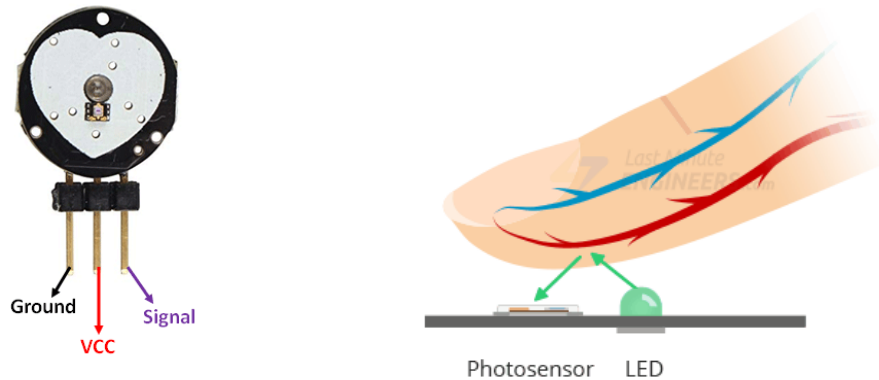


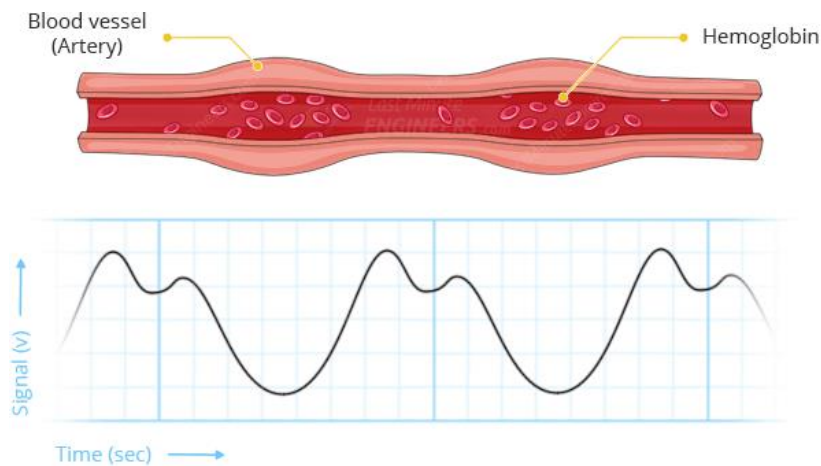
Arduino a senzor pulzu

1. Princíp činnosti

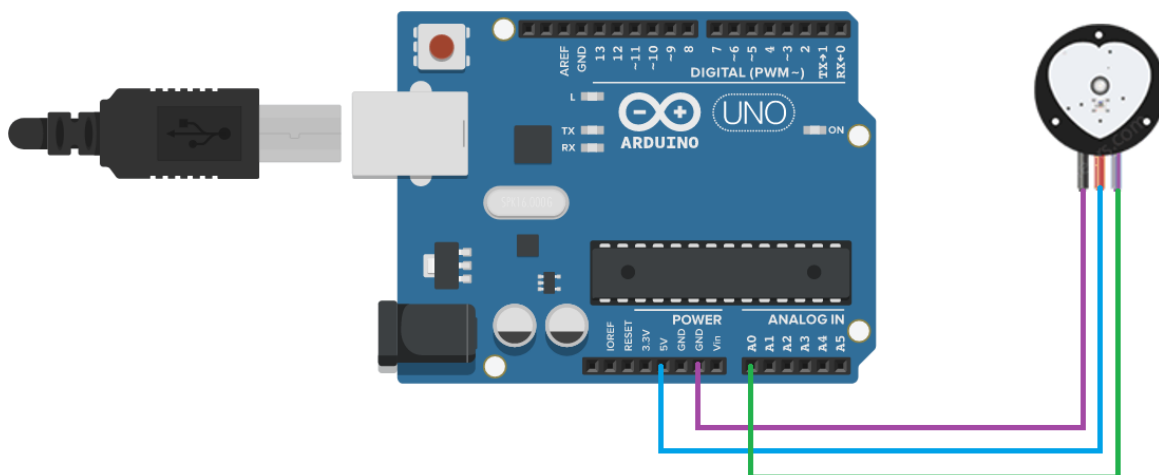
Senzor pulzu (pulse sensor) je nízkoenergetický snímač srdcovej frekvencie typu plug-and-play pre Arduino. Na prednej strane je otvor, v ktoro svieti zelená LED dióda. Pod ňou je snímač svetla, fotosenzor. Senzor pulzu priložíme na prst - na prst svieti zelené svetlo (~ 550 nm), ktoré sa odráža a množstvo odrazeného svetla sa meria pomocou fotosenzora.



Okysličená krv je červenšia ako neokysličená a preto lepšie absorbuje zelené svetlo. Ako je krv pumpovaná cez prst pri každom údere, mení sa množstvo odrazeného svetla a od toho závisí hodnota výstupu senzora. Táto metóda detekcie impulzov svetlom sa nazýva fotopletyzmozgram.

