SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

POIT

Monitorovanie a regulácia rezistora v RLC obvode

Martin Košický

1 Úvod

Cieľom zadania je monitorovať resp. riadiť signály získané z reálnych senzorov resp. simulačných a virtuálnych prostredí. Monitorovanie resp. riadenie sa má uskutočňovať prostredníctvom webovej aplikácie, aby bola naplnená koncepcia IoT. Hardvér a softvér, ktorý bol použitý v tomto zadaní je opísaný nižšie.

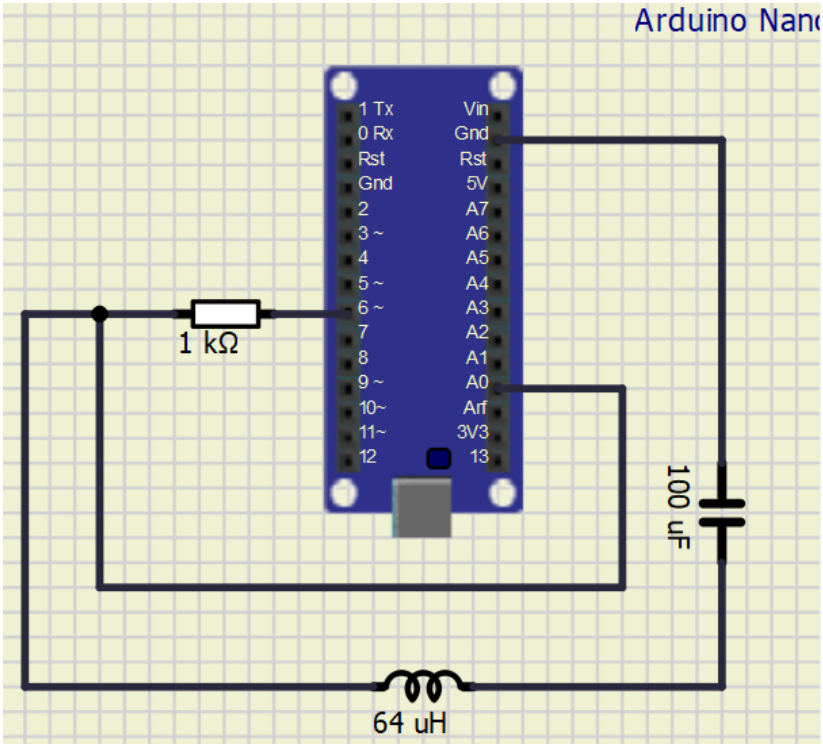
Táto dokumentácia obsahuje vývojovú a užívateľskú časť opisujúcu reguláciu resp. čítanie dát rezistora zapojeného v RLC obvode.

Pre vyššie popísaný systém sme použili nasledujúce súčiastky a rozhrania:

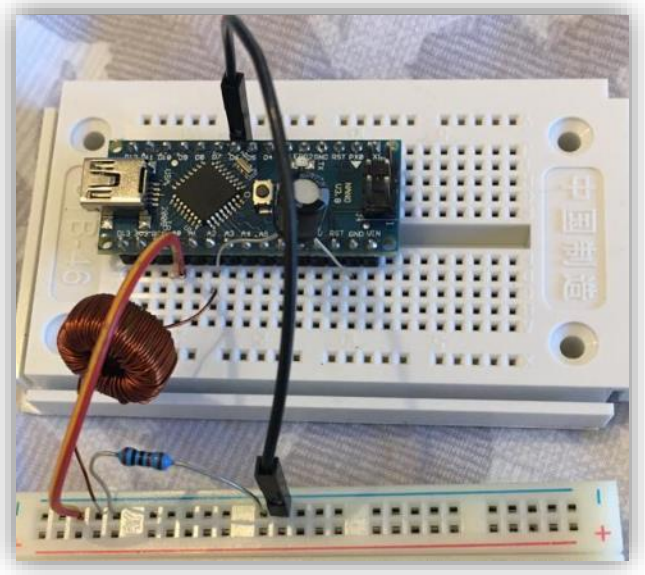
1. Vývojovú platformu Arduino Nano.
2. Virtuálnu platformu Raspberry Pi.
3. Rezistor 1 kOhm.
4. Vinutá cievka 64uH.
5. Kondenzátor 100 uF.

