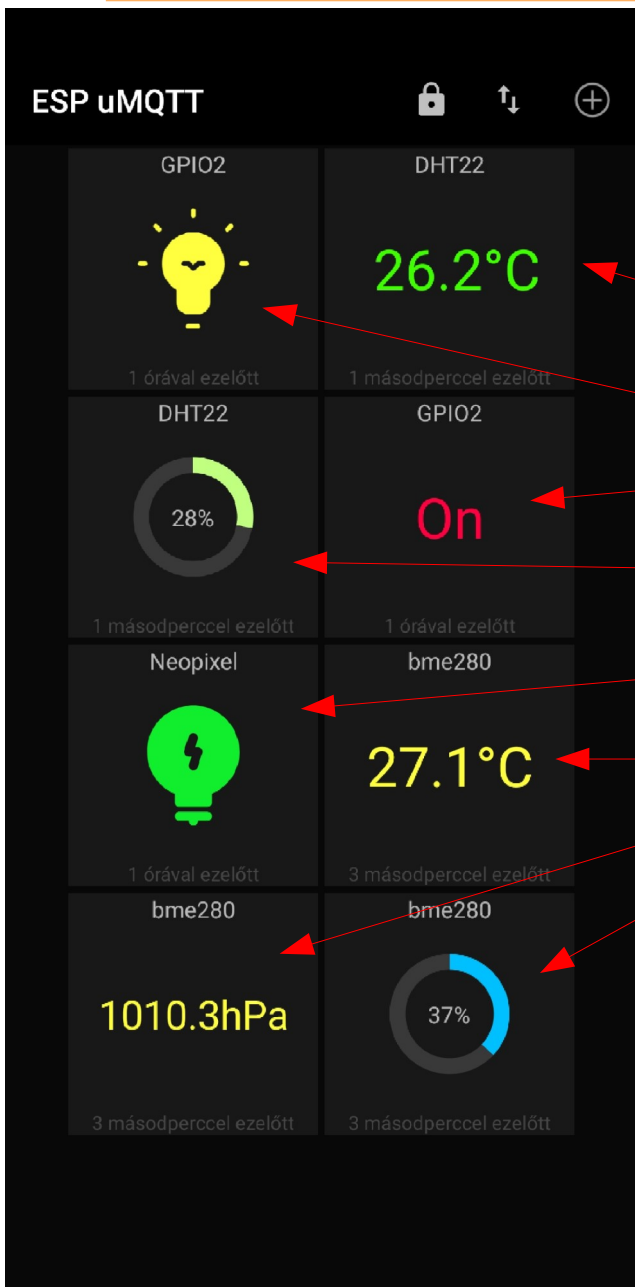
# Még egy kliens: MQTT Dash



# MQTT Dash konfigurálás



# MQTT Dash konfigurálás



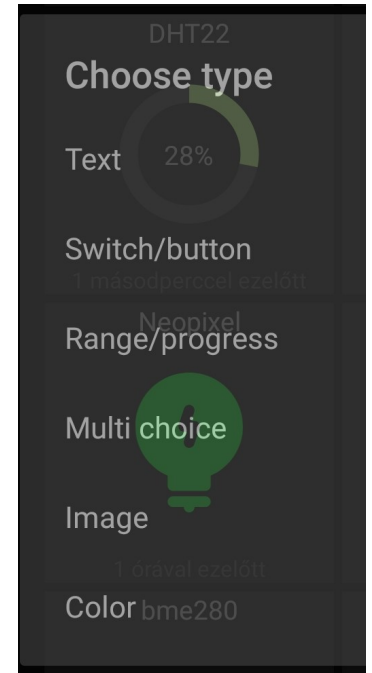
