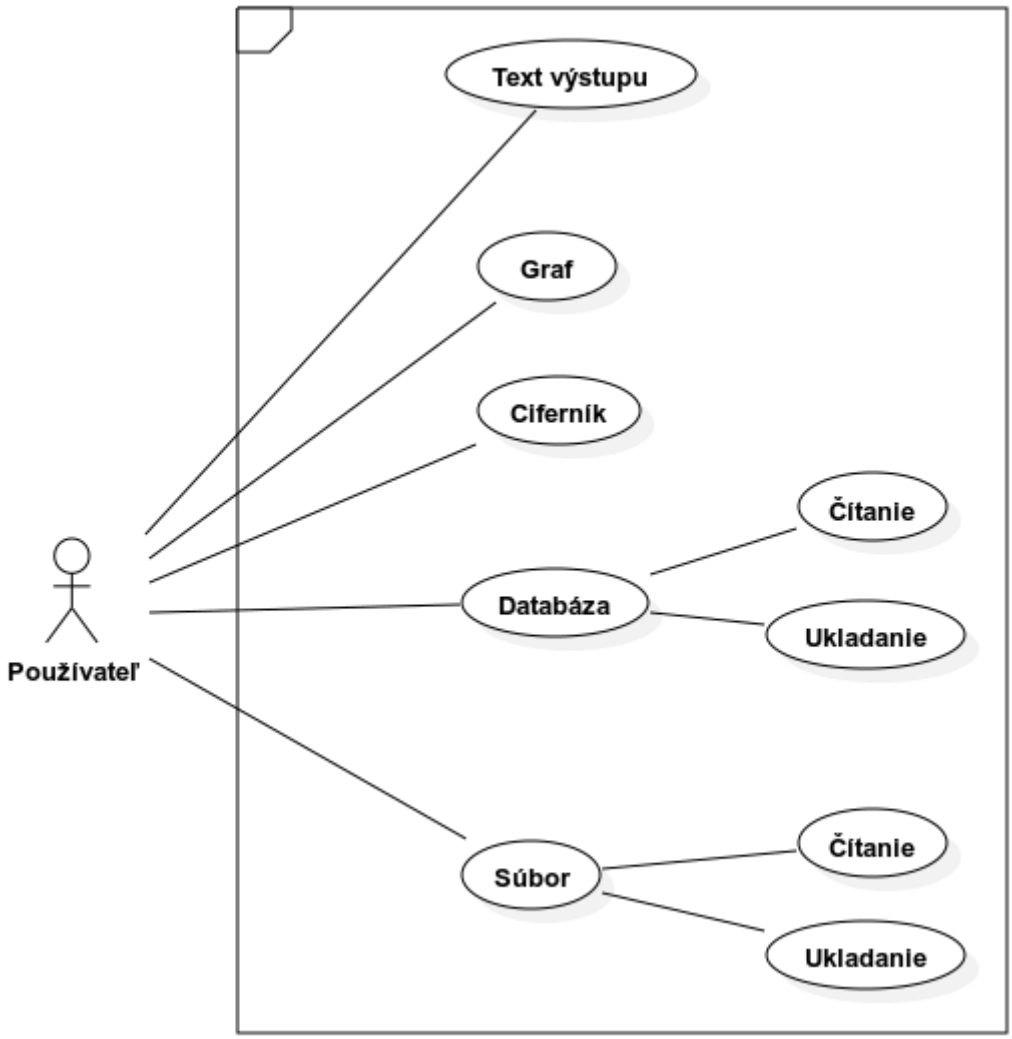
POIT - Pokročilé informačné technológie

PROJEKT: Riadenie napätia na zapojení

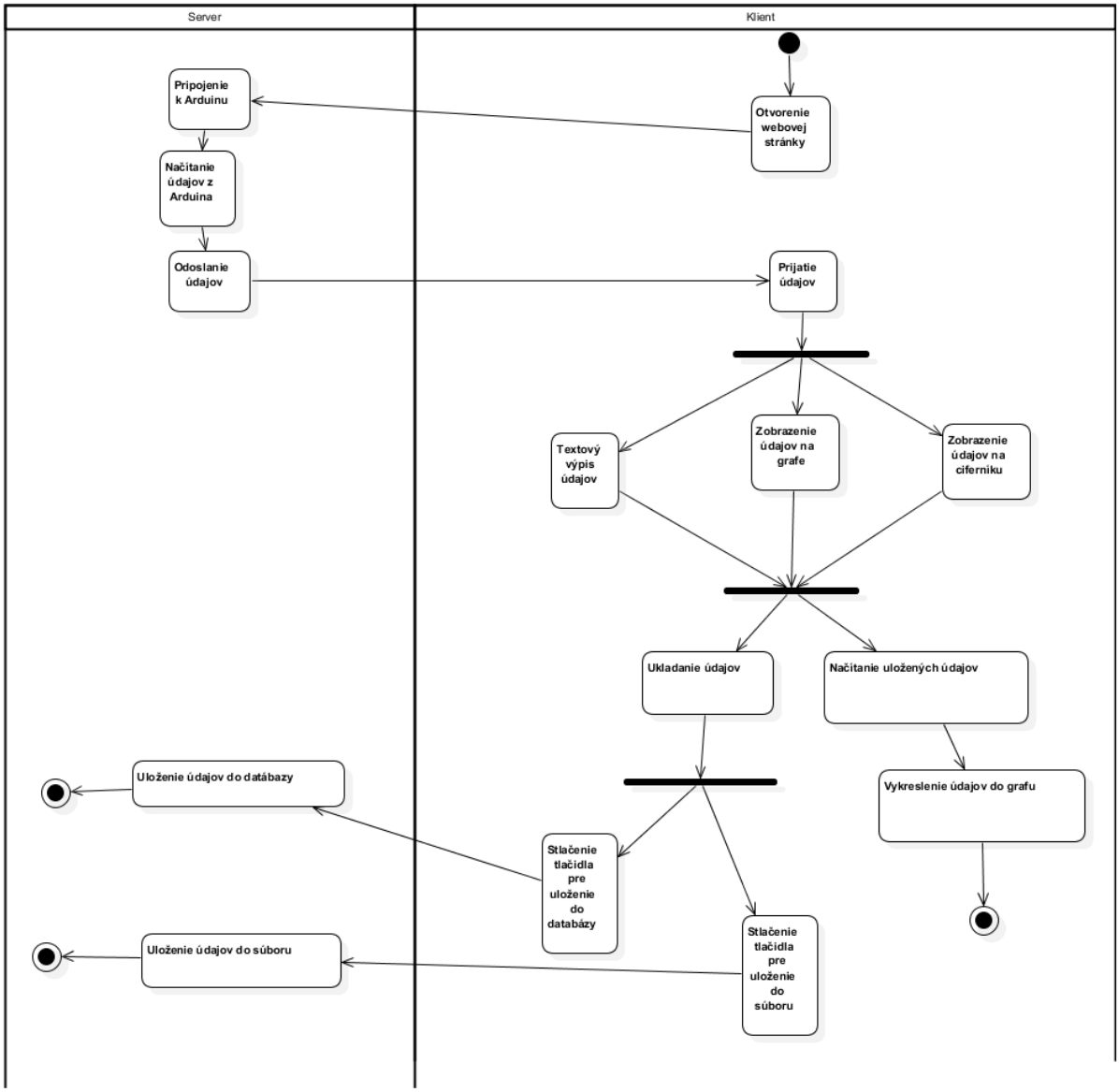
Úlohou nášho záverečného zadania z predmetu POIT bolo vytvoriť zapojenie zo súčiastok, ktoré sme mali k dispozícii a v takom tvare, aký nám bol zadaný prednášajúcim. Následne bolo potrebné vytvoriť pre túto schému ovládaciu webovú stránku a namerané údaje na nej zobrazovať. V našom prípade išlo o zapojenie obvodu tak, aby R1 < R2, C1 < C2 a meraná veličina bola napätie na C2.