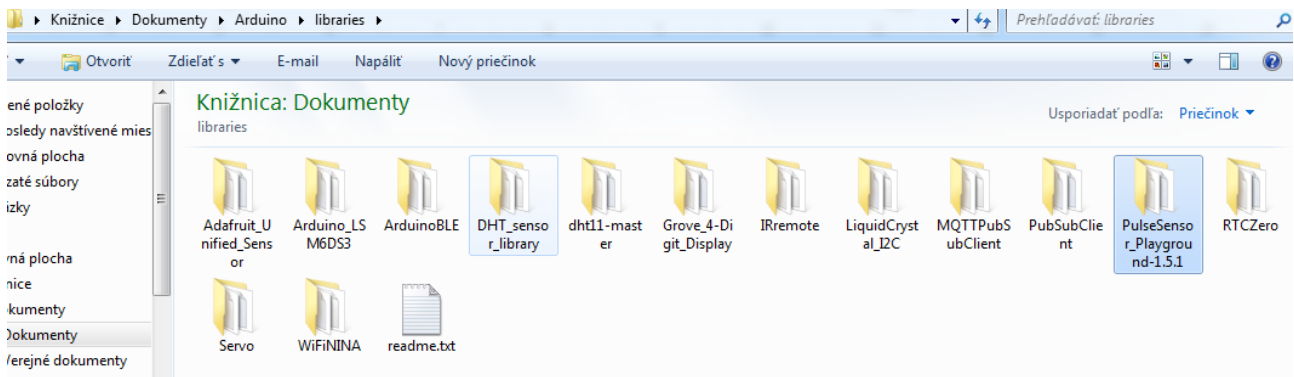


2. Schéma zapojenia



3. Inštalácia knižnice

Knižnicu **PulseSensor.h** stiahneme odiaľto <https://www.arduino-libraries.info/libraries/pulse-sensor-playground> ako zip súbor. Rozbalíme je do priečinka **Libraries**:



V správcovi knižníc si overíme, že sa doinštalovala:



V menu **Súbor-Príklady** nájdeme vzorové programy, ktoré môžeme vyskúšať a obmieňať.

4. Program na nastavenie prahovej hodnoty

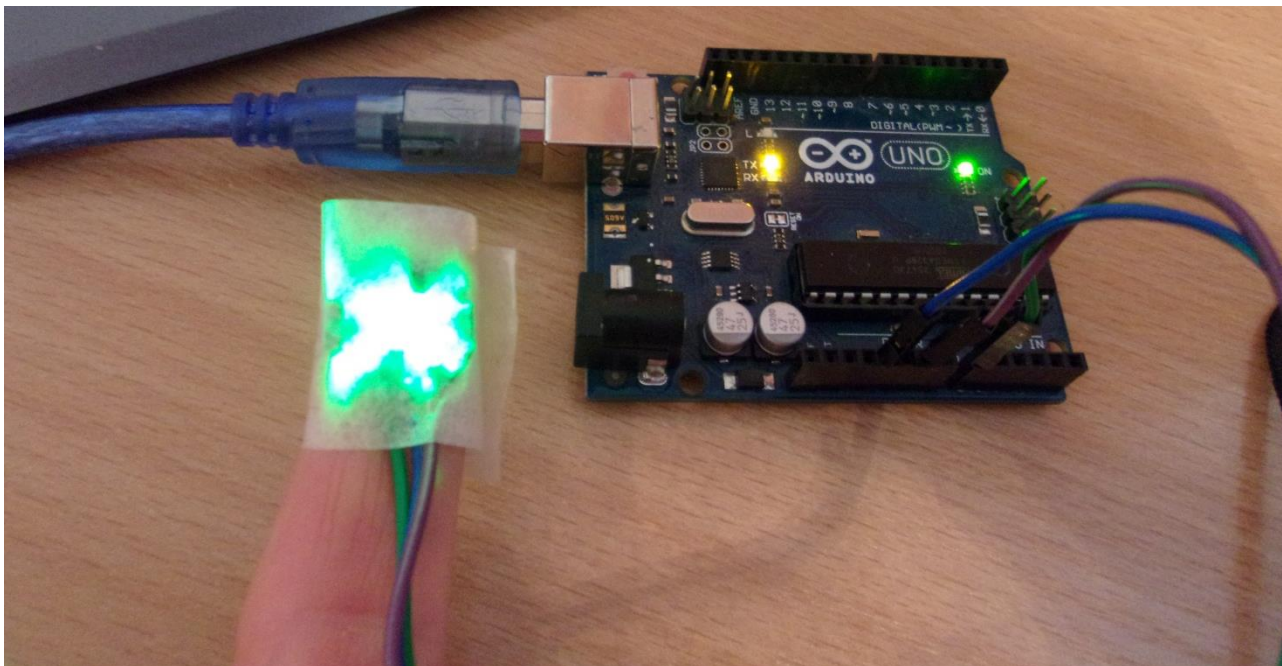
V tomto programe bliká interná LED dióda v rytme srdcového pulzu.

```
int PulseSensor = A0; // na A0 je pripojený vodič so signálom zo senzora pulzu
int LED13 = 13; // interná LED na Arduino doske

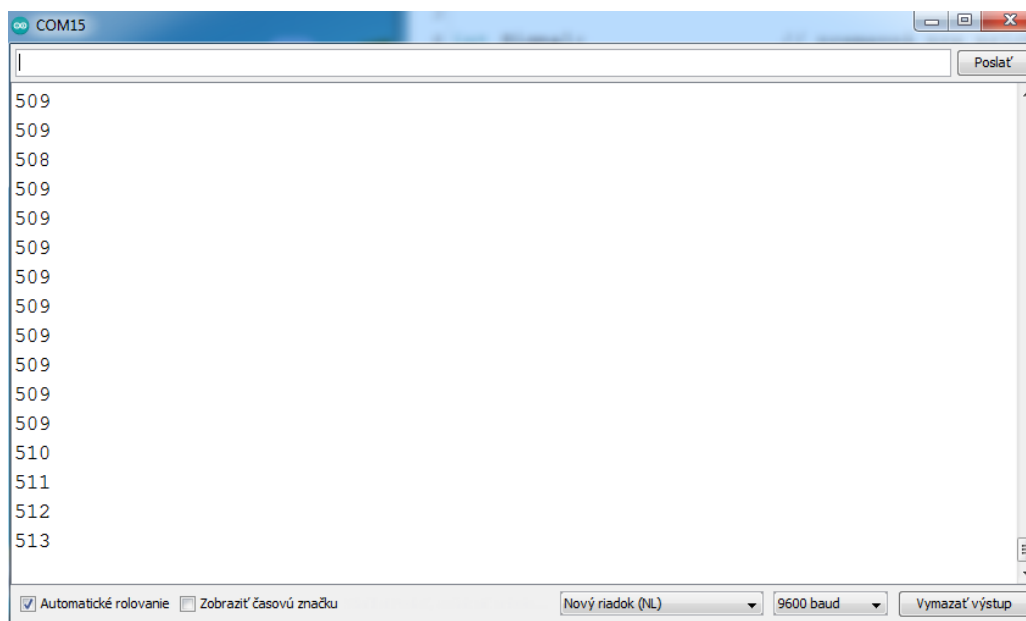
int Signal; // premenná pre prichádzajúce dáta v rozsahu 0-1024
int Threshold = 511; //prahová hodnota, ktorá určuje, čo sa bude počítat ako úder a
// čo sa bude ignorovať

void setup() {
  pinMode(LED13,OUTPUT); // nastavenie výstupu na internú LED
  Serial.begin(9600); // nastavenie rýchlosti sériovej komunikácie
}
void loop() {
  Signal = analogRead(PulseSensor); // prečítanie hodnoty zo senzora
  Serial.println(Signal); // výpis hodnoty
  if(Signal > Threshold){ // interná LED bude blikat v rytme srdcového pulzu
    digitalWrite(LED13,HIGH);
  } else {
    digitalWrite(LED13,LOW); //
  }
  delay(10);
}
```