`cspista/home/temp`

`cspista/home/led`

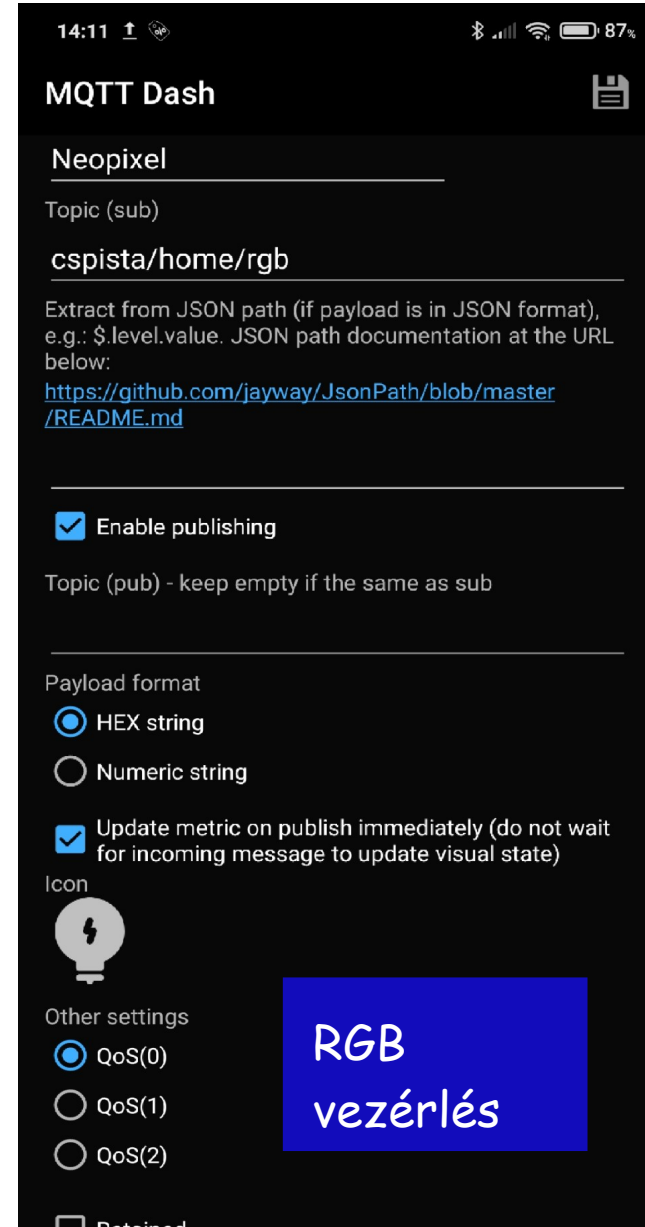
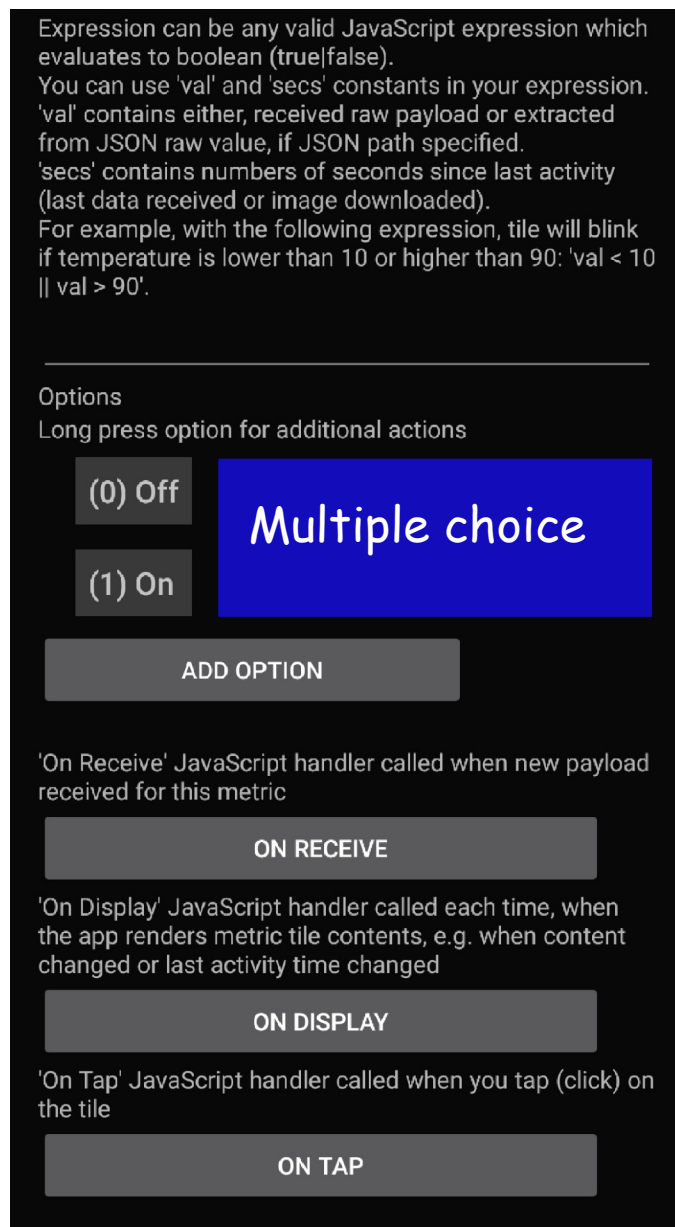
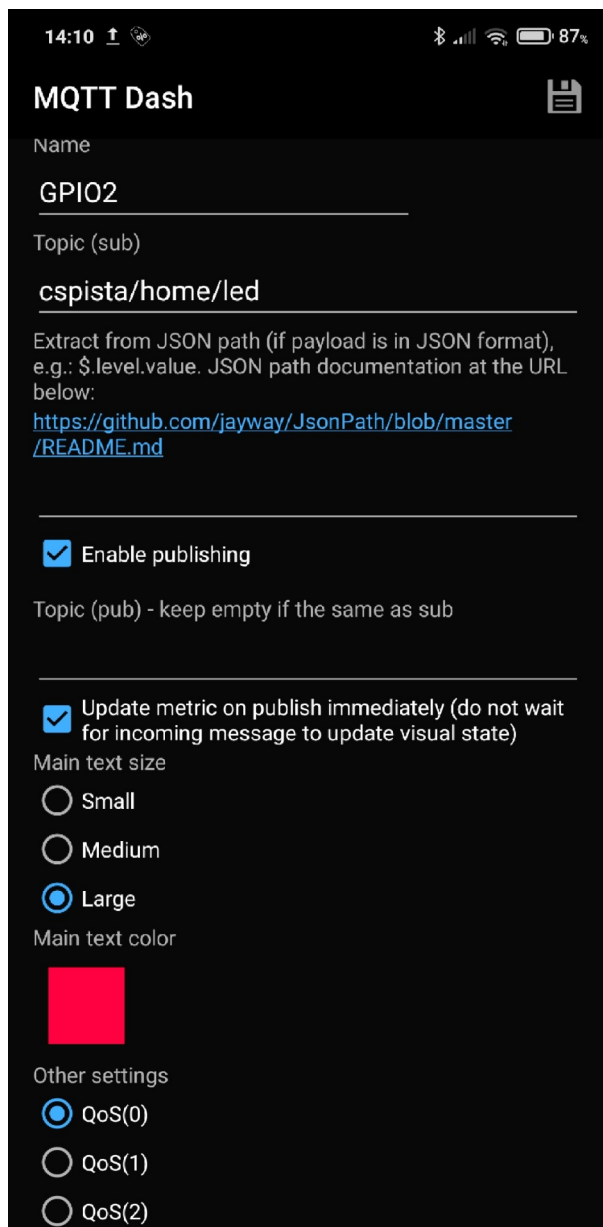
`cspista/home/humidity`