1 Use Case Diagram



Obrázok 1 - Use Case Diagram

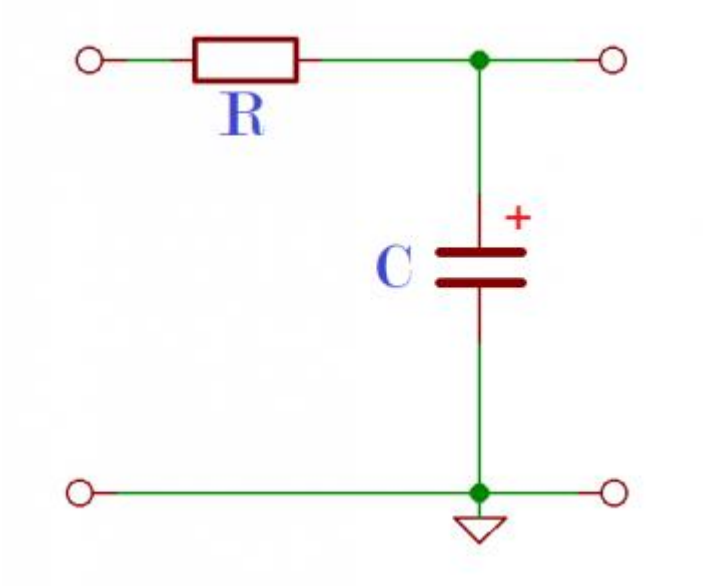
2 Activity Diagram



Obrázok 2 - Activity Diagram

3 Zapojenie obvodu

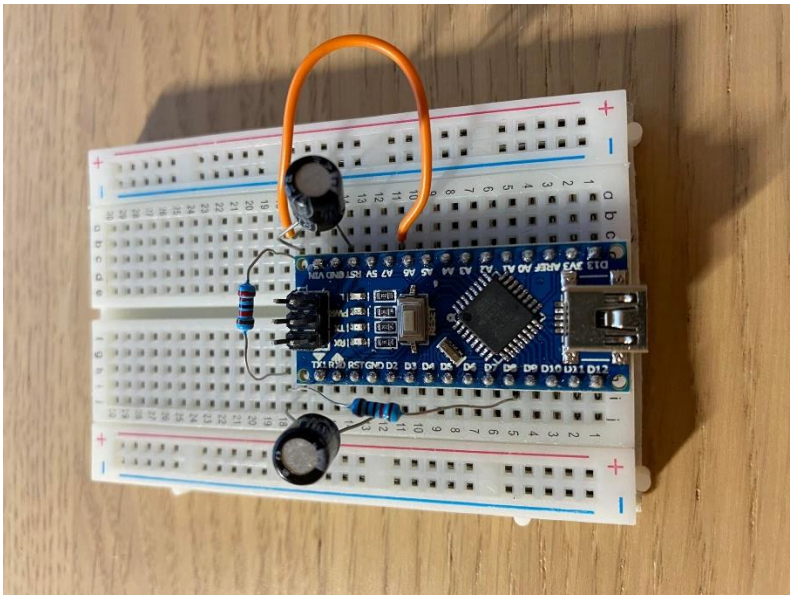
Ako základ obvodu sme použili schému pre riadenie sústavy 1. rádu, kde je medzi vstup a výstup zapojený rezistor, ku ktorému je paralelne pripojený kondenzátor.



Obrázok 3 - Sústava 1. rádu

K takto vytvorenému obvodu sme pripojili do série ďalší rezistor a k nemu paralelne kondenzátor. Pre správne zapojenie z nášho zadania sme museli zistiť veľkosť odporov jednotlivých rezistorov a veľkosť kapacity kondenzátorov. Nakoľko sme mali dostupný multimeter, odpory sme zmerali cez neho. Taktiež je možné veľkosti odporov zistiť pomocou ich čiarového označenia. Veľkosti kapacít kondenzátorov sú uvedené na jednotlivých kondenzátoroch.

$$R_1 = 10\ k\Omega$$
$$R_2 = 22\ k\Omega$$
$$C_1 = 100\ \mu F$$
$$C_2 = 220\ \mu F$$



Obrázok 4 - Zapojenie obvodu

Požiadavkou na meranie bolo merať napätie na kondenzátore C2, pripojili sme preto analógový vstup Arduina A7 na vstup kondenzátora. Tento vstup budeme používať ako voltmeter.

4 Programovanie Arduina

V novovytvorenom programe vo vývojovom prostredí pre Arduino sme vo funkcii setup() nastavili digitálny bod D8 ako výstup. Taktiež sme vytvorili Sériové pripojenie.

```
void setup()
{
  pinMode(8, OUTPUT);
  Serial.begin(9600);
}
```

Obrázok 5 - Nastavenie Arduina

Vo funkcii loop() sme z analógového vstupu A7 načítali nameranú hodnotu a upravili ju. Túto hodnotu sme vypísali do sériovej komunikácie.

Nakoľko je pri práci s kondenzátormi potrebné ich vybíjanie, vytvorili sme jednoduchý kód, ktorý v prípade, že sa kondenzátor nabíja, až kým nedosiahne hodnotu 4.5V, vysiela na digitálny výstup hodnota HIGH, v prípade že hodnota prekročí 4.5V, zmení sa nastavenie a na digitálny výstup za zapisuje hodnota LOW, až pokiaľ napätie neklesne pod hodnotu 0.1V.

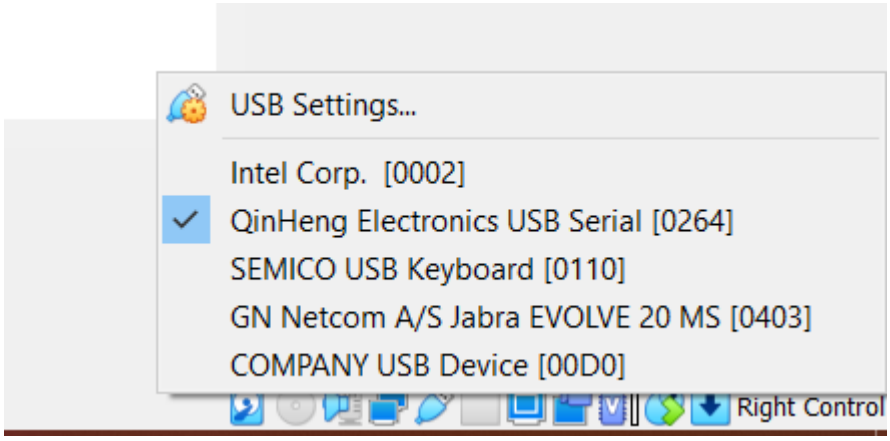
```
void loop(){
  float n1 = (float)analogRead(A7)*5/1023;

  if(n1 > 4.5){
    setting = 0;
  }else if(n1 < 0.1 ){
    setting = 1;
  }

  delay(100);
  if(setting){
    digitalWrite(8, HIGH);
  }else{
    digitalWrite(8, LOW);
  }
  delay(100);
  Serial.println(n1);
}
```

Obrázok 6 - Hlavná funkcia Arduina

Takto pripravený program sme zapísali do Arduina.