Snímač prilepíme jemne na prst.

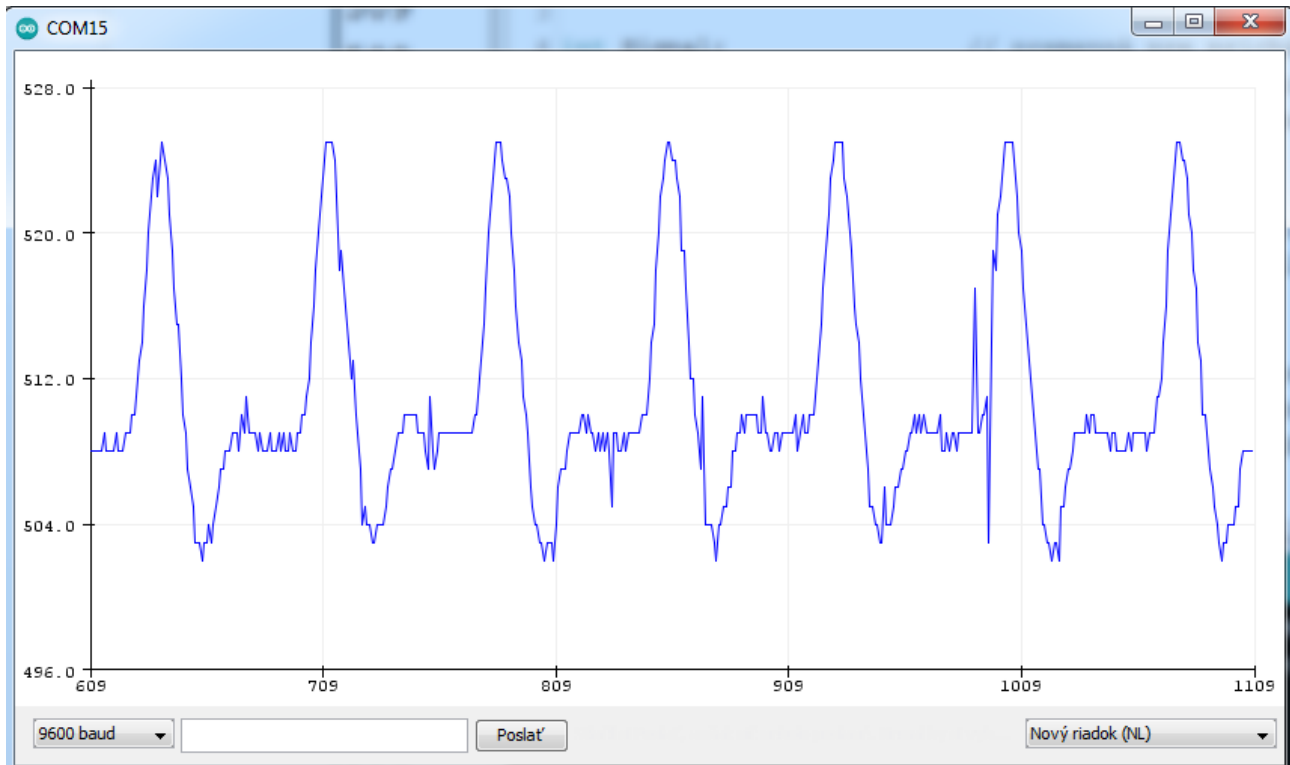


V programe treba experimentovať s prahovou hodnotou *Threshold* podľa výsledkov, ktoré vidíme v sériovom monitore. Na sériovom monitore si pozrieme namerané hodnoty a premennú *Threshold* nastavíme na prostrednú hodnotu zo všetkých nameraných.



Ak už máme zistenú a nastavenú hodnotu premennej *Threshold*, program znovu skompilujeme a nahráme do Arduina.

