

## Arduino a oxymeter MAX30100

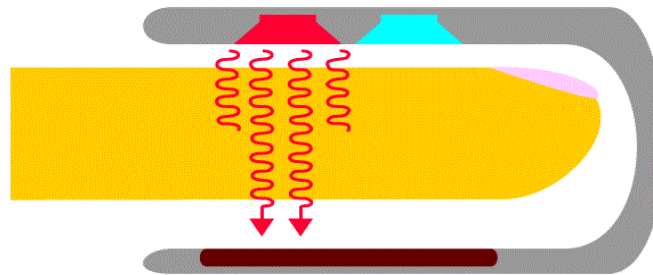
### 1. Princíp činnosti

**Senzor MAX30100** je schopný merať krvný kyslík ( $O_2$ ) a srdcovú frekvenciu (BPM = *beats per minute* = počet úderov za minútu). Koncentrácia kyslíka v krvi nazývaná SpO2 sa meria v percentách a srdcový tep/tepová frekvencia sa meria v BPM. MAX30100 je kombináciou senzora pulznej oxymetrie a monitora srdcového tepu. Má dve LED diódy, jedna vyžaruje červené svetlo a druhá vyžaruje infračervené svetlo. Na pulzovú frekvenciu je potrebné iba infračervené svetlo, na meranie hladiny kyslíka v krvi sa používa červené aj infračervené svetlo. MAX30100 číta úrovně absorpcie pre oba svetelné zdroje a ukladá ich do vyrovnávacej pamäte, ktorú je možné čítať prostredníctvom komunikačného protokolu I2C .

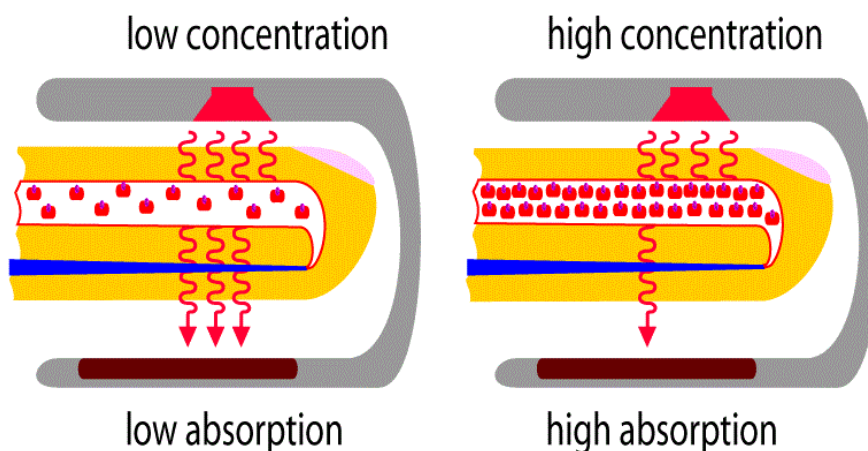


### 2. Ako funguje pulzný oxymeter?

Pri čítaní pulznej oxymetrie je na prst na ruke alebo nohe, príp. na ušný lalôčik umiestnená malá svorka.

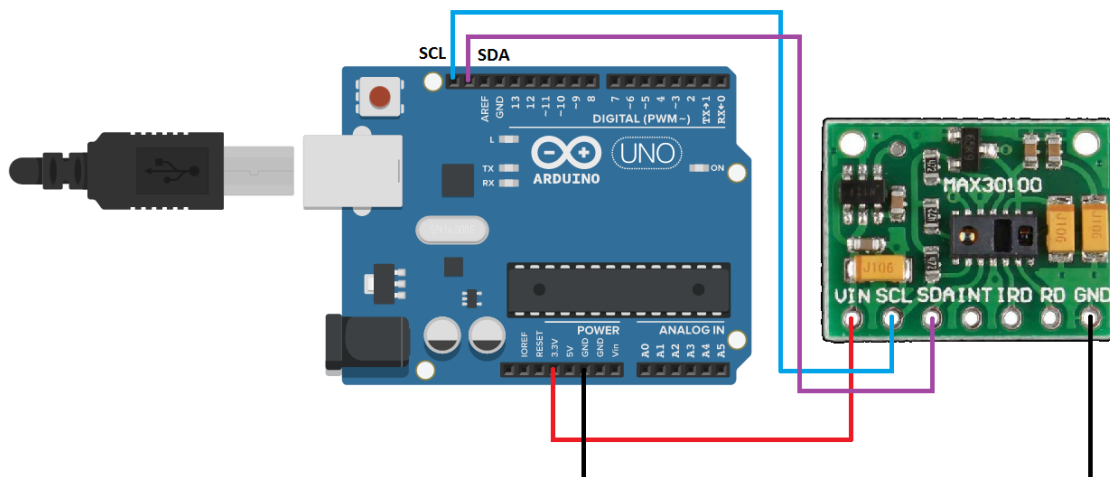


Lúče svetla prechádzajú prstom, v ktorom je krv. Okysličená krv, ktorá obsahuje viac kyslíka, absorbuje viac infračerveného svetla a prepúšťa viac červeného svetla. Odkysličená krv naopak: prepúšťa viac infračerveného svetla a absorbuje viac červené svetlo. Meraním zmien absorpcie svetla v krvi vieme zistiť koncentráciu  $O_2$ .



Svalovina srdca sa pravidelne zmršťuje (systola) a ochabuje (diastola). Pri systole vytlačí do krvného obehu okysličenú krv. Pri diastole je objem okysličenej krvi najmenší. Odmeraním času medzi nárastom a poklesom okysličenej krvi môžeme určiť pulzovú frekvenciu .

### 3. Schéma zapojenia:

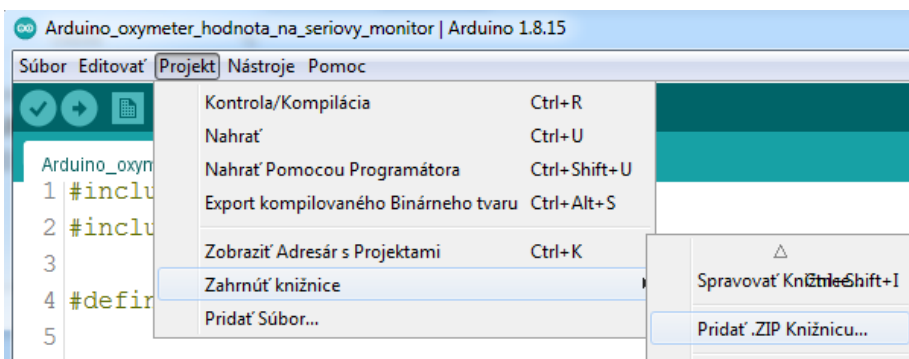


### 4. Inštalácia knižnice

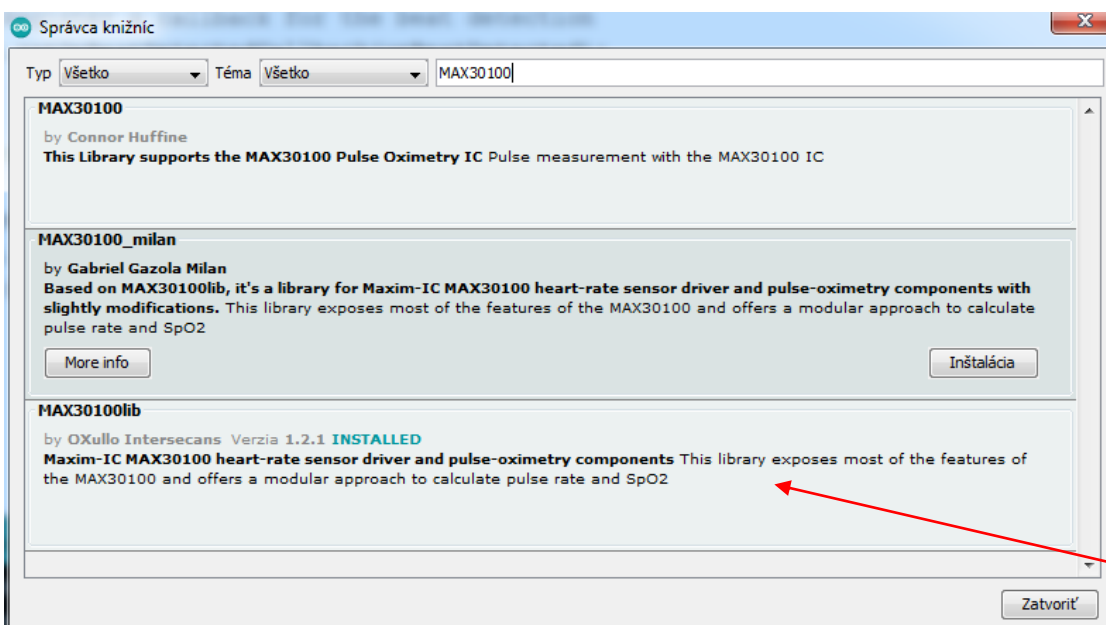
Knižnicu **MAX30100\_PulseOximeter.h** stiahneme odtiaľto