5 Prepojenie s Raspbianom

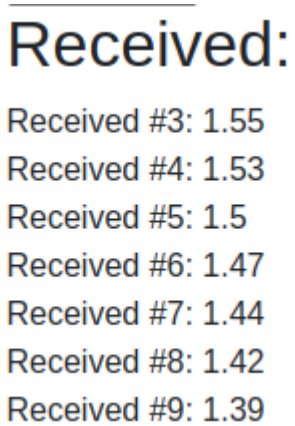
V prostredí VirtualBoxu je možné k virtuálnemu operačnému systému pripojiť USB zariadenia najjednoduchšie cez ikonku USB vpravo dole. Arduino je označené ako QinHeng Electronics USB Serial.



Obrázok 7 - Pripojenie Arduina k Raspbianu

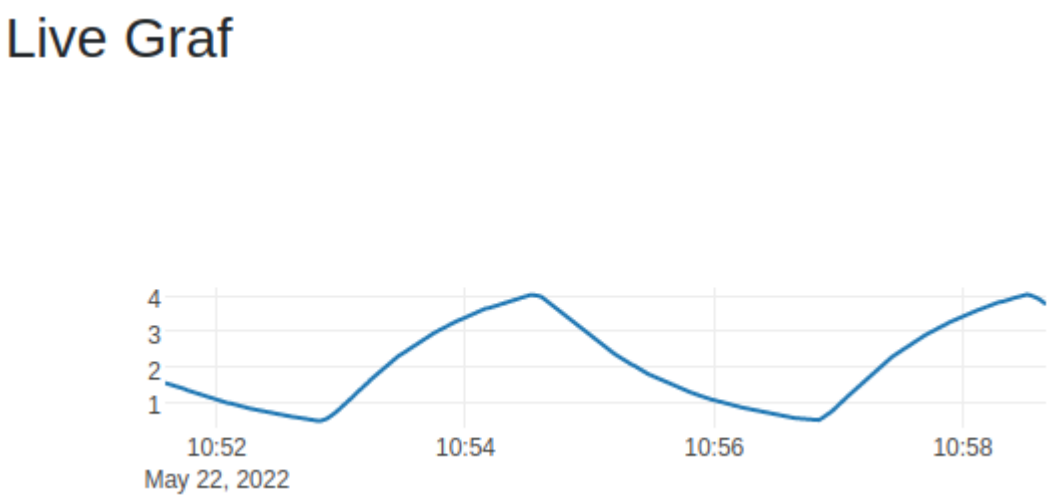
6 Vytvorenie webovej stránky a zobrazovanie údajov

Pre vytvorenie webovej stránky sme použili jedno z nami vypracovaných zadaní z cvičenia, konkrétne zadanie s ukladaním do DB. Pomocou Python knižnice Serial sme sa pripojili na zariadenie *dev/ttyUSBX* kde X označuje USB port kam sa Arduino pripojí. Tento port sa pri každom odpojení a pripojení Arduina mení. Následne sme v kóde nahradili funkciu Math.sin() funkciou *ser.readline()*, ktorá z pripojeného sériového rozhrania prečíta aktuálny riadok. Tento riadok je ešte potrebné upraviť cez *decode* a *rstrip*. Takto získame hodnotu aktuálneho napätia na kondenzátore C2. Nakoľko výpis hodnôt máme vypracovaný zo zadania, použijeme ho.



Obrázok 8 - Výpis hodnôt v textovej forme

Graf sa vytvorí pri otvorení stránky, pri bežiacej komunikácii sa následne cez funkciu *Plotly.extendTraces* rozšíri o nameranú hodnotu.



Obrázok 9 - Zobrazenie aktuálnych hodnôt na grafe

Rovnako sa tieto hodnoty zobrazia aj na ciferníku, ktorý sme upravili tak, aby maximálna hodnota bola 5.