Namiesto sériového monitora si zapneme sériový plotter:



5. Program Getting BPM to monitor

BPM = *beats per minute* = počet úderov za minútu. Tento program vypočítava čas medzi impulzmi na získanie srdcovej frekvencie a zobrazuje ju na sériovom monitore. Hodnotu premennej *Threshold* treba nastaviť na základe skúsenosti z predchádzajúceho programu.

```

/* Getting_BPM_to_Monitor prints the BPM to the Serial Monitor, using the least lines of code
and PulseSensor Library.
 * Tutorial Webpage: https://pulsesensor.com/pages/getting-advanced
 *
-----Use This Sketch To-----
1) Displays user's live and changing BPM, Beats Per Minute, in Arduino's native Serial Monitor.
2) Print: "♥ A HeartBeat Happened !" when a beat is detected, live.
2) Learn about using a PulseSensor Library "Object".
4) Blinks LED on PIN 13 with user's Heartbeat.
-----*/

#define USE_ARDUINO_INTERRUPTS true // Set-up low-level interrupts for most accurate BPM math.
#include <PulseSensorPlayground.h> // Includes the PulseSensorPlayground Library.

// Variables
const int PulseWire = 0; // PulseSensor PURPLE WIRE connected to ANALOG PIN 0
const int LED13 = 13; // The on-board Arduino LED, close to PIN 13.
int Threshold = 512; // Determine which Signal to "count as a beat" and which to
ignore.

PulseSensorPlayground pulseSensor; // Creates an instance of the PulseSensorPlayground object
called "pulseSensor"

void setup() {
  Serial.begin(9600); // For Serial Monitor
  pulseSensor.analogInput(PulseWire);
  pulseSensor.blinkOnPulse(LED13); //blink Arduino's LED with heartbeat.
  pulseSensor.setThreshold(Threshold);

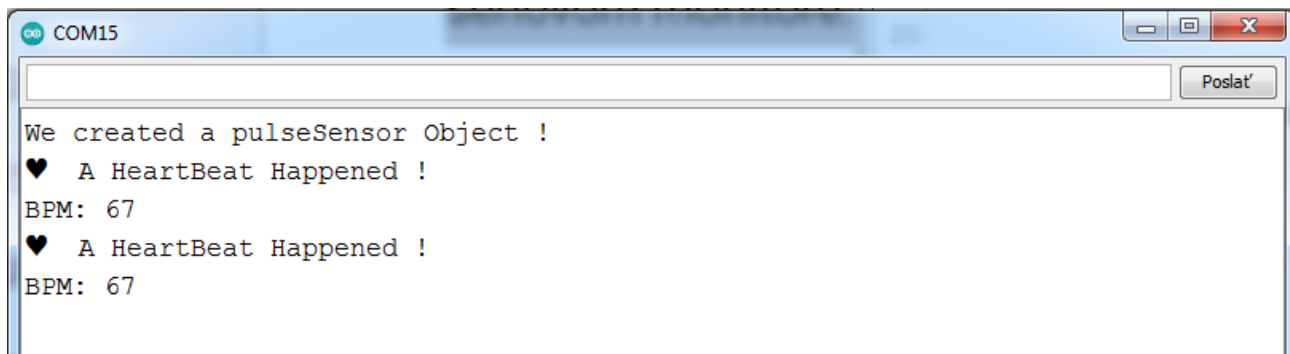
```

```
// Double-check the "pulseSensor" object was created and "began" seeing a signal.
if (pulseSensor.begin()) {
  Serial.println("We created a pulseSensor Object !"); //This prints one time at Arduino
power-up, or on Arduino reset.
}
}

void loop() {
  int myBPM = pulseSensor.getBeatsPerMinute(); // Calls function on our pulseSensor object that
returns BPM as an "int".

  // "myBPM" hold this BPM value now.
  if (pulseSensor.sawStartOfBeat()) { // Constantly test to see if "a beat happened".
    //Serial.println("♥ A HeartBeat Happened ! "); // If test is "true", print a message "a
heartbeat happened".
    //Serial.print("BPM: "); // Print phrase "BPM: "
    Serial.println(myBPM); // Print the value inside of myBPM.
  }
  delay(20); // considered best practice in a simple sketch.
}
```

Po nahratí programu do Arduina si zapneme sériový monitor:



The screenshot shows the Arduino Serial Monitor window titled 'COM15'. The output text is as follows:

```
We created a pulseSensor Object !
♥ A HeartBeat Happened !
BPM: 67
♥ A HeartBeat Happened !
BPM: 67
```

Ak chceme hodnoty odoslané z Arduina spracovávať v Node-RED, výstup zjednodušíme takto:

```
35 | if (pulseSensor.sawStartOfBeat()) { // Co
36 | //Serial.println("♥ A HeartBeat Happened ! "); //
37 | //Serial.print("BPM: "); //
38 | Serial.println(myBPM); // P:
39 | }
40 | delay(20); // considered best pra
41 | }
```

Arduino IDE môžeme zavrieť a spustiť si Node-RED.