`cspista/home/rgb`

`cspista/home/bme280`

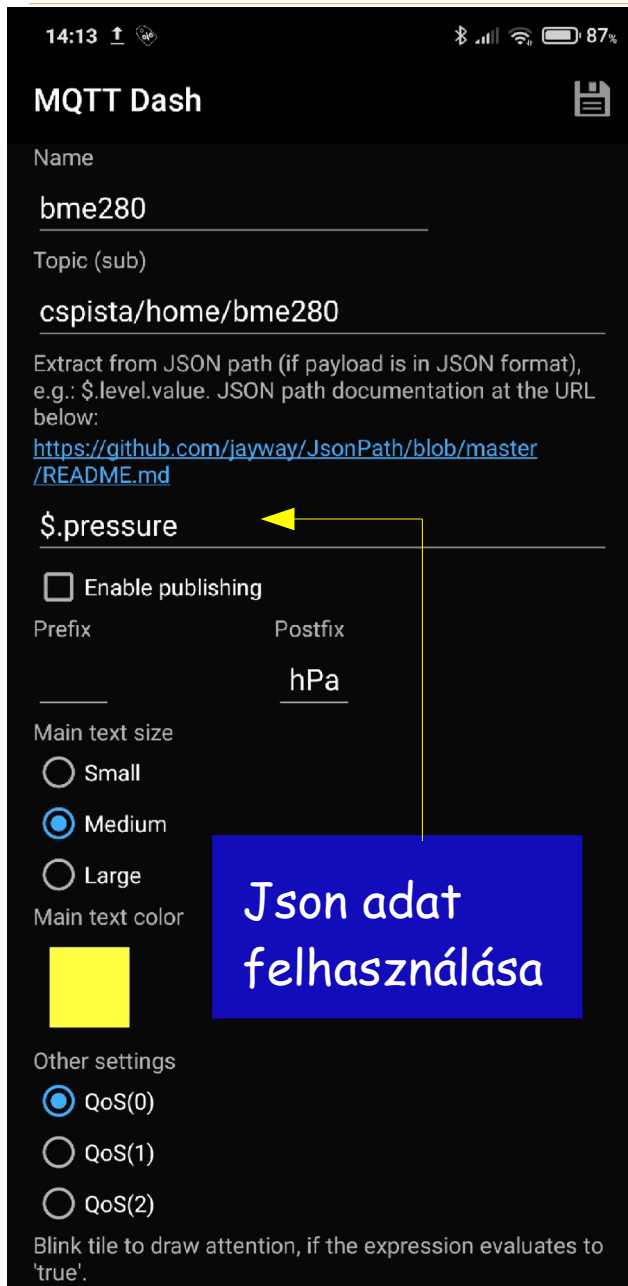


# MQTT Dash konfigurálás



# MQTT Dash konfigurálás

- Többszintű adatstruktúra esetén a neveket ponttal elválasztva adhatjuk meg
- A \$. a kiindulási szintet jelenti (gyökér)

