SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta elektrotechniky a informatiky

Záverečné zadanie ku skúške

Pokročilé Informačné Technológie

2024

Bc. Juraj Kišš

Obsah

1 Zadanie 3

2 Diagramy a Zapojenie 4

3 Serverová časť 5

4 Klientská časť 8

5 Časť mikrokontrólera 9

6 Používateľská príručka 11

1 Zadanie

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.

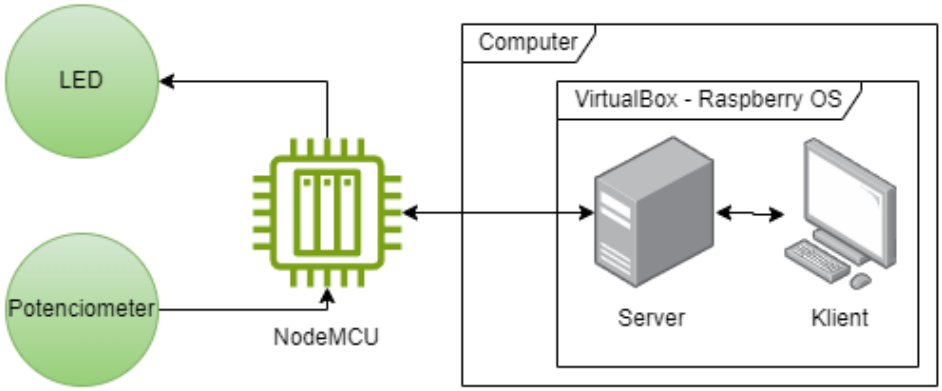
V zadaní sme použili 10kΩ potenciometer ktorý slúži ako náhrada senzoru, zelenú LEDku, predradný rezistok k LEDke o hodnote 230Ω a mikrokontróler NodeMCU 1.0.

V zadaní meriame analógový signál na potenciometri, ktorý posielame cez sériovú linku do Raspberry OS virtuálnej mašiny. Túto hodnotu potom vypisujeme a vykresľujeme na stránke.

Cez stránku vieme vypnúť a zapnúť meranie. To sa nám odzrkadľuje na LEDke, ktorá svieti ak meranie je zapnuté.

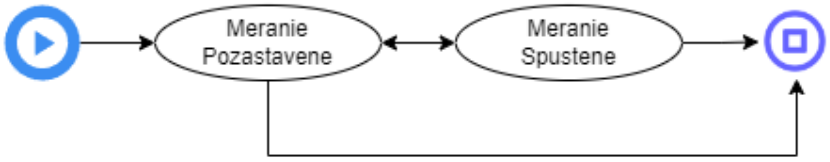
2 Diagramy a Zapojenie

Ako súčasť zadanie sme vypracovali UML diagram, ktorý môžeme vidieť na Obrázku 1. Tento diagram ukazuje tok dát a cez komunikačné kanály.



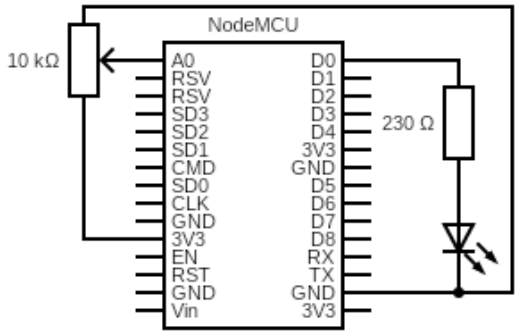
Obrázok 1 - UML diagram

Dalej sme vypracovali aj stavový diagram, ktorý môžeme vidieť na Obrázku 2.



Obrázok 2 - Stavový diagram

Ako posledné sme vypracovali schému zapojenia súčiastok, ktorú je na Obrázku 2.