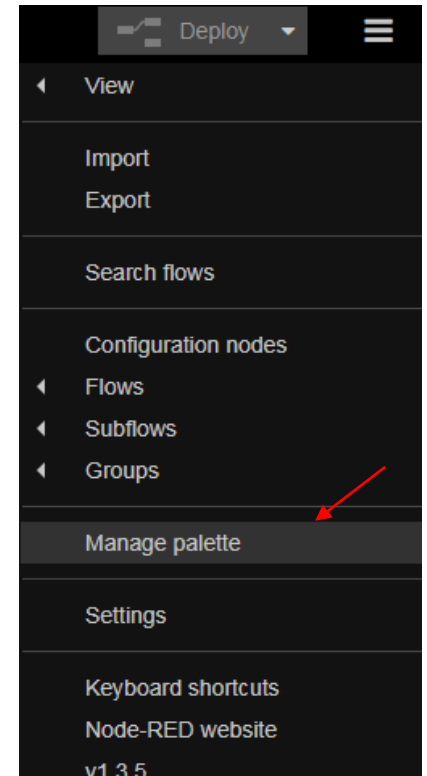
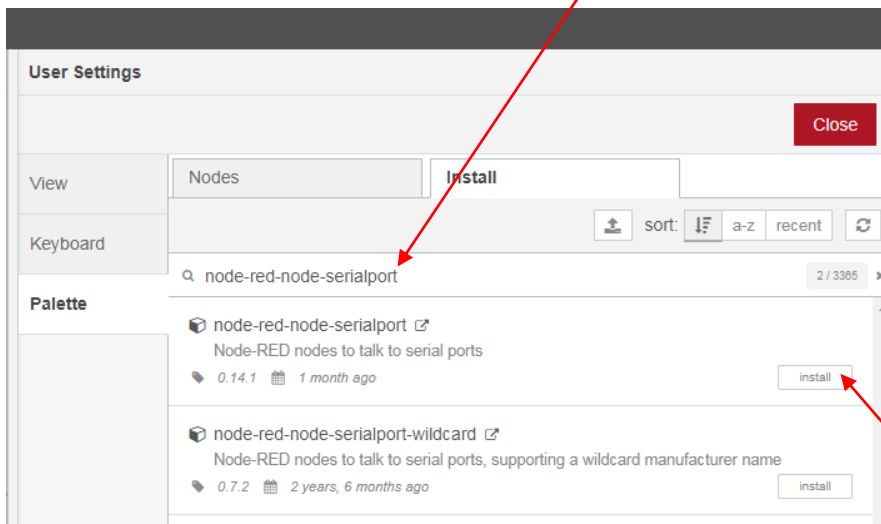
6. Node-RED

V Príkazovom riadku zadáme príkaz `node-red`

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

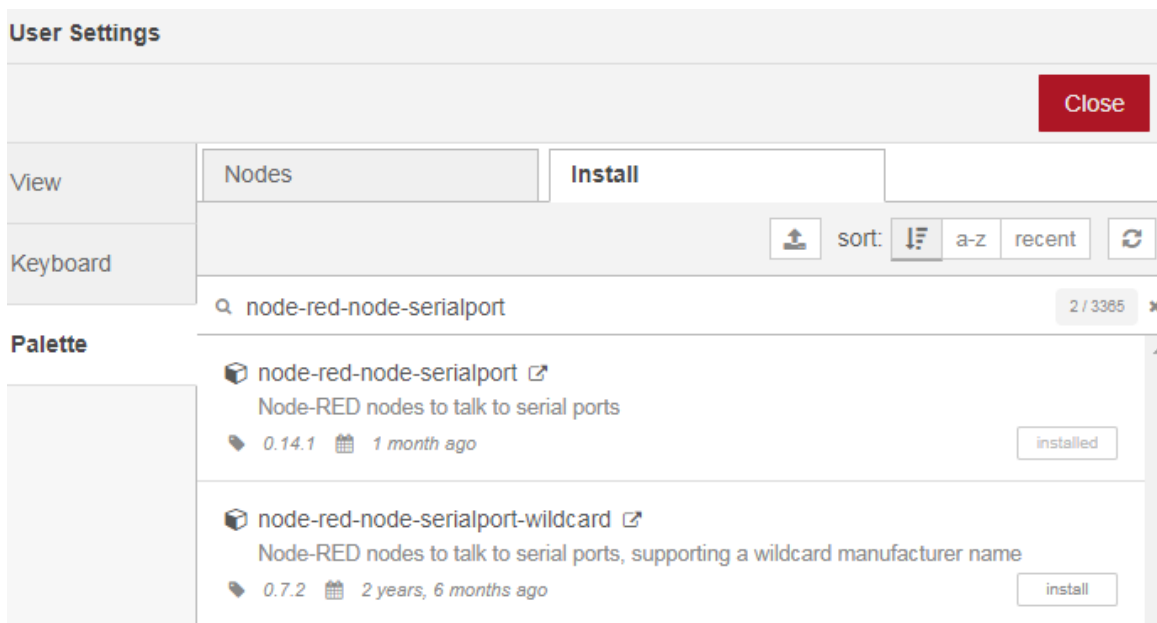
Node-RED nemá predinštalované uzly pre Arduino a Serial Port. Doinštalujeme ich pomocou Manažéra palet:

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

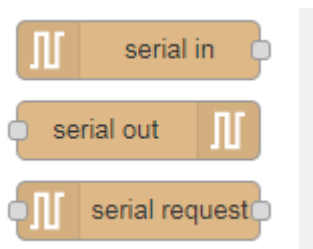


Túto nainštalujeme.

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, specifically the 'Install' tab. The search bar contains the text 'Arduino'. Below the search bar, a list of nodes is displayed in a 'Palette' view. Each node entry includes its name, a brief description, version number, and release date, along with an 'install' button. A red arrow points from the search bar to the search results. Another red arrow points from the 'install' button of the 'node-red-node-arduino' node to a text box below the screenshot.

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 firmata	0.3.1	1 year, 9 months ago	install

Túto
nainštalujeme.

Na pracovnú plochu vložíme uzol **serial in**:



Zadáme mu takéto vlastnosti:

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
9600	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