[https://drive.google.com/file/d/15w7Hp\\_Lg7FVVQoou1A56JgNDZADBuN15/view](https://drive.google.com/file/d/15w7Hp_Lg7FVVQoou1A56JgNDZADBuN15/view) ako zip súbor.

Nainštalujeme ju príkazom **Projekt-Zahrnúť knižnice-Pridať .ZIP Knižnicu...**



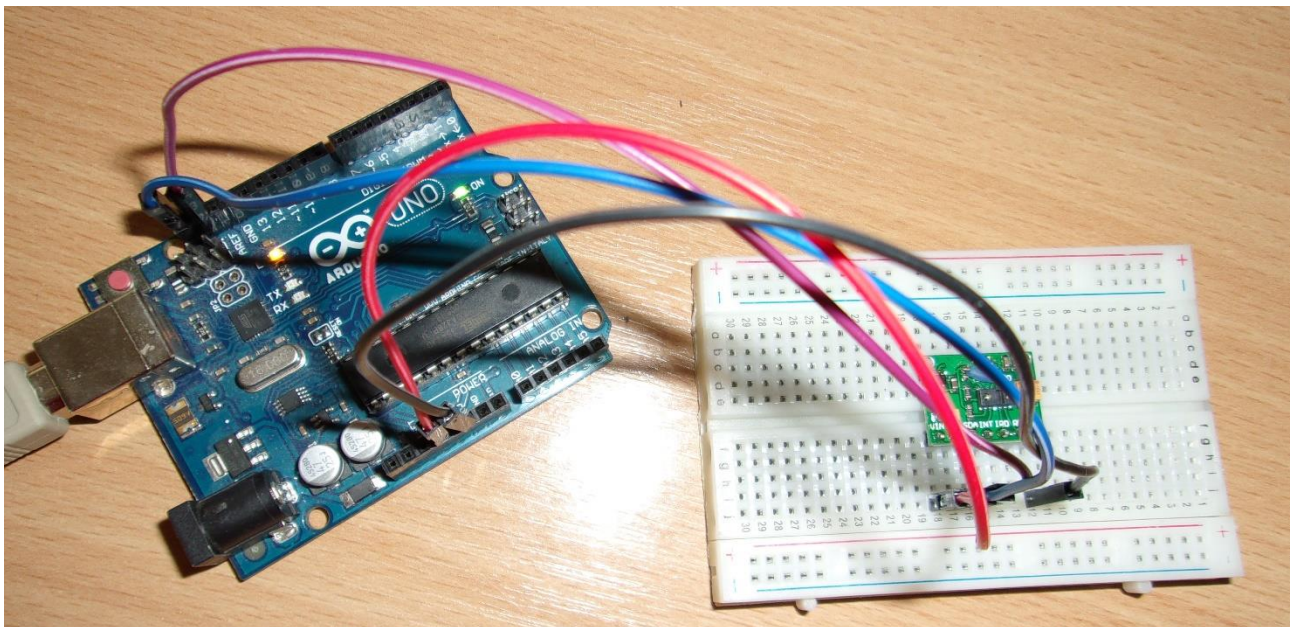
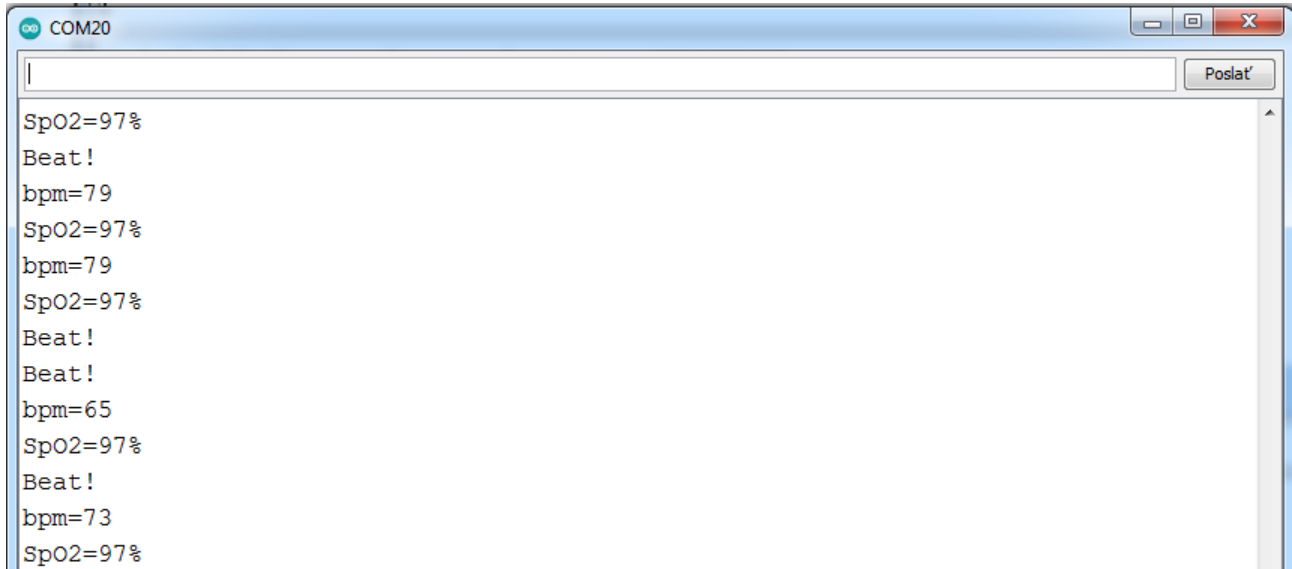
V správcovi knižníc si overíme, že sa doinštalovala:



Knižnicu môžeme samozrejme nainštalovať aj cez Správca knižníc.