Obrázok 3 - Zapojenie súčiastok

3 Serverová časť

Na serverovej časti zadania sme použili kód z cvičenia, ktorý sme upravili na naše potreby. Ako prvé sme pridali importovanie knižnice serial a inicializovali sme sériové pripojenie (Obrázok 4).

```
1   from threading import Lock
2   from flask import Flask, render_template, session, request, jsonify, url_for
3   from flask_socketio import SocketIO, emit, disconnect
4   import serial
5   import time
6   import random
7   import math
8
9   ser=serial.Serial("/dev/ttyUSB0",9600)
10  ser.baudrate=9600
```

Obrázok 4 - Importovanie knižníc a inicializácia sériového pripojenia

V nasledujúcej časti kódu (Obrázok 5) sme pridali premennú generate, ktorá slúži na uchovávnie informácie či máme alebo nemáme merať.

```
12  async_mode = None
13
14  app = Flask(__name__)
15
16  app.config['SECRET_KEY'] = 'secret!'
17  socketio = SocketIO(app, async_mode=async_mode)
18  thread = None
19  thread_lock = Lock()
20
21  generate = False
```

Obrázok 5 - Nastavenie Flasku, Vlákna a pomocných premenných

V nasledujúcej časti kódu (Obrázok 6) sme pridali posielanie dát na klientskú časť iba ak premenná generate je True. Taktiež sme pridali získavanie dát funkciou ReadValue() (Obrázok 7)

V nasleducich častiach kódu (Obrázok 8 a Obrázok 9) sme nič nemeli oproti cvičeniu

```
24  def background_thread(args):
25      count = 0
26      while True:
27          if generate:
28              socketio.sleep(2)
29              data = ReadValue()
30              count += 1
31              socketio.emit('my_response',
32                            {'data': data, 'count': count},
33                            namespace='/test')
34
```

Obrázok 6 - Vlákno bežiace na pozadí posielajúce dáta na klientskú časť

```
65  def ReadValue():
66          read_ser=ser.read_all().decode("utf-8")
67          lines = read_ser.split("\n")
68          line = lines[-2]
69          cleaned = line.strip()
70          parts = cleaned.split()
71          value = float(parts[1])
72
73          return value
```

Obrázok 7 - Funkcia na čítanie sériovej linky a získanie dát

```
35  @app.route('/')
36  def index():
37      return render_template('tabs.html', async_mode=socketio.async_mode)
38
39  @socketio.on('disconnect_request', namespace='/test')
40  def disconnect_request():
41      session['receive_count'] = session.get('receive_count', 0) + 1
42      emit('my_response',
43           {'data': 'Disconnected!', 'count': session['receive_count']})
44      disconnect()
```

Obrázok 8 - Inicializácia stránky a Ziadosť na odpojenie

V nasledujúcej časti kódu (Obrázok 10) sme pridali ziadosť pre klienta na spustenie/zastavenie merania a poslanie správy pre NodeMCU pomocou funkcie SendStop() (Obrázok 11), ktorá pošle do sériovej linky správu „Light“.

```
53  @socketio.on('connect', namespace='/test')
54  def test_connect():
55      global thread
56      with thread_lock:
57          if thread is None:
58              thread = socketio.start_background_task(target=background_thread, args=session._get_current_object())
59      emit('my_response', {'data': 'Connected', 'count': 0})
60
61  @socketio.on('disconnect', namespace='/test')
62  def test_disconnect():
63      print('Client disconnected', request.sid)
```

Obrázok 9 – Pripojenie na socket

```
46  @socketio.on('generate_request', namespace='/test')
47  def generate_request():
48      global generate
49      generate = not generate
50      print(generate)
51      SendStop()
```

Obrázok 10 - Žiadosť na zapnutie/vypnutie merania

```
75  def SendStop():
76          message = "Light"
77          byte_message = message.encode("utf-8")
78          ser.write(byte_message)
```

Obrázok 11 - Funkcia na poslanie správy do NodeMCU

4 Klientská časť

Vzhľadom na to, že sme v tejto časti zadania robili len minálne zmeny, ukážememe si len tú najdôležitejšiu. Ostatné zmeny ktoré sme vzkonali boli čisto výzorové.