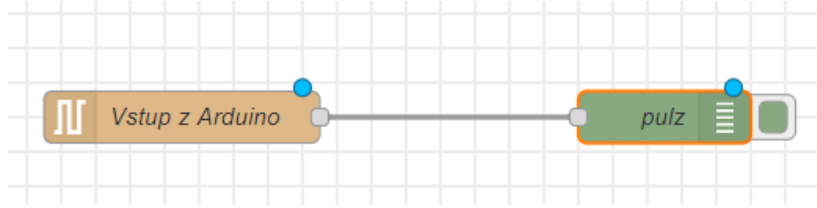
Add character to output messages

Request

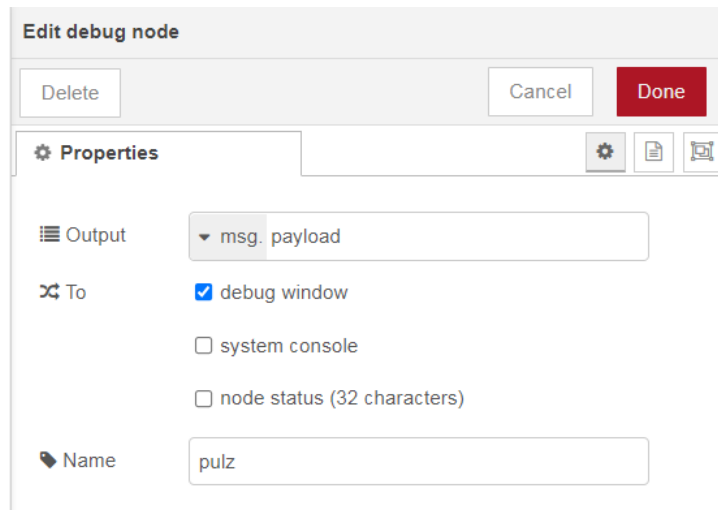
Default response timeout ms

Zadáme taký port, na ktorom máme pripojené Arduino..

Pridáme uzol **debug**:



Vlastnosti uzla **debug**:



Edit debug node

Delete Cancel Done

Properties

Output: msg. payload

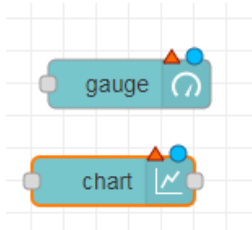
To: debug window
 system console
 node status (32 characters)

Name: pulz

Po kliknutí na **Deploy** sa v debug okne každú 1 sekundu zobrazujú hodnoty pulzu zo senzoru, pripojenom k Arduino (pozor, Arduino sme naprogramovali tak, že vysiela hodnoty každých 20 milisekúnd):



Pridáme uzly **gauge** a **chart**:



Vlastnosti uzla **gauge** (ciferník):

Edit gauge node

Delete Cancel Done

Properties

Group: [Pracovňa Mirka] Mirka - pulz ✎

Size: 8 x 8

Type: Donut

Label: pulz

Value format: {{value}}

Units: units

Range: min 0 max 150

Colour gradient:

Sectors: 0 ... optional ... optional ... 150

Name: pulz meradlo

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).

Nezabudneme nastaviť veľkosť a typ ciferníka.

Zadanie mena skupiny (group node):

Edit gauge node > Edit dashboard group node

Delete Cancel Update

Properties

Name: Mirka - pulz

Tab: Pracovňa Mirka ✎

Width: 20

Display group name

Allow group to be collapsed

Klikneme na ikonu ceruzky. Dostaneme sa do ďalšieho dialógového okna, kde vyberáme tabuľku. Ak tabuľku ešte nemáme vytvorenú, tak ju nemáme z čoho vybrať, takže ju musíme vytvoriť.

Nezabudneme nastaviť šírku tabuľky.

Výber alebo vytvorenie tabuľky (tab node):

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


Z dialógových panelov sa vraciame do vyššej úrovne cez tlačidlo **Update** resp. **Done**.

Vlastnosti uzla **chart** (graf):

Edit chart node


Delete Cancel Done

Properties

Group [Pracovňa Mirka] Mirka - pulz 

Size 8 x 8

Label pulz


Type  Line chart enlarge points

X-axis last 1 minute: OR 1000 points

X-axis Label HH:mm:ss as UTC

Y-axis min 0 max 200

Legend None Interpolate linear

Series Colours 

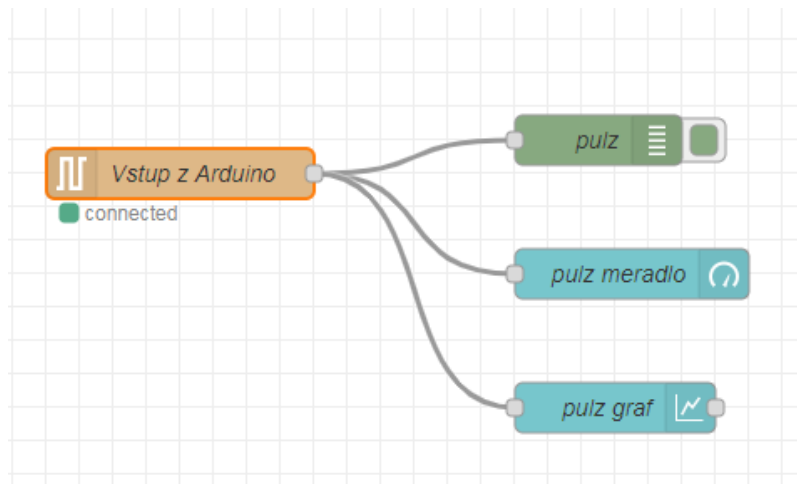
Blank label display this text before valid data arrives