Obrázok 10 - Zobrazenie aktuálnych hodnôt na ciferníku

7 Práca s DB

Nakoľko vychádzame zo zadania, v ktorom bolo úlohou zapisovať dáta do databázy, použili sme tento kód, v ktorom bolo nutné upraviť komunikáciu používateľa a aplikácie. Na používateľovej strane sme vytvorili tlačidlo, ktorým používateľ môže zapnúť zapisovanie do DB. Po stlačení sa pomocou socketio vytvorí požiadavka na server, v správe tejto požiadavky je buď hodnota *start* pre začatie nahrávania hodnôt pre zápis, alebo hodnota *stopDB*, ktorá slúži na ukončenie nahrávania a zápis do DB. Na serverovej strane sme v hlavnej funkcii *background_thread* upravili sledovanie, či sa získaná hodnota z *args* rovná *start* alebo *stopDB*.

```
if record == 'start':
  dataDict = {
    "t": timestamp,
    "y": prem,
    }
  dataList.append(dataDict)
else:
  if len(dataList)>0:
    print(str(dataList))
    fuj = str(dataList).replace("'", "\"")
    print(fuj)
    if(record == 'stopDB'):
        cursor = db.cursor()
        cursor.execute("SELECT MAX(id) FROM graph")
        maxid = cursor.fetchone()
        cursor.execute("INSERT INTO graph (id, hodnoty) VALUES (%s, %s)", (maxid[0] + 1, fuj))
        db.commit()
```

```
    idsDB = cursor.fetchall()
    fo = open("static/files/test.txt", "
    idsFILE = len(fo.readlines())
    print(idsFILE)
    emit('my_response',
         {'data': message['value'], 'cou

@socketio.on('db_event', namespace='/tes
def db_message(message):
    print(message['value'] == 'startDB')
    if(message['value'] == 'startDB'):
        session['record'] = 'start'
    else:
        session['record'] = 'stopDB'
```

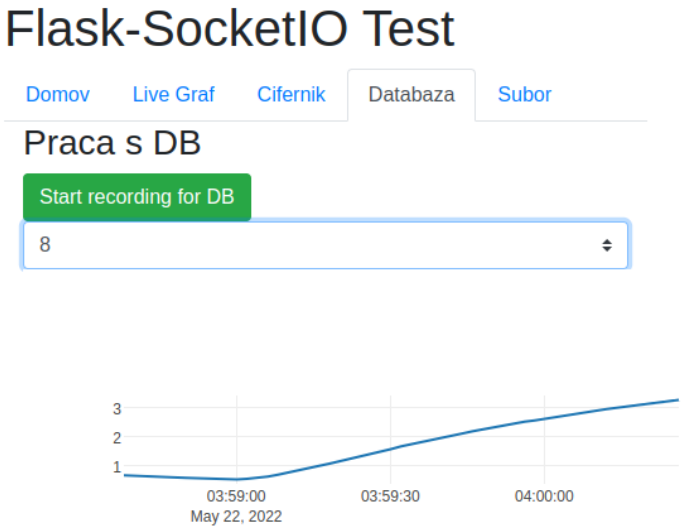
Obrázok 11 - Zápis do DB na serverovej strane

Pre čítanie z DB bolo potrebné získať aké ID záznamov v tabuľke je možné volať. Pridali sme preto do kódu na serverovej strane *SELECT* z DB v ktorom získame všetky dostupné ID záznamov. Tieto záznamy sme preniesli na klientskú stranu, kde sme ich pridali ako možnosti do HTML select-u.

```
@socketio.on('my_event', namespace='/test')
def test_message(message):
    session['receive_count'] = session.get('receive_count', 0) + 1
    cursor = db.cursor()
    cursor.execute("SELECT id FROM graph")
    idsDB = cursor.fetchall()
    emit('my_response',
         {'data': message['value'], 'count': session['receive_count'], 'idsDB': idsDB, 'initial': 1})
```

Obrázok 12 - Získanie dostupných ID z DB

Pokiaľ používateľ zmení hodnotu tohto selectu, odošle sa na server požiadavka o dáta s DB, kde v správe požiadavky je uvedené ID záznamu. Server tento záznam získa a vráti ako odpoveď. Nakoľko ukladáme záznamy do DB vo formáte JSON, na klientskej strane je s nimi jednoduché pracovať. Získané dáta vykreslíme do grafu.



Obrázok 13 - Klientská strana práce s DB

8 Práca so súborom

Pri práci so súborom sme vychádzali z kódu pre prácu s DB, nakoľko všetky volania klienta na server sú z klientského hľadiska takmer rovnaké, na serverovej strane sme namiesto získavania ID záznamov z DB získavali počet riadkov v súbore a následne jednoduchým for cyklom od 0 po počet riadkov vytvárali možnosti v HTML select-e. Po zmene hodnoty selectu sa zo súboru prečíta zadaný riadok. Hodnoty sú v súbore ukladané rovnako ako v DB, teda vo formáte JSON, preto aj čítanie je rovnaké.

```
@socketio.on('file_event', namespace='/test')
def file_message(message):
    if(message['value'] == 'startFILE'):
        session['record'] = 'start'
    else:
        session['record'] = 'stopFILE'

@socketio.on('get_file_event', namespace='/test')
def get_file_message(message):
    fo = open("static/files/test.txt", "r");
    rows = fo.readlines()
    data = rows[int(message['file_id'])-1]
    emit('file_response', {'data': data})
```

Obrázok 14 - Práca so súborom

Záver

Našou úlohou bolo monitorovať hodnotu napätia na kondenzátore C2 v zadanom zapojení dvoch rezistorov a dvoch kondenzátorov, medzi ktorými musí platiť vzťah, že hodnoty R1 < R2 a C1 < C2. Vytvorili sme preto takéto zapojenie pomocou dosky Arduino Uno, vytvorili program, pomocou ktorého meriame napätie na C2 a na základe tohto napätia meníme hodnotu digitálneho vstupu do obvodu z HIGH na LOW a opačne.