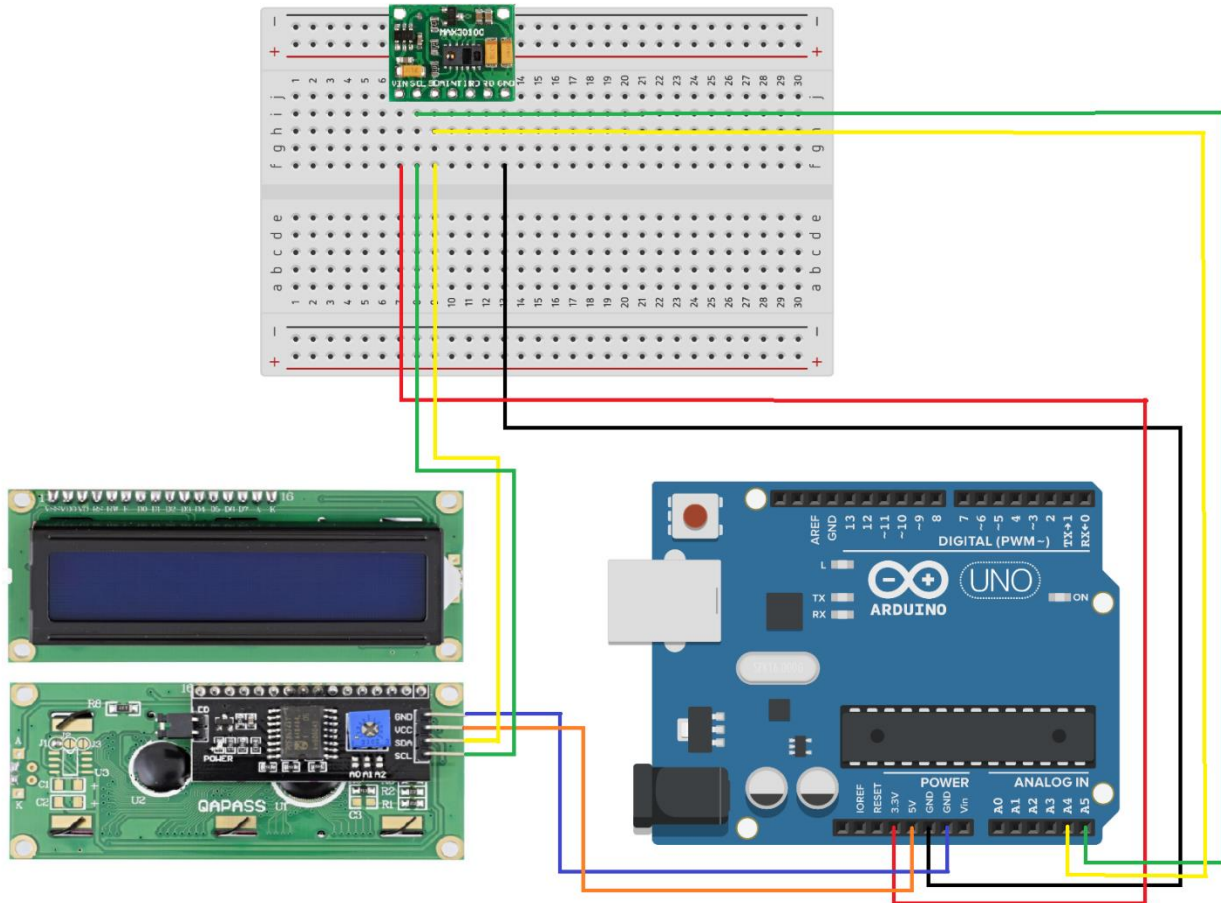
Správne hodnoty sa začnú zobrazovať po cca 15 sekundách:



## 6. Zobrazovanie hodnôt BPM a SpO2 na LCD displeji

Teraz pridáme LCD displej 16X2 s I2C na zobrazenie hodnôt. Obe zariadenia, senzor MAX30100 aj displej 16x2 používajú bus I2C, ale prevádzka na tej istej zbernici môže koexistovať s rôznymi adresami.

## 7. Schéma zapojenia:



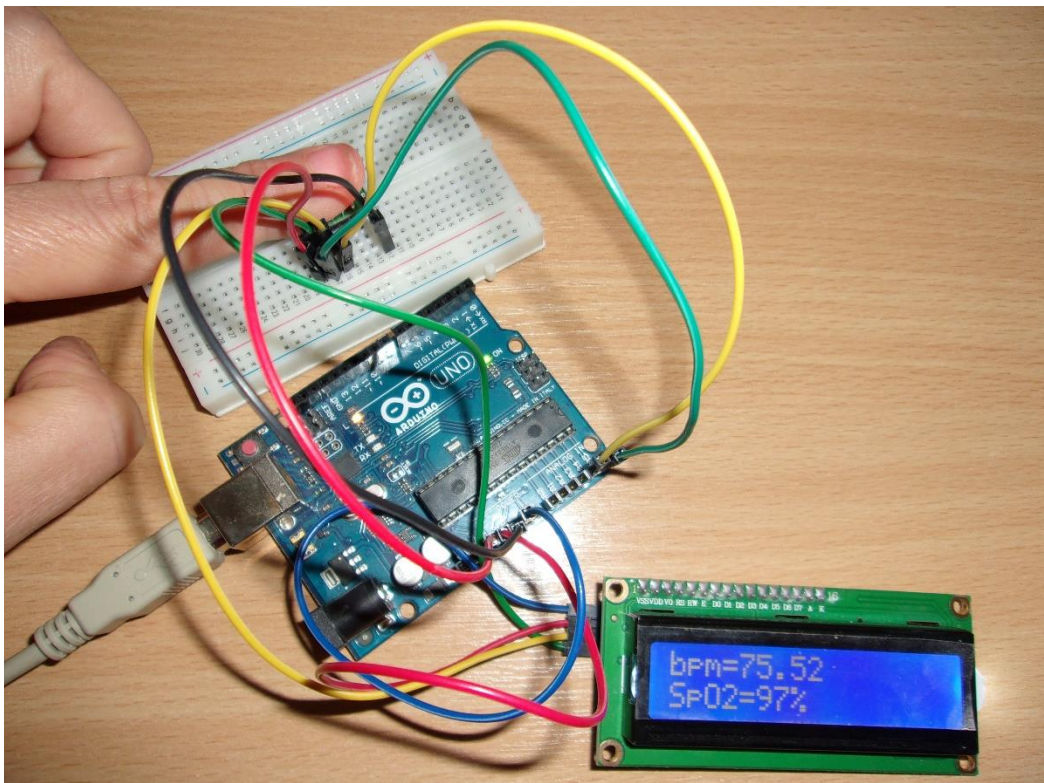
## 8. Program pre MAX30100 a displej 16x2

V tomto programe sa namerané hodnoty pulzu BPM a koncentrácie kyslíka SpO2 zobrazujú na sériovom monitore a aj na displeji.

```
Arduino_oxymeter_displej16x2 $
1 #include <Wire.h>
2 #include "MAX30100_PulseOximeter.h"
3 #define REPORTING_PERIOD_MS    1000
4 PulseOximeter pox; uint32_t tsLastReport = 0;
5 #include <LiquidCrystal_I2C.h>
6 LiquidCrystal_I2C lcd(0x27,16,2);
7
8 void onBeatDetected()
9 {
10   Serial.println("Beat!");
11 }
12
```



```
12
13 void setup()
14 {
15     Serial.begin(115200);
16     lcd.init(); lcd.backlight(); lcd.clear();
17     Serial.print("Initializing pulse oximeter..");
18     if (!pox.begin()) {
19         Serial.println("FAILED");
20         for(;;);
21     } else {
22         Serial.println("SUCCESS");
23     }
24     pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
25     pox.setOnBeatDetectedCallback(onBeatDetected);
26 }
27
28 void loop()
29 {
30     pox.update();
31     if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
32         Serial.print("bpm="); Serial.println(int(pox.getHeartRate()));
33         Serial.print("SpO2="); Serial.print(pox.getSpO2()); Serial.println("%");
34
35         lcd.clear();
36         lcd.setCursor(0,0); lcd.print("bpm="); lcd.print(pox.getHeartRate());
37         lcd.setCursor(0,1); lcd.print("SpO2="); lcd.print(pox.getSpO2()); lcd.print("%");
38         tsLastReport = millis();
39     }
40 }
```