Name pulz graf

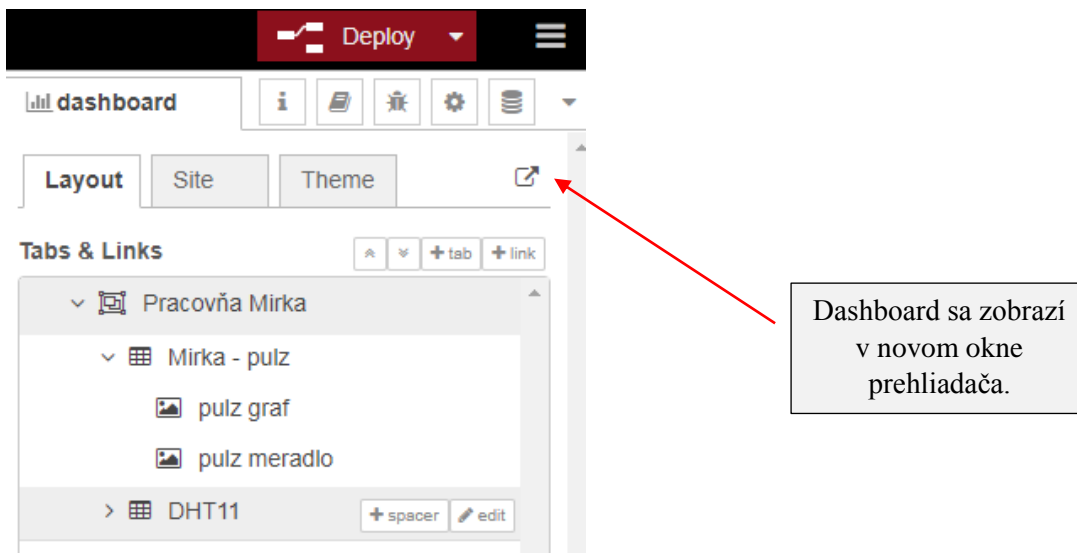
Klikneme na ikonu ceruzky. Dostaneme sa do ďalšieho dialógového okna, kde vyberáme tabuľku (jej názov je v tej hranatej zátvorke) a zadávame meno skupiny (to je uvedené za zátvorkou).

Nezabudneme nastaviť veľkosť a typ grafu.

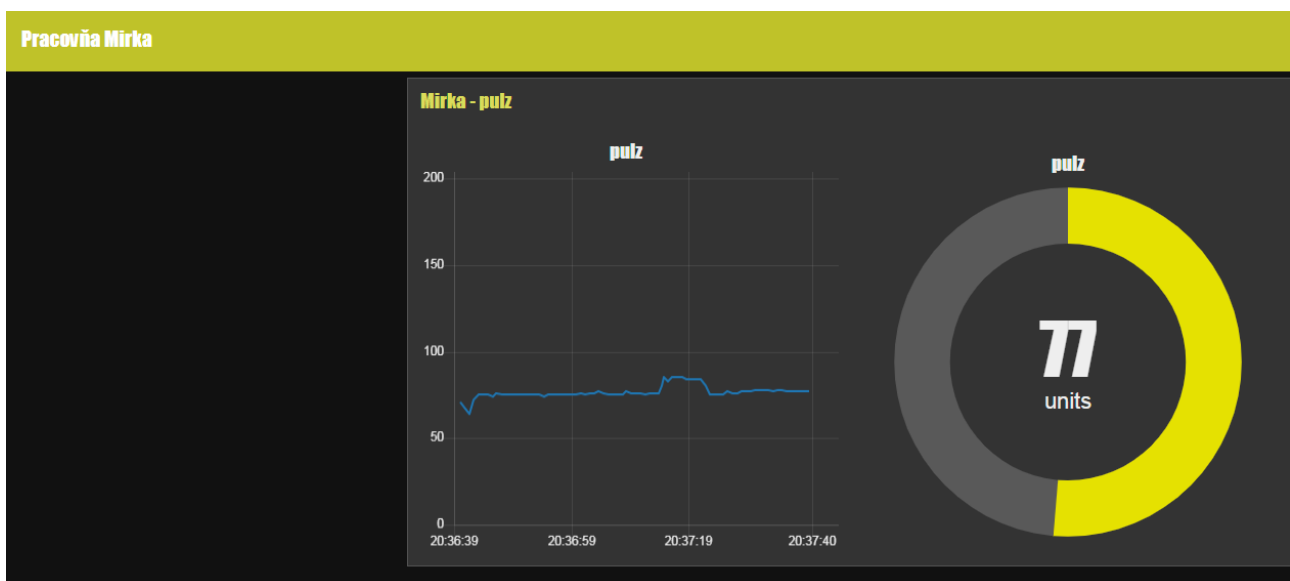
Takže máme vytvorený takýto **flow** (tok):



Po kliknutí na **Deploy** si zobrazíme **Dashboard** (palubnú dosku):

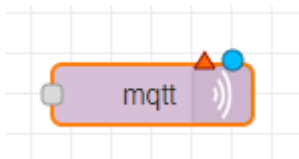


Výsledok:



7. Publikovanie na HiveMQ

Pridáme uzol **mqtt out**:



Vlastnosti uzla **mqtt out**:

Edit mqtt out node

Delete Cancel Done

Properties

Server Mirka

Topic pulz_Mirka

QoS Retain true

Name pulz odosielanie na HiveMQ

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

Nastavenie mqtt brokera:

Edit mqtt out node > Edit mqtt-broker node

Delete Cancel Update

Properties

Name Mirka

Connection Security Messages

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

Edit mqtt out node > **Edit mqtt-broker node**

Delete Cancel Update

Properties

Name Mirka

Connection Security **Messages**

Message sent on connection (birth message)

Topic pulz_Mirka Retain true

Payload msg.payload QoS 0

Message sent before disconnecting (close message)

Topic pulz_Mirka Retain

Payload končím QoS 0

Message sent on an unexpected disconnection (will message)