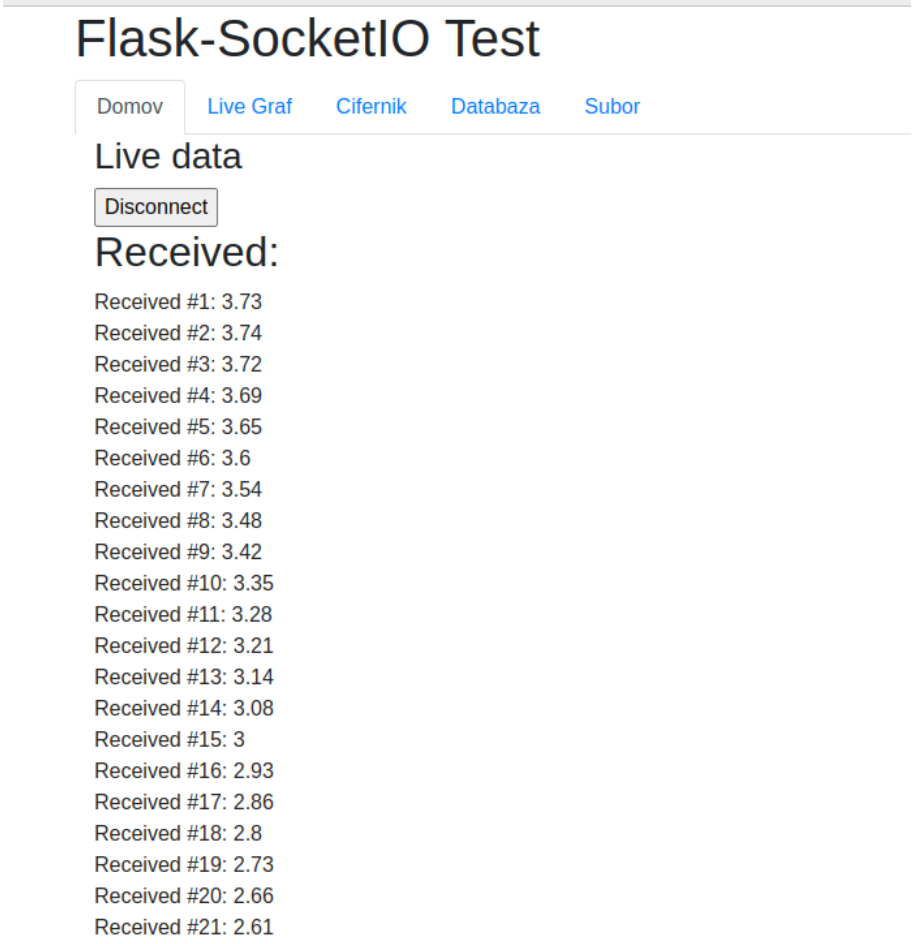
Vytvorili sme taktiež webovú aplikáciu, ktorá je funguje v prostredí Raspbian vo VirtualBox-e. Webová aplikácia pomocou sériového portu komunikuje s Arduinom, číta z neho údaje a používateľovi ich zobrazuje v textovom výpise, pomocou grafu a pomocou ciferníku.

Údaje si používateľ môže namerať a uložiť do databázy a do súboru, z ktorých je možné spätne tieto údaje získať a vykresliť na grafe.

Aplikáciu sme priebežne archivovali na GIT-e s linkom: https://github.com/veve107/poit . Záznam z práce je možné nájsť na videu tu.