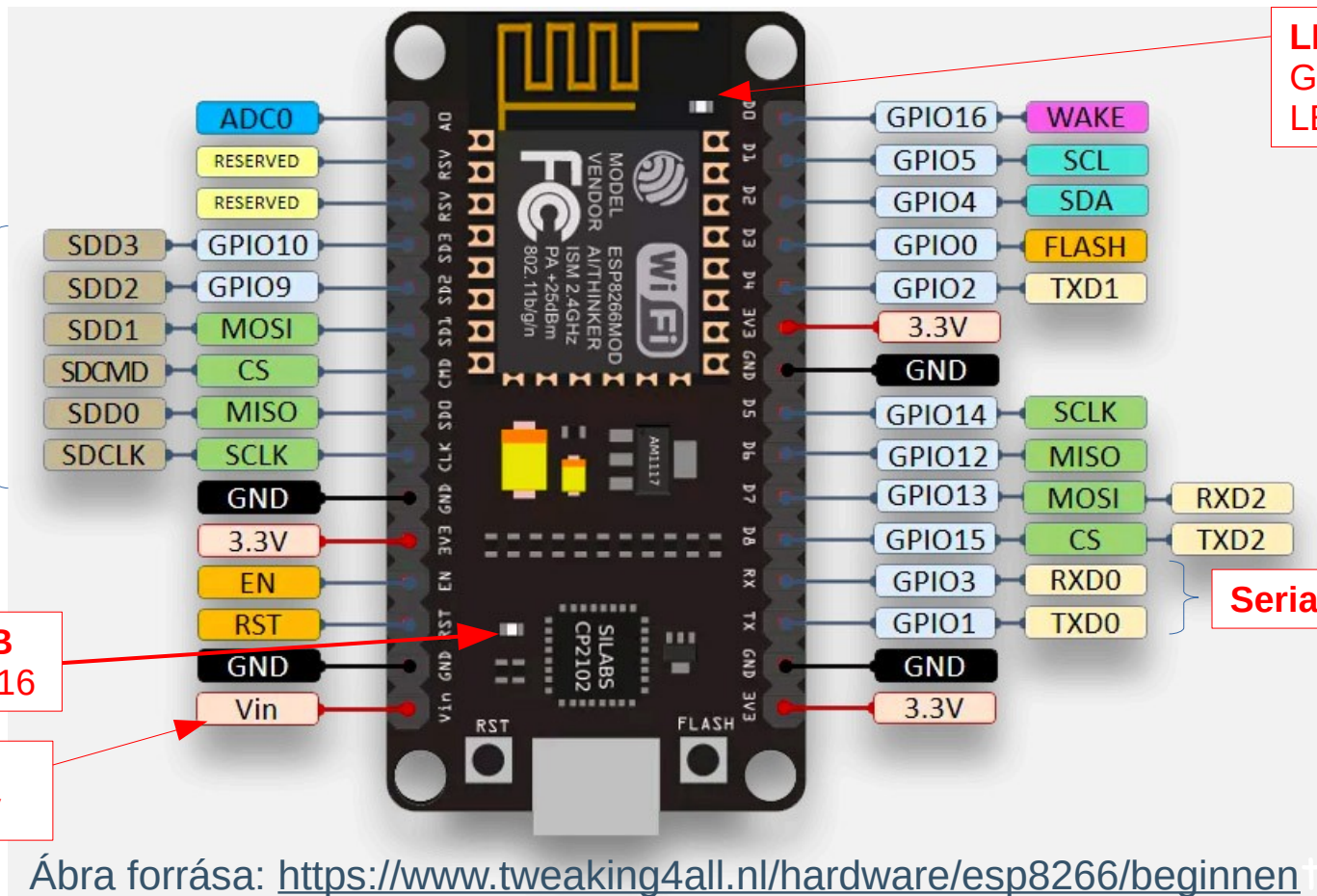




# NodeMCU GPIO kivezetések

- A ki- és bemenetek jelszintje 3,3 V (ADC0 esetén 1 V lenne, de a NodeMCU kártyán van egy 200 k + 100 k előosztó)
- FLASH a jobboldali nyomógombhoz csatlakozik
- Az Arduino számozás a **GPIO $n$**  jelölésből leválasztott  **$n$**  szám

A Flash memóriát kezelő kivezetések foglaltak!



LED A  
GPIO2,  
LED\_BUILTIN

LED B  
GPIO16

Vin  
+5V - +10 V

Serial

Név	GPIO
D0	16
D1	5
D2	4
D3	0
D4	2
D5	14
D6	12
D7	13
D8	15
A0	19

Ábra forrása: <https://www.tweaking4all.nl/hardware/esp8266/beginnen>