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

MQTT Websocket Client

www.hivemq.com/demos/websocket-client/ Ako začať

HIVEMQ Websockets Client Showcase

Connection

Host broker.mqttdashboard.com Port 8000 ClientID clientId-jtT5vdLFdt **Connect**

Username Password Keep Alive 60 SSL Clean Session

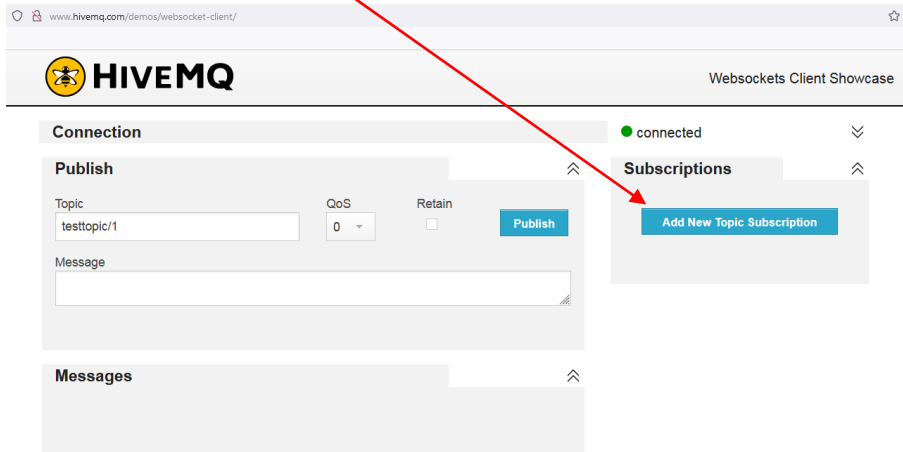
Last-Will Topic Last-Will QoS 0 Last-Will Retain

Last-Will Message

Publish Subscriptions

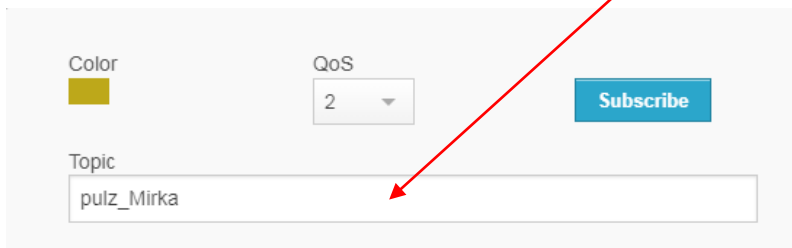
Messages

Po pripojení zadáme odber témy (topic):



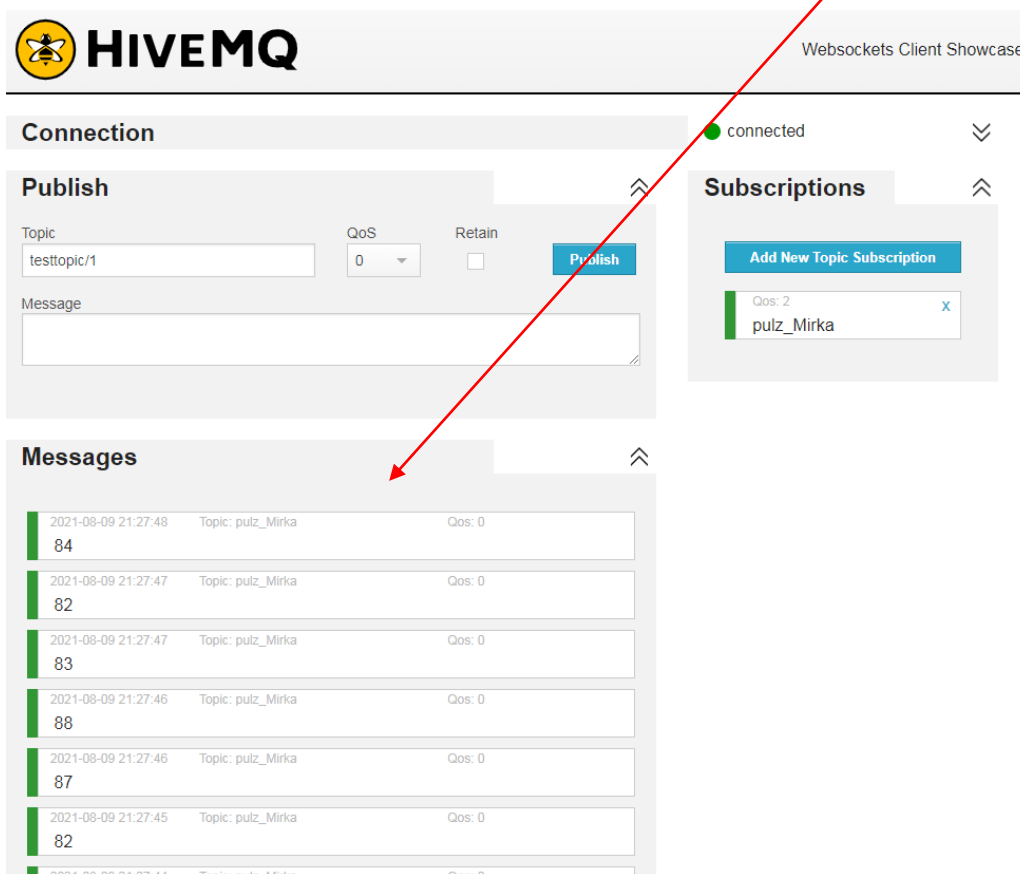
The screenshot shows the Hivemq Websockets Client Showcase interface. The 'Subscriptions' panel is visible, showing a button labeled 'Add New Topic Subscription'. A red arrow points from the text above to this button.

Zadáme ten istý topic, ktorý sme zadali aj v Node-Red:



The screenshot shows a Node-Red subscription node configuration. The 'Topic' field contains the text 'pulz_Mirka'. A red arrow points from the text above to this field.

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



The screenshot shows the Hivemq Websockets Client Showcase interface. The 'Messages' panel is visible, displaying a list of received messages. A red arrow points from the text above to the 'Messages' panel.

Timestamp	Topic	QoS
2021-08-09 21:27:48	pulz_Mirka	0
84		
2021-08-09 21:27:47	pulz_Mirka	0
82		
2021-08-09 21:27:47	pulz_Mirka	0
83		
2021-08-09 21:27:46	pulz_Mirka	0
88		
2021-08-09 21:27:46	pulz_Mirka	0
87		
2021-08-09 21:27:45	pulz_Mirka	0
82		
2021-08-09 21:27:44	pulz_Mirka	0