Ako najdôležitejšiu zmenu sme vykonali pridanie tlačítka na zastavenie/spustenie merania pomocou formuláru s id = “generate“ (Obrázok 12 a Obrázok 13), ktorý volá žiadosť na serverovej časti s rovnakým názvom (Obrázok 10).

```
121         $('form#generate').submit(function(event) {
122             generate = !generate;
123 
124             if (generate)
125               document.getElementById("generateBTN").innerHTML = "Turn Off";
126             else
127               document.getElementById("generateBTN").innerHTML = "Turn On";
128 
129             socket.emit('generate_request');
130             return false; });
```

Obrázok 12 - Javascript časť formuláru generate

```
161       <form id="generate" method="POST" action='#'>
162         <button type="submit" id="generateBTN" value="generate">Turn On</button>
163       </form>
```

Obrázok 13 - HTML časť formuláru generate

5 Časť mikrokontrólera

V tejto kapitole si vysvetlíme kód nahraný na mikrokontróler NodeMCU 1.0

Ako prvé si nastavíme globálne premenné a funkcie jednotlyvých pinov (Obrázok 14).

```
1     bool generate = false;
2
3     void setup() {
4       Serial.begin(9600); // Starts the serial communication
5
6       pinMode(A0, INPUT);
7
8       pinMode(D0, OUTPUT);
9     }
```

Obrázok 14 - NodeMCU setup

V loope budeme čítať hodnotu na potenciometri a ak je premenná generate v hodnote True, tak túto hodnotu pošleme do sériovej linky (Obrázok 15).

```
11    void loop() {
12      int potenValue = analogRead(A0); // Read the potentiometer value
13
14      if (generate) {
15        Serial.print("Sending ");
16        Serial.println(potenValue);
17      }
18
```

Obrázok 15 - Čítanie hodnoty na potenciometri

Dalej sledujeme prichádzajúce správy do sériovej linky a ak sa nachádza správa „Light“, voláme funkciu GenerateSwitch() (Obrázok 16). Funkcia Generate switch() preklápa hodnotu generate a podľa nej zasvecuje alebo rozdsvecuje LEDku, na indikáciou či meriame alebo nie (Obrázok 17).

```
19      if (Serial.available() > 0) {
20        String inputMessage = Serial.readString();
21
22        if (inputMessage.indexOf("Light") >= 0) {
23          GenerateSwitch();
24        }
25      }
26    }
```

Obrázok 16 - Sledovanie prichádzajúcich správ do sériovej linky

```
28    void GenerateSwitch() {
29      generate = !generate;
30      if (generate){
31        digitalWrite(D0, HIGH);
32      } else{
33        digitalWrite(D0, LOW);
34      }
35    }
```

Obrázok 17 - Funkcia GenerateSwitch()

6 Používateľská príručka

1. Stiahneme si repozitár z https://github.com/JurajKiss/POIT_FINAL
2. Pripojíme NodeMCU ku počítaču.
3. Súbor POIT_Final/NodeMCU/zaverecne_zadanie.ino nahráme na NodeMCU
4. Otvoríme Raspberry OS virtuálnu mašinu
5. Nahráme do nej priečinok POIT_Final/Raspberry
6. Otvoríme v nej terminál
7. Pomocou príkazu cd sa dostaneme do nami nahraného priečinku
8. Príkazom ***sudo python3 'Zaverecne Zadanie.py'*** spustíme server
9. Otvoríme localhost stránku 0.0.0.0
10. Na stránke klikneme na tlačidlo „Turn On" na spustenie merania.
11. Kurzorom prejdeme na inú kartu podľa toho, či chceme vidieť vypisované hodnoty, graf alebo ciferník
12. Na stránke klikneme na tlačidlo „Turn Off" na vypnutie merania
13. Na stránke klikneme na tlačidlo „Disconnect" na odpojenie
14. V terminály stlačíme CRTL + C pre vypnutie servera.
15. Zatvoríme Raspberry OS virtuálnu mašinu
16. Odpojíme NodeMCU