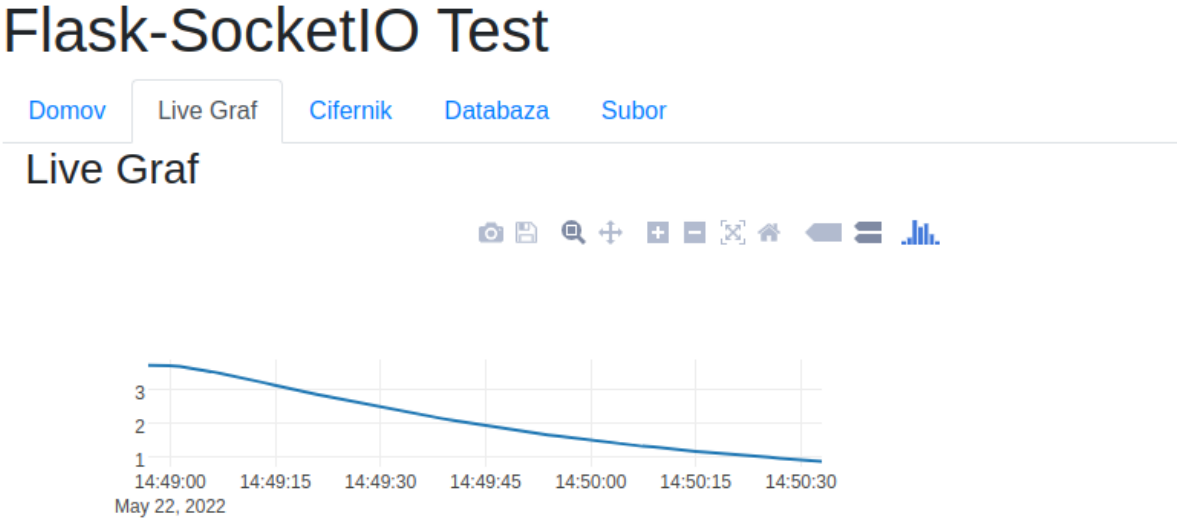
Používateľská príručka

Po otvorení stránky používateľ automaticky vidí namerané dáta v textovom formáte.



Obrázok 15 - Domovská stránka

Po kliknutí na tab Live Graf vidí namerané hodnoty zobrazené na grafe.

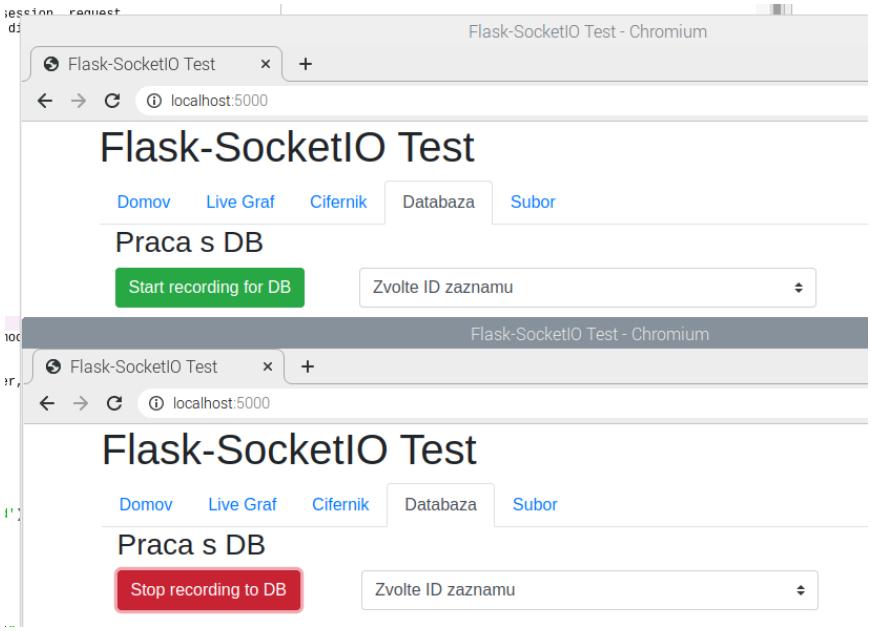


Obrázok 16 - Hodnoty na grafe

Po kliknutí na tab Ciferník sa používateľovi zobrazí ciferník s aktuálnou nameranou hodnotou.