2 Zapojenie obvodu

Na Obr.1 je znázornené zapojenie Arduina Nano s RLC obvodom sústredeným na monitorovanie resp. ovládanie sériovo zapojeného rezistora.



Obr.1.: Schéma zapojenia obvodu



Obr.2.: Reálne zapojenie obvodu

2.1 Kód Arduino

Pri našom zadaní sme využili reálnu HW platformu Arduina. Kde sme v prvej časti zadefinovali sériovú komunikáciu medzi Arduinom a PC (serverom). Následne sme zadefinovali čítanie analógových hodnôt v porte A0. Z tohto portu čítame hodnotu signálu na rezistore. Následne využívame A/D prevodník, ktorý priradzuje analógovej hodnote jej digitálny ekvivalent. Napríklad hodnote 255 v digitálnej oblasti, zodpovedá hodnota 1024 v tej analógovej. A tento jav sa vykonáva v slučke.

arduino_kod

```
int b;
int constant;


void setup() {
  Serial.begin(9600);
  Serial.setTimeout(1);
  pinMode(6, OUTPUT);
   constant = 1;
}

void loop() {

  int analogPin = analogRead(A0);
  analogWrite(6 , constant);

  //Serial.println("Rezistor:");
  Serial.println(analogPin);


  if (Serial.available()>0){
  b = Serial.readString().toInt();
  Serial.println(b);
  constant=b;
  }
delay(500);
}
```

Obrázok 3.: Zdrojový kód nahratý do Arduina

3 Serverové rozhranie (Python)

V tejto kapitole bude po častiach zobrazený a popísaný kód na vytvorenie serverovej komunikácie v jazyku Python.

- Na začiatok sme si importovali potrebné knižnice a funkcie, ktoré zabezpečujú potrebnú sériovú komunikáciu, flasky a podobne:

```
from threading import Lock
from flask import Flask, render_template, session, request, jsonify, url_for
from flask_socketio import SocketIO, emit, disconnect
import time
import random
import math
import serial
```

- Po vytvorení komunikácie medzi serverom a Arduinom a vykonaní prvého vlákna dochádza k načítavaní hodnôt z Arduina a priradenie ich do premennej hodnota.

```
    def write_read(x):
        if B!=A:
            print("Sent value")
            arduino.write(bytes(x, 'utf-8'))
            time.sleep(1)
            hodnota = arduino.readline().strip()
            return hodnota
        else:

            hodnota = arduino.readline().decode().strip('\r\n')
            hodnota = format(hodnota)
            return hodnota
```

- V tejto časti dochádza k načítavaniu vstup od užívateľa, akým je začiatok zasielania údajov a reakciu na zmenu amplitúdy. Rovnako dochádza k vypisovaniu hodnôt.

```
while True:
    if args:
      A = dict(args).get('A')
    else:
      A = 1

    y = dict(args).get('start')
    A=str(A)

    socketio.sleep(1)
    count += 1
    print("A=",A)



    outa = write_read(A)
    outa = str(outa)

    B = A

    print("Go Button",y)
    if y =="1":
        socketio.emit('my_response',
                      {'data': outa, 'count': count},
                      namespace='/test')
```

- V poslednej časti dochádza k definovaniu, priradeniu a vykonávaniu funkcií pre tlačidla „Start“, „Stop“, „Connect“, „Disconnect“.

```python
@socketio.on('disconnect_request', namespace='/test')
def disconnect_request():
    session['receive_count'] = session.get('receive_count', 0) + 1
    emit('my_response',
         {'data': 'Disconnected!', 'count': session['receive_count']})
    disconnect()

@socketio.on('start', namespace='/test')
def db_message(message):
    session['start'] = message['value']

@socketio.on('stop', namespace='/test')
def db_message(message):
    session['start'] = message['value']

@socketio.on('connect', namespace='/test')
def test_connect():
    global thread
    with thread_lock:
        if thread is None:
            thread = socketio.start_background_task(target=background_thread, args=session._get_current_
    emit('my_response', {'data': 'Connected', 'count': 0})

@socketio.on('disconnect', namespace='/test')
def test_disconnect():
    print('Client disconnected', request.sid)

if __name__ == '__main__':
    socketio.run(app, host="0.0.0.0", port=80, debug=True)
```

4 Užívateľské rozhranie (Klientska časť)

Užívateľská časť je realizovaná prostredníctvom webovej stránky (http:localhost), kde dochádza k prepojeniu so serverom. V tejto časti budú okomentované najdôležitejšie časti html. kódu a náhľad do užívateľského prostredia.