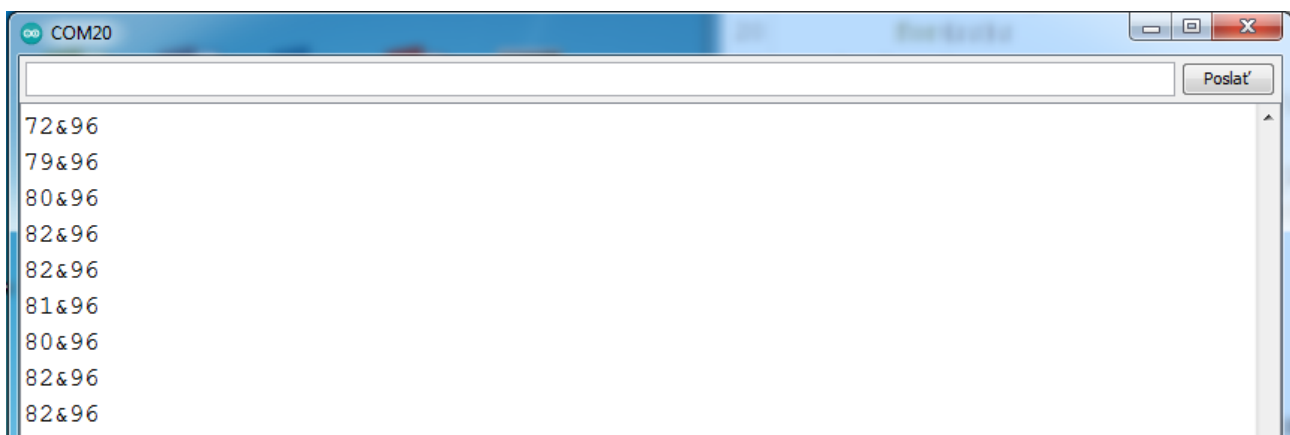


## 9. Zjednodušenie programu pre Node-RED

Ak chceme hodnoty odoslané z Arduina spracovávať v Node-RED, budeme na sériový port odosielať iba hodnoty pulzu a kyslíka, nič viac:

```
10 //Serial.println("Beat!");
11 }
12
13 void setup()
14 {
15     Serial.begin(115200);
16     lcd.init(); lcd.backlight(); lcd.clear();
17     //Serial.print("Initializing pulse oximeter..");
18     if (!pox.begin()) {
19         //Serial.println("FAILED");
20         for(;;);
21     } else {
22         //Serial.println("SUCCESS");
23     }
24     pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
25     pox.setOnBeatDetectedCallback(onBeatDetected);
26 }
27
28 void loop()
29 {
30     pox.update();
31     if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
32         //Serial.print("bpm=");
33         Serial.print(int(pox.getHeartRate()));
34         //Serial.print("SpO2=");
35         Serial.print("&");
36         Serial.println(pox.getSpO2());
37         //Serial.println("%");
38     }
```

Prvá hodnota je pulz, druhá je kyslík:



## 10. Node-RED

Arduino IDE môžeme zavrieť a spustiť si Node-RED. Spolu sa to niekedy neznáša.

V Príkazovom riadku zadáme príkaz `node-red`

```

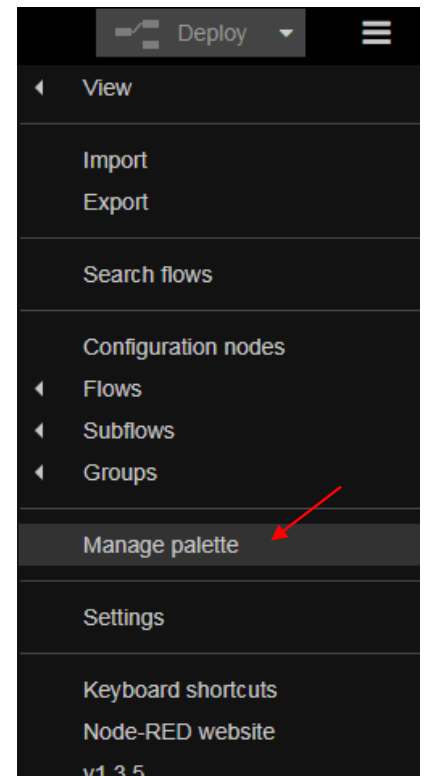
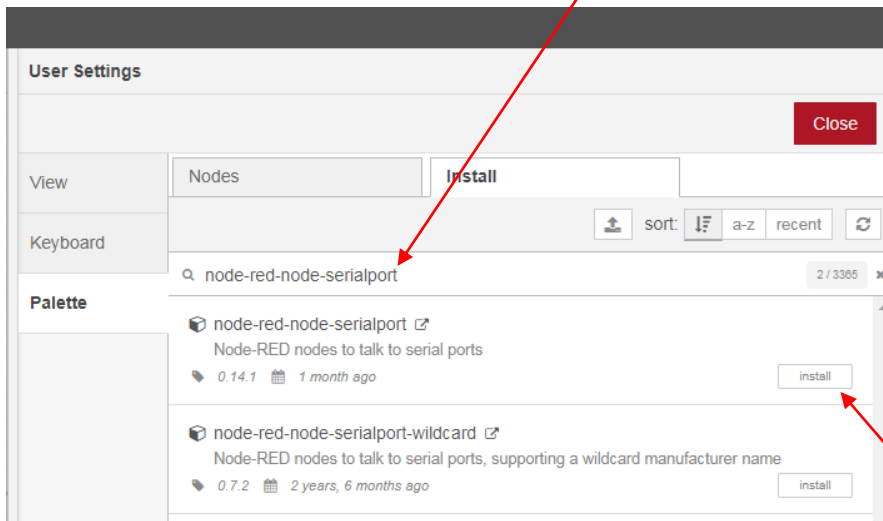
C:\Users\hubrik>node-red
15 Aug 19:29:17 - [info]

Welcome to Node-RED
=====
  
```

Otvoríme si nejaký internetový prehliadač a zadáme adresu `localhost:1880`

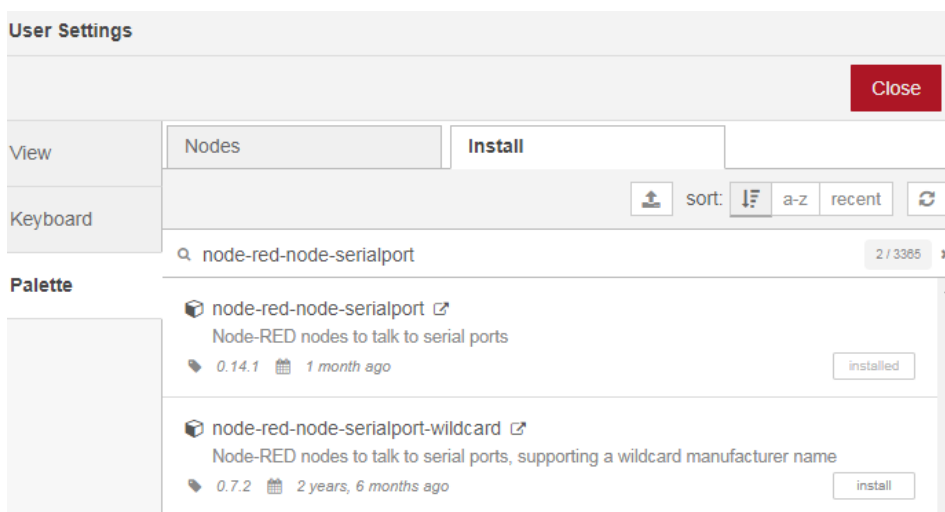
Node-RED nemá predinštalované uzly pre Arduino a Serial Port. Doinštalujeme ich pomocou Manažéra paliet:

Sem napíšeme iba slovo *serialport*, nájde nám 2 palety.



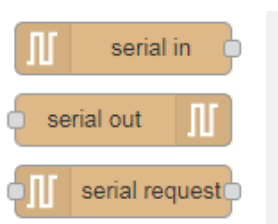
Túto nainštalujeme.

Po nainštalovaní to vyzerá takto:





V paneli uzlov, v skupine network nám pribudnú tieto uzly:



Podobne doinštalujeme uzly pre Arduino:

The screenshot shows the 'User Settings' dialog box in Node-RED, with the 'Install' tab selected. A search bar contains the text 'Arduino', with a red arrow pointing to it. Below the search bar is a list of nodes. The last node in the list is 'node-red-node-arduino', which has a red arrow pointing to its 'install' button. A callout box on the right contains the text 'Túto nainštalujeme.' (We will install this one).

Node Name	Description	Version	Release Date	Action
node-red-contrib-idm	Nodes for IDM and DUO	0.4.10	4 years, 6 months ago	install
node-red-contrib-johnny-five	A set of node-red nodes for using johnny-five and IO plugins	1.0.0-beta.2	1 year ago	install
node-red-contrib-johnny5	A set of node-red nodes for using Johnny-Five and IO plugins (fork)	0.50.0	1 year, 8 months ago	install
node-red-contrib-simplecomm-node	A simple communication node for node-RED	1.0.2	2 years, 8 months ago	install
node-red-contrib-thinger	Node-Red library for Thinger.io Platform	0.0.4	3 years, 2 months ago	install
node-red-contrib-webduino	Node-RED nodes for Webduino	0.0.15	3 years, 4 months ago	install
node-red-node-arduino	A Node-RED node to talk to an Arduino running firmware	0.3.1	1 year, 9 months ago	install

## 11. Pokusný flow

Zostavíme pokusný tok z uzlov **serial in** a **debug**:



Konfigurácia uzla **serial in**:

**Edit serial in node**

Delete Cancel Done

⚙ Properties

Serial Port COM20:9600-8N1

Name Vstup z Arduino

Klikneme na ikonu ceruzky.