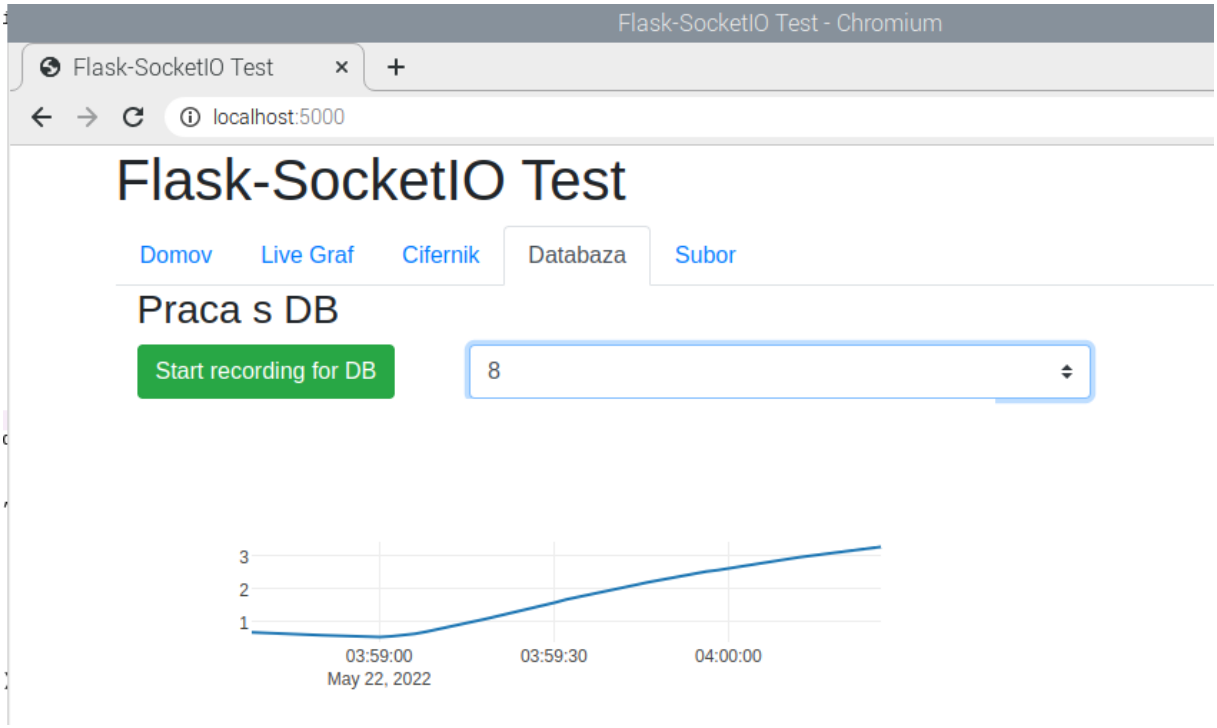
Obrázok 17 – Ciferník

Na tabe Databaza môže používateľ kliknúť na tlačidlo Start recording for DB, pomocou ktorého začne zaznamenávať údaje. Tlačidlu sa zmení farba na červenú a text na Stop recording for DB. Po stlačení tlačidla znova sa zaznamenávanie ukončí a údaje sa zapíšu do DB.



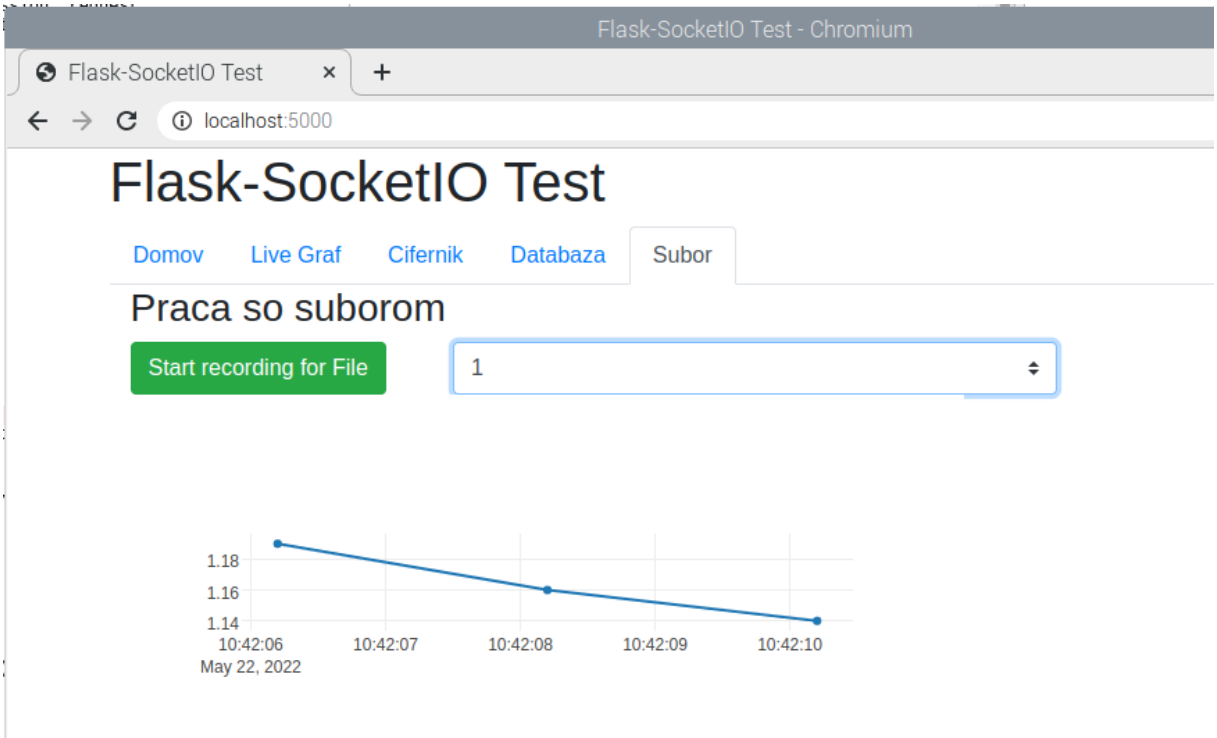
Obrázok 18 - Zaznamenávanie do DB

Používateľ môže na tomto tabe taktiež zobraziť už predtým uložené údaje, výber toho ktoré chce zobraziť je realizovaný cez select box. Dáta sa zobrazia v grafe.



Obrázok 19 - Zobrazenie uložených dát v DB

Rovnako môže používateľ pristupovať aj k zápisu a čítaniu zo súboru na tabe Subor.



Obrázok 20 - Zobrazenie uložených dát v súbore