- Vytvorenie funkcie otáčkomera (ciferníka)

```javascript
var gauge = new RadialGauge({
    renderTo: 'canvasID',
    width: 300,
    height: 300,
    units: "Data",
    minValue: 0,
    maxValue: 1000,
    majorTicks: [
        "0",
        "100",
        "200",
        "300",
        "400",
        "500",
        "600",
        "700",
        "800",
        "900",
        "1000"
    ],
```

- Vytvorenie funkcie grafu

```
73      socket.on('my_response', function(msg) {
74        console.log(msg.data);
75        $('#log').append('Received #'+msg.count+': '+msg.data+'<br>').html();
76
77        x.push(parseFloat(msg.count));
78        y.push(parseFloat(msg.data));
79        trace = {
80            x: x,
81            y: y,
82        };
83        layout = {
84          title: 'Data',
85          xaxis: {
86              title: 'x',
87          },
88          yaxis: {
89              title: 'y',//range: [-1,1]
90
91          }
92        };
93        console.log(trace);
94        var traces = new Array();
95        traces.push(trace);
96        Plotly.newPlot($('#plotdiv')[0], traces, layout);
97
98        console.log(msg.data);
99        $('#log').append('Received #'+msg.count+': '+msg.data+'<br>').html();
100       gauge.value = msg.data;
101       });
```

- Zasielanie a komunikácia so serverom

```
105     $('form#emit').submit(function(event) {
106         socket.emit('my_event', {value: $('#emit_value').val()});
107         return false; });
108
109
110
111     $('#start').click(function(event) {
112         console.log($('#start').val());
113         socket.emit('start', {value: $('#start').val()});
114         return false; });
115
116
117
118     $('#stop').click(function(event) {
119         console.log($('#stop').val());
120         socket.emit('stop', {value: $('#stop').val()});
121         return false; });
122
123
124
125     $('form#disconnect').submit(function(event) {
126         socket.emit('disconnect_request');
127         return false; });
128     });
129   </script>
```

- Vytvorenie tlačidiel na schránke

```
133   </head>
134   <body>
135    <h1> Ovladanie rezistora</h1>
136    <h2>Zadajte amplitudu pre rezistor od 0 po 255 </h2>
137
138
139    <form id="emit" method="POST" action='#'>
140      <input type="text" name="emit_value" id="emit_value" placeholder="Amplituda">
141      <input type="submit" value="Send">
142    </form>
143
144    <button id="start" type="submit" value=1>start</button>
145    <button id="stop" type="submit" value=0>stop</button>
146
147
148    <form id="disconnect" method="POST" action="#">
149      <input type="submit" value="Disconnect">
150    </form>
151
152     <div id="plotdiv" style="width:600px;height:250px;"></div>
153    <div></div><canvas style="width:300px;height:300px;" id="canvasID"></canvas></div>
154
155    <h2>Vystupna hodnota rezistora:</h2>
156    <div id="log"></div>
157   </body>
158  </html>
```

4.1 Užívateľské prostredie

- Web stránka

- Vypisovanie analógových údajov

```
Received #20: 255
Received #20: 255
Received #21: 6
Received #21: 6
Received #22: 1022
Received #22: 1022
Received #23: 1023
Received #23: 1023
Received #24: 1023
Received #24: 1023
Received #25: 1023
Received #25: 1023
Received #26: 1023
Received #26: 1023
Received #27: 1023
Received #27: 1023
Received #28: 1022
Received #28: 1022
Received #29: 1023
Received #29: 1023
Received #30: 1023
Received #30: 1023
Received #31: 1023
Received #31: 1023
Received #32: 1022
Received #32: 1022
```

- Výpis údajov v command window

```
message['value']
1
A= 1
Sent value
Go Button None
A= 1
Go Button None
A= 1
Go Button 1
A= 1
Go Button 1
A= 1
Go Button 1
A= 1
Go Button 1
A= 1
Go Button 1
message['value']
255
A= 1
Go Button 1
A= 255
```