**Edit serial in node > Edit serial-port node**

Delete Cancel Update

⚙ Properties

Serial Port COM20

Settings

Baud Rate	Data Bits	Parity	Stop Bits
115200	8	None	1
DTR	RTS	CTS	DSR
auto	auto	auto	auto

Input

Optionally wait for a start character of , then

Split input on the character  and deliver ASCII strings

Output

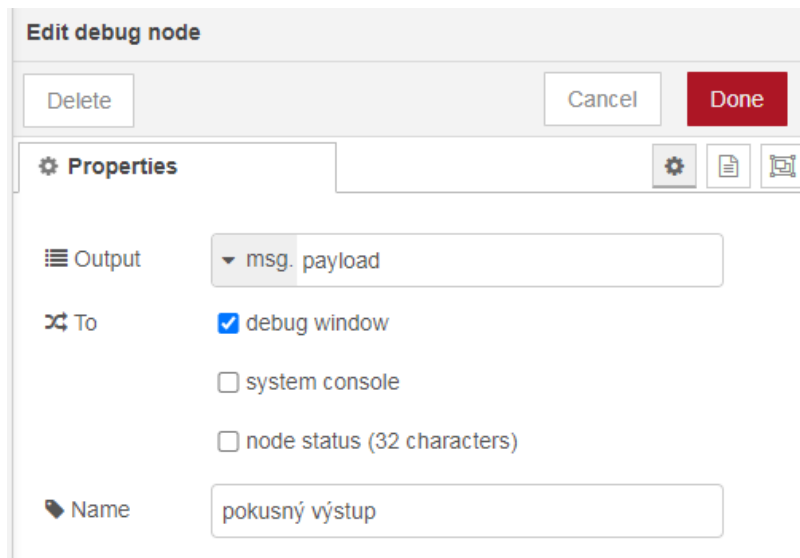
Add character to output messages false

Request

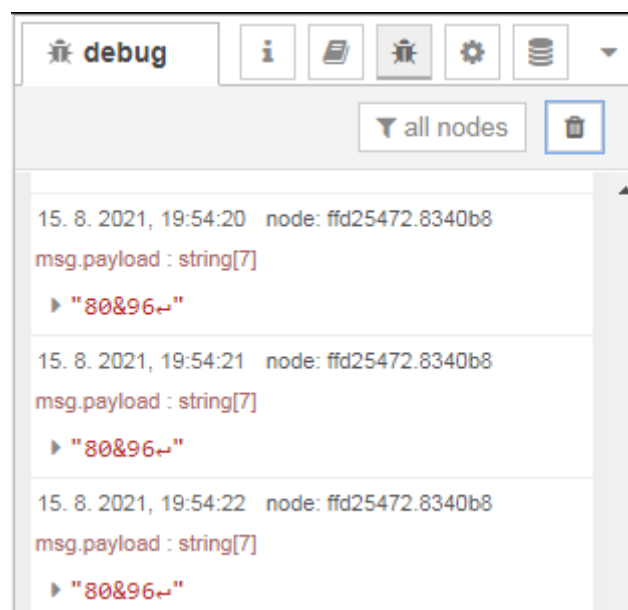
Default response timeout 10000 ms

Zadáme taký port, na ktorom máme pripojené Arduino..

Konfigurácia uzla **debug**:

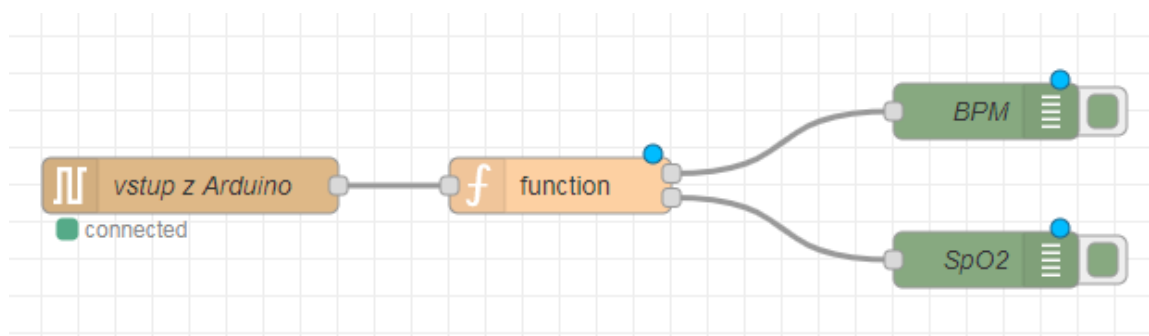


Po kliknutí na tlačidlo **Deploy** sa v debug okne každú 1 sekundu zobrazujú hodnoty pulzu a koncentrácie kyslíka zo senzoru MAX30100, pripojenom k Arduinu (Arduino sme naprogramovali tak, že vysiela hodnoty každú 1 sekundu):



## 12. Vylepšený flow

Náš flow teraz vylepšíme takto – pridáme uzol **function** s 2 výstupmi a ešte jeden uzol **debug**:



Konfigurácia uzla **function**:

**Edit function node**

Delete Cancel Done

**Properties**

Name

**Setup** On Start On Message On Stop

Outputs 2

**Edit function node**

Delete Cancel Done

**Properties**

Name

Setup On Start **On Message** On Stop

```
1 var t1=msg.payload[0];
2 var t2=msg.payload[1];
3 var res1="BPM="+t1+t2;
4 var result1={payload:res1};
5
6 var t3=msg.payload[3];
7 var t4=msg.payload[4];
8 var res2="SpO2="+t3+t4;
9 var result2={payload:res2};
10
11 return [result1,result2];
```

Konfigurácia horného uzla **debug**:

**Edit debug node**

Delete Cancel Done

**Properties**

Output

To  debug window  
 system console  
 node status (32 characters)

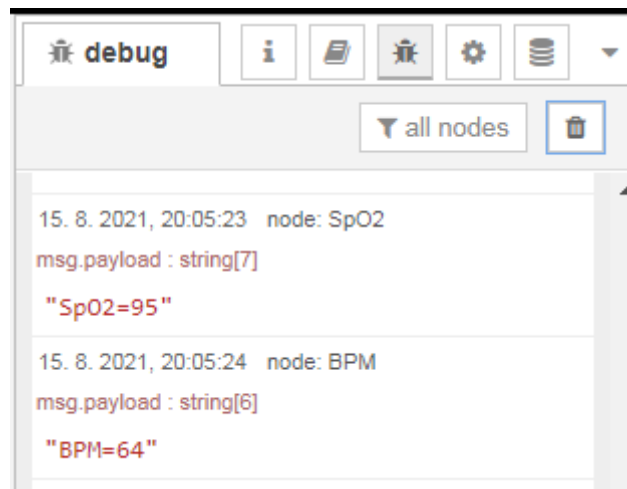
Name

Konfigurácia dolného uzla **debug**:

The screenshot shows the 'Edit debug node' dialog box. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. Below this is a 'Properties' section with a gear icon, a document icon, and a refresh icon. The configuration includes:

- Output:** A dropdown menu set to 'msg. payload'.
- To:** A checked checkbox for 'debug window', and unchecked checkboxes for 'system console' and 'node status (32 characters)'.
- Name:** A text input field containing 'SpO2'.

Výstup v debug okne po kliknutí na tlačidlo **Deploy**:



### 13. Grafické znázornenie hodnôt na Dashboarde

Zvýšime počet výstupov funkcie na 4:

The screenshot shows the 'Edit function node' dialog box. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. Below this is a 'Properties' section with a gear icon, a document icon, and a refresh icon. The configuration includes:

- Name:** A text input field containing 'Name'.
- Setup:** A row of three buttons: 'On Start', 'On Message', and 'On Stop'.
- Outputs:** A numeric input field set to '4' with up and down arrow buttons.



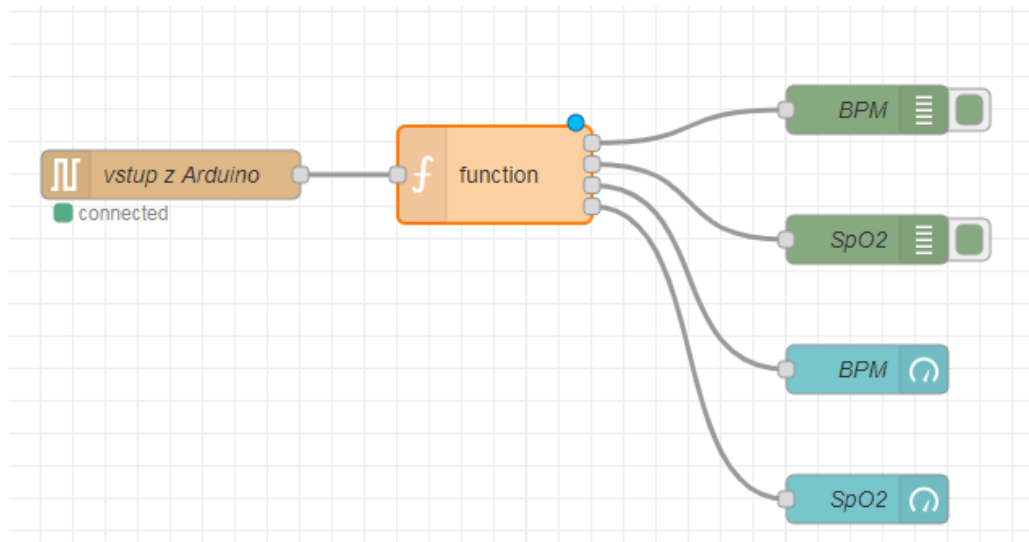
Program funkcie doplníme takto:

```

Edit function node
Delete Cancel Done
Properties
Name Name
Setup On Start On Message On Stop
1 var t1=msg.payload[0];
2 var t2=msg.payload[1];
3 var res1="BPM="+t1+t2;
4 var result1={payload:res1};
5
6 var t3=msg.payload[3];
7 var t4=msg.payload[4];
8 var res2="SpO2="+t3+t4;
9 var result2={payload:res2};
10
11 var res3=t1+t2;
12 var result3={payload:res3};
13
14 var res4=t3+t4;
15 var result4={payload:res4};
16
17 return [result1,result2,result3,result4];

```

Pridáme uzly **gauge**:



Konfigurácia horného uzla **gauge**:

**Edit gauge node**

Delete Cancel Done

**Properties**

Group [Pracovňa Mirka] Meranie pulzu a konc...

Size auto

Type Gauge

Label BPM

Value format {{value}}

Units units

Range min 0 max 200

Colour gradient

Sectors 0 ... optional ... optional ... 200

Name BPM

**Edit gauge node > Edit dashboard group node**

Delete Cancel Update

**Properties**

Name Meranie pulzu a koncentrácie kyslíka

Tab Pracovňa Mirka

Width 20

Display group name

Allow group to be collapsed

**Edit gauge node > Edit dashboard group node > Edit dashboard tab node**

Delete Cancel Update

**Properties**

Name Pracovňa Mirka

Icon dashboard

State  Enabled

Nav. Menu  Visible

Klikneme na ikonu ceruzky. Dostaneme sa do ďalšieho dialógového okna, kde vyberáme tabuľku (jej názov je v tej hranatej zátvorke) a zadávame meno skupiny (to je uvedené za zátvorkou).

Konfigurácia dolného uzla **gauge**:

The screenshot shows the 'Edit gauge node' configuration window. At the top, there are 'Delete', 'Cancel', and 'Done' buttons. Below is the 'Properties' section with various settings:

- Group:** [Pracovňa Mirka] Meranie pulzu a konce
- Size:** auto
- Type:** Gauge
- Label:** SpO2
- Value format:** {{value}}%
- Units:** units
- Range:** min 0, max 100
- Colour gradient:** A horizontal bar with green, yellow, and red segments.
- Sectors:** 0, optional, optional, 100
- Name:** SpO2


Nastavenie dashboardu:

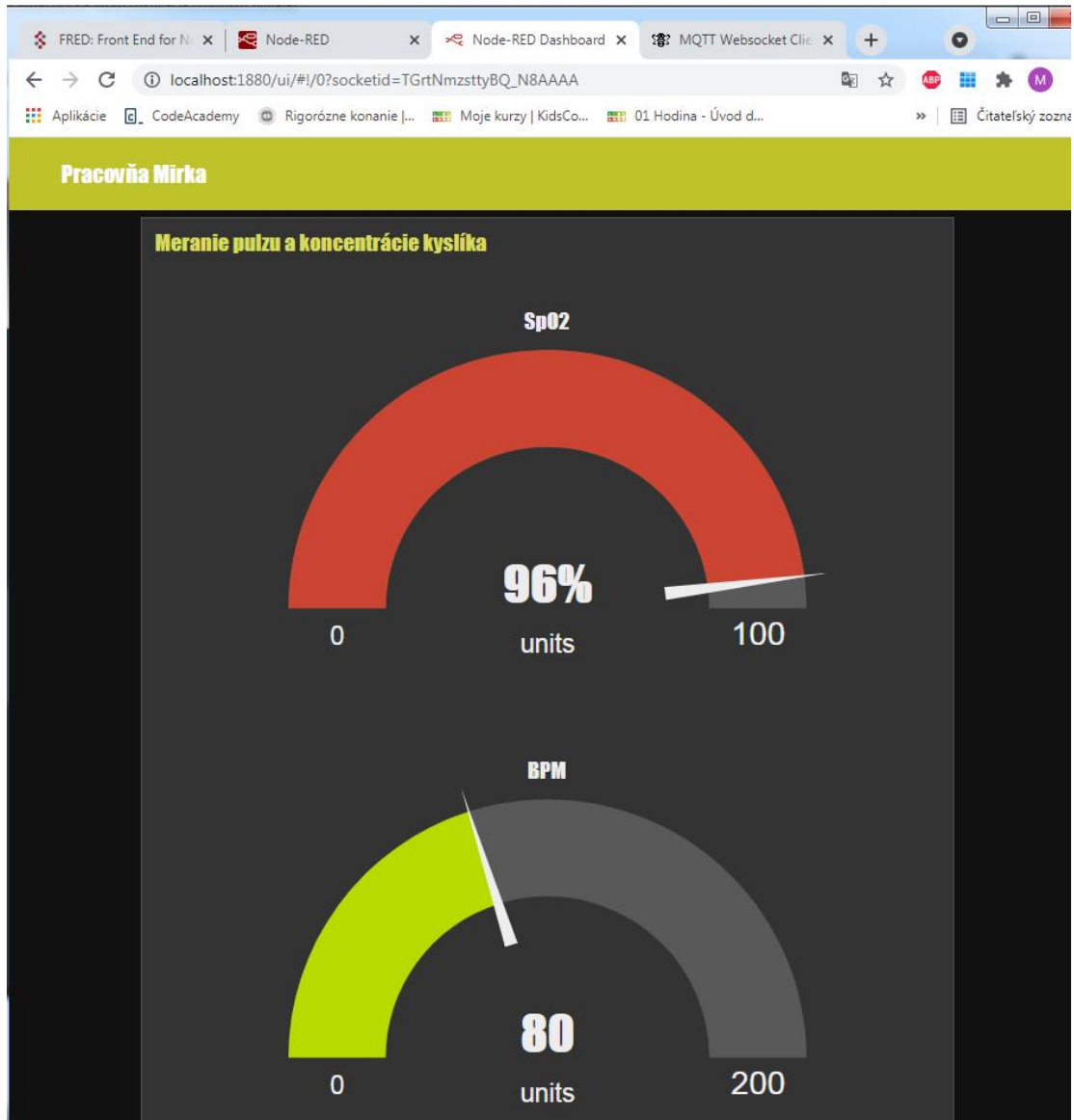
The first screenshot shows the 'dashboard' configuration panel with tabs for 'Layout', 'Site', and 'Theme'. Under the 'Theme' tab, there is a 'Tabs & Links' section showing a tree structure:

- Pracovňa Mirka
  - Meranie pulzu a koncentráci
    - SpO2
    - BPM

The second screenshot shows the 'Theme' configuration panel with the following settings:

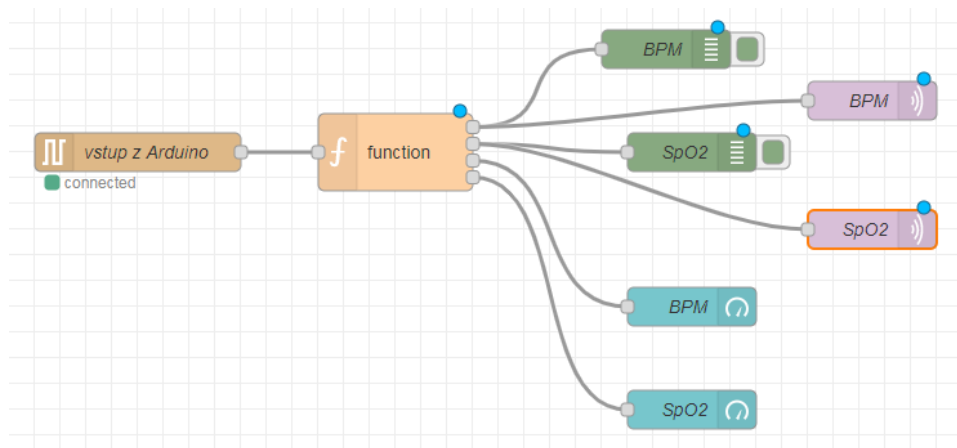
- Style:** Dark
- Base Settings:**
  - Colour:** A yellow-green color swatch.
  - Font:** Impact

Po kliknutí na ikonku  sa na novej karte prehliadača zobrazí dashboard:

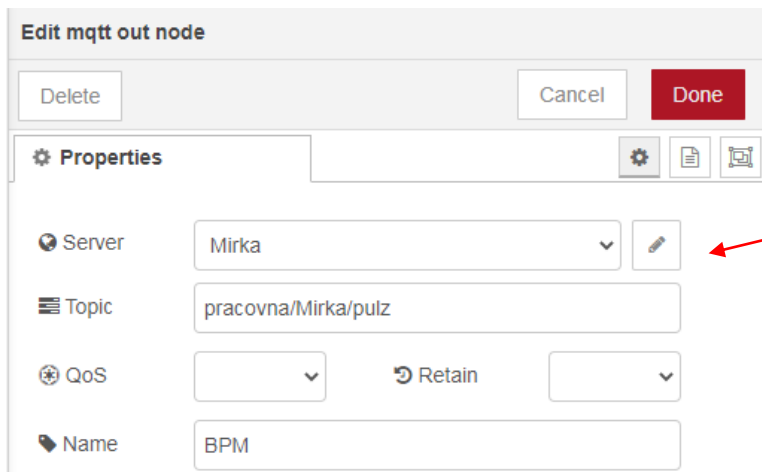


#### 14. Publikovanie správ (hodnoty pulzu) na HiveMQ

Do flowu pridáme 2 uzly **mqtt out**:



Vlastnosti uzla horného uzla **mqtt out**:

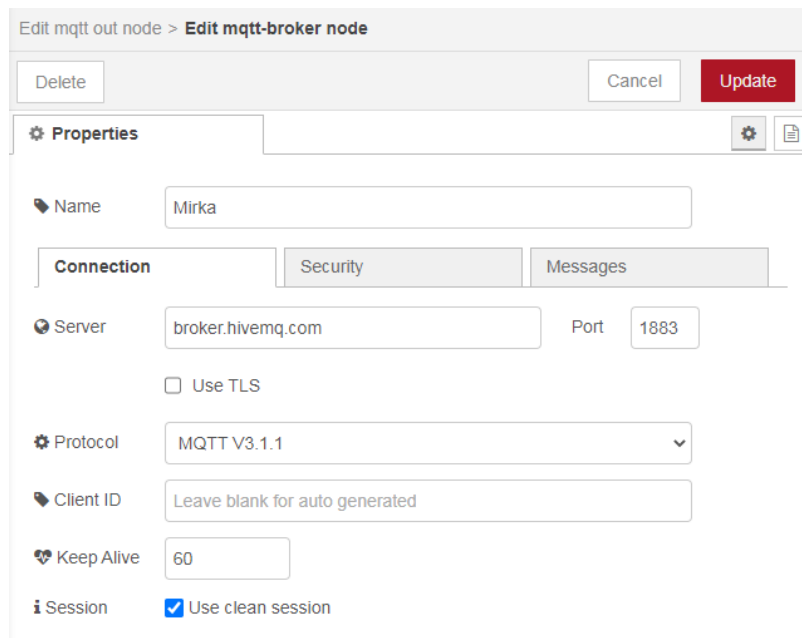


Dialog box "Edit mqtt out node" with the following fields:

- Server: Mirka
- Topic: pracovna/Mirka/pulz
- QoS: 0
- Retain:
- Name: BPM

Meno servera zadáme v ďalšom dialógovom okne, ktoré sa zobrazí po kliknutí na ikonu ceruzky.

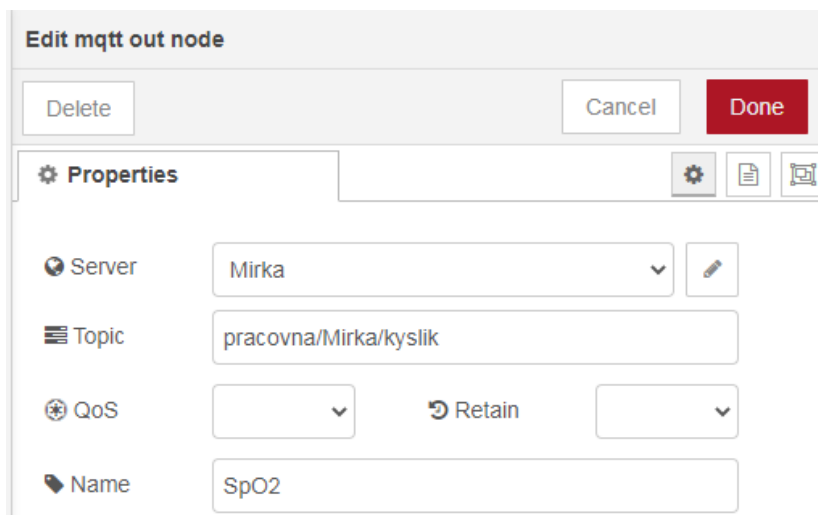
Nastavenie mqtt brokera:



Dialog box "Edit mqtt out node > Edit mqtt-broker node" with the following fields:

- Name: Mirka
- Connection tab selected
- Server: broker.hivemq.com
- Port: 1883
- Use TLS:
- Protocol: MQTT V3.1.1
- Client ID: Leave blank for auto generated
- Keep Alive: 60
- Session:  Use clean session

Vlastnosti uzla dolného uzla **mqtt out**:



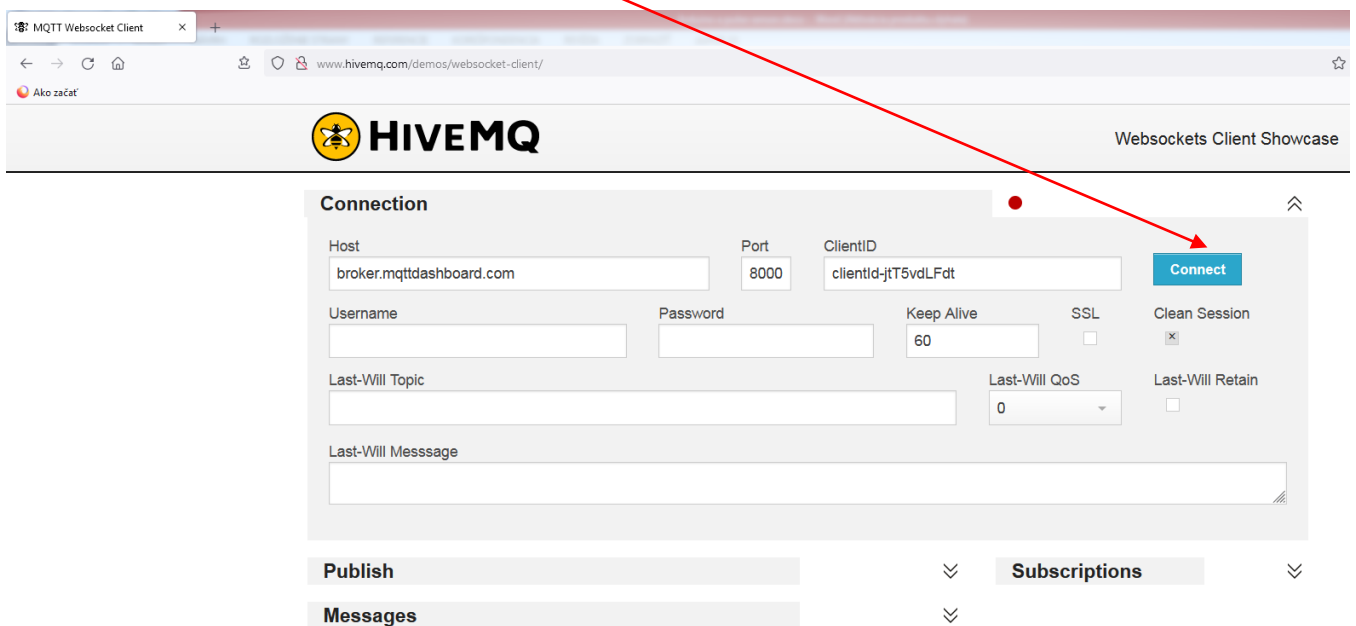
Dialog box "Edit mqtt out node" with the following fields:

- Server: Mirka
- Topic: pracovna/Mirka/kyslik
- QoS: 0
- Retain:
- Name: SpO2



## 15. Pripojenie na brokera HiveMQ

Pripojíme sa na brokera HiveMQ <http://www.hivemq.com/demos/websocket-client/> a klikneme na tlačidlo **Connect**:

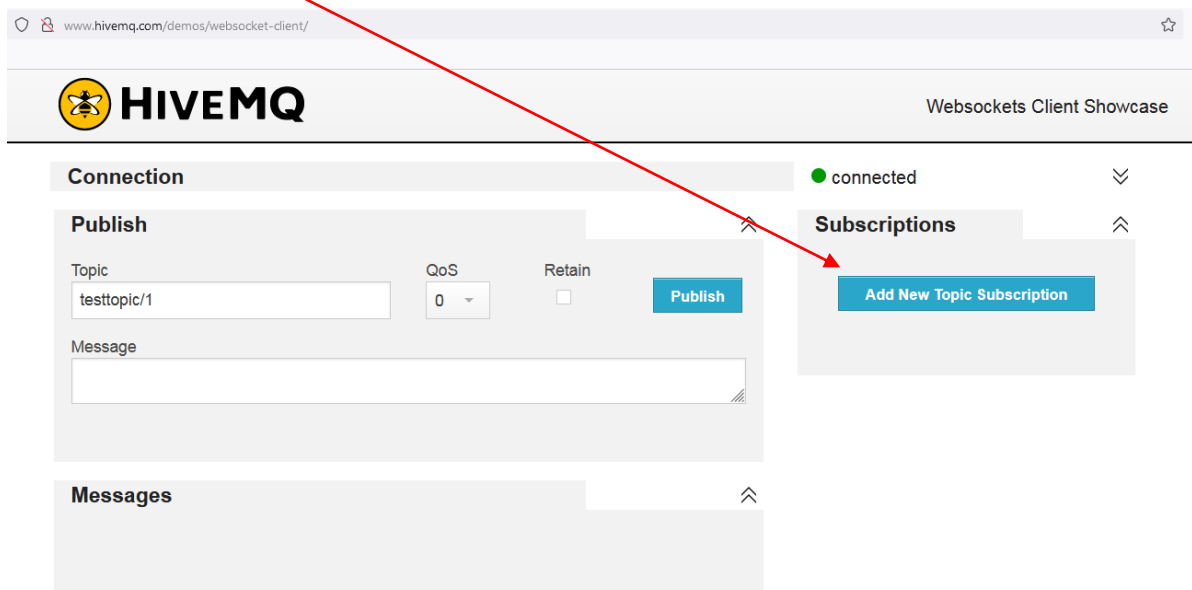


The screenshot shows the HiveMQ Websockets Client Showcase interface. The 'Connection' section is active, displaying the following fields:

- Host: broker.mqttdashboard.com
- Port: 8000
- ClientID: clientId-jtT5vdLFdt
- Username: (empty)
- Password: (empty)
- Keep Alive: 60
- SSL: (unchecked)
- Clean Session: (checked)
- Last-Will Topic: (empty)
- Last-Will QoS: 0
- Last-Will Retain: (unchecked)
- Last-Will Message: (empty)

A red arrow points to the 'Connect' button. Below the connection form are sections for 'Publish', 'Subscriptions', and 'Messages', all currently collapsed.

Po pripojení zadáme odber témy (topic):

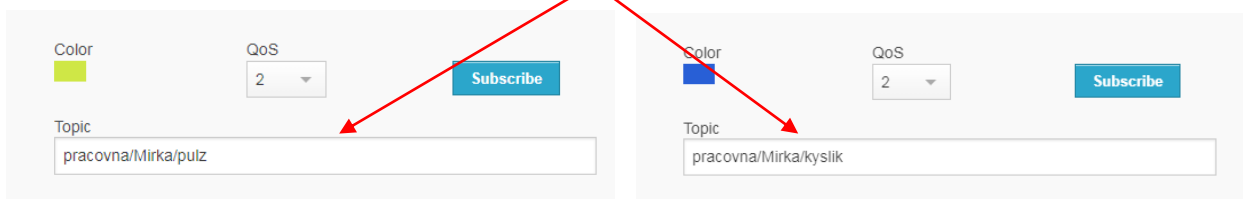


The screenshot shows the HiveMQ Websockets Client Showcase interface after a successful connection. The 'Connection' section is now labeled 'connected'. The 'Subscriptions' section is expanded, showing an 'Add New Topic Subscription' button. The 'Publish' section is also visible, with the following fields:

- Topic: testtopic/1
- QoS: 0
- Retain: (unchecked)
- Message: (empty)

A red arrow points to the 'Add New Topic Subscription' button. The 'Messages' section is also visible but empty.

Zadáme tie isté témy (topic), ktoré sme zadali aj v Node-Red:



The image shows two screenshots of Node-Red MQTT client nodes. The left node has the following configuration:

- Color: Green
- QoS: 2
- Topic: pracovna/Mirka/pulz

The right node has the following configuration:

- Color: Blue
- QoS: 2
- Topic: pracovna/Mirka/kyslik

Red arrows point from the 'Add New Topic Subscription' button in the previous screenshot to the 'Topic' input fields of these two Node-Red nodes.

V okne **Messages** sa budú zobrazovať hodnoty, ktoré Node-Red vyslal po prijatí z Arduina:

**HIVEMQ** Websockets Client Showcas

**Connection** ● connected

**Publish**

Topic: testtopic/1 QoS: 0 Retain:  Publish

Message:

**Messages**

2021-08-15 20:40:55	Topic: pracovna/Mirka/kyslik	Qos: 0
SpO2=97		
2021-08-15 20:40:55	Topic: pracovna/Mirka/pulz	Qos: 0
BPM=80		

**Subscriptions**

Add New Topic Subscription

- Qos: 2 pracovna/Mirka/pulz
- Qos: 2 pracovna/Mirka/kyslik