



# Rampa s ultrazvukovým senzorom

Technická dokumentácia

Pokročilé informačné technológie  
I-AME-INTS

## Obsah technickej dokumentácie

1.	Definícia systému a používateľských požiadaviek .....	3
1.1	Používateľské požiadavky .....	3
1.1.1	Funkcionálne požiadavky .....	3
1.1.2	Nefunkcionálne požiadavky .....	3
2.	Systémová špecifikácia .....	4
2.1	Hardvér .....	4
2.2	Softvér .....	4
2.3	Use Case Diagram .....	5
2.4	Popis rolí .....	5
2.5	Diagram aktivít .....	6
2.6	Sekvenčný diagram .....	7
3.	Príručka .....	7
3.1	Arduino Uno .....	7
3.2	Raspberry Pi .....	7
3.3	Python-Flask webová aplikácia .....	9
4.	Verziovací systém .....	9

# 1. Definícia systému a používateľských požiadaviek

Zadaný systém má byť schopný nezávisle od interakcie používateľa monitorovať vzdialenosť použitím ultrazvukového senzoru HC-SR04, pričom v prípade, že zaznamená vzdialenosť menšiu než 20 cm, rampa tvorená servo motorčekom a ramenom sa otvorí, resp. otočí o 90°.

## 1.1 Používateľské požiadavky

Používateľské požiadavky určujú, čo je od systému očakávané zo strany používateľa. Sú to nie len funkcionálne požiadavky, ktoré udávajú aké funkcie má systém plniť, ale aj nefunkcionálne požiadavky, ako napríklad vzhľad.

### 1.1.1 Funkcionálne požiadavky

- Arduino Uno slúžiace na vykonávanie meraní a ovládanie akčných členov
- Meranie v reálnom čase
- Ukazovatele merania v reálnom čase
- Ukladanie zaznamenaných dát do databázy
- Ukladanie zaznamenaných dát do textového súboru
- Vizualizácia dát uložených v databáze a v textovom súbore

### 1.1.2 Nefunkcionálne požiadavky

- Spoľahlivosť systému
- Presnosť meraní
- Jednoduchá a intuitívna webová aplikácia

## 2. Systémová špecifikácia

Systém sa skladá z 2 zariadení- z Arduino Uno, ktoré monitoruje hodnoty a ovláda servo motorček a z virtuálneho Raspberry Pi, ktoré cez sériovú komunikáciu prijíma monitorované hodnoty a ďalej s nimi pracuje.

Na virtuálnom Raspberry Pi je spustený jednoduchý server. Po pripojení klienta cez webový prehliadač je nutné inicializovať spojenie, po čom začne výpis hodnôt na stránke. Po stlačení tlačítka „Start“ sa hodnoty začnú ukladať, po opätovnom stlačení tlačítka „Stop“ sa uložené hodnoty zapíšu do databázy a do textového súboru.

### 2.1 Hardvér

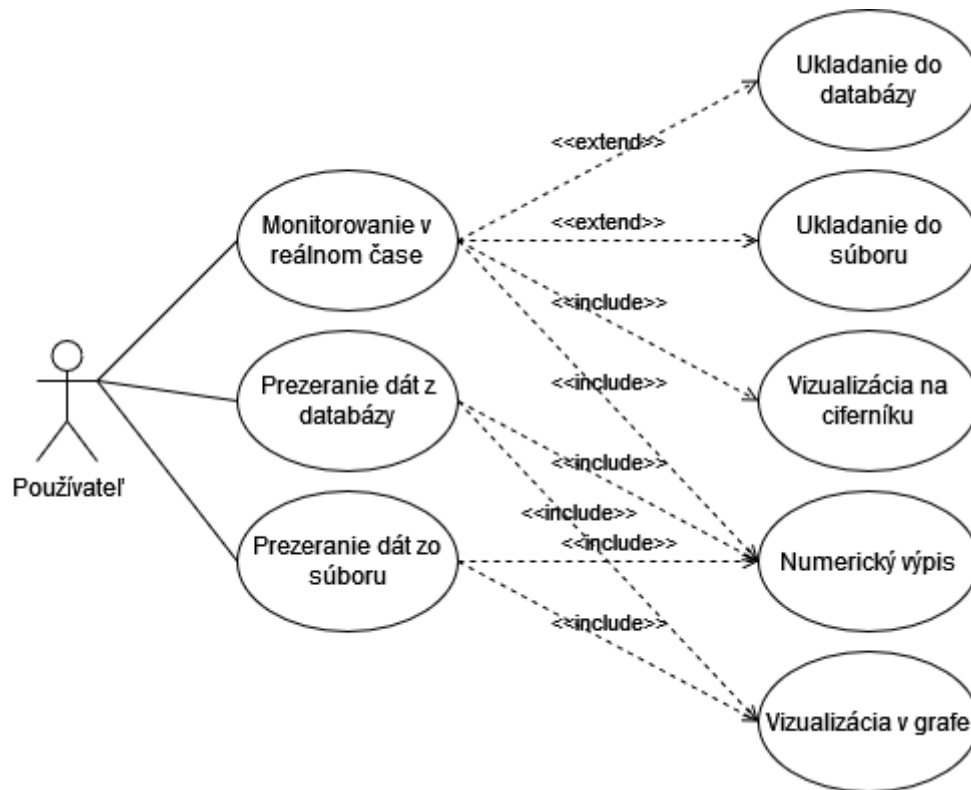
Na zostrojenie hardvérovej časti systému bol použitý klon Arduino Uno od Elegoo, neoznačený micro servo motorček a ultrazvukový senzor HC-SR04.

### 2.2 Softvér

Zdrojový kód pre Arduino Uno bol písaný v Arduino IDE, v jazyku C s použitím knižnice Servo.h. Táto knižnica umožňuje jednoduché ovládanie servo motorčeka jedným kontaktom (signal) bez nutnosti prepočtu pulzov. Meranie vzdialenosti je uskutočnené pomocou dvoch digitálnych pinov, pričom jeden slúži ako spúšť (Trigger) na zapnutie vysielania ultrazvuku a druhý (Echo) pomocou príkazu pulseIn meria dobu, za koľko sa vyslaný ultrazvukový signál vrátil späť do modulu HC-SR04. Pulzy sú následne vynásobené rýchlosťou zvuku a vydelené 2, nakoľko toto trvanie obsahuje „cestu tam aj späť“. Ak je získaná hodnota menšia ako 20 cm, servo motorček dostane signál na otočenie o 90 stupňov. Získaná hodnota v cm je taktiež následne zapísaná na sériový výstup Arduina.

Na serverovej strane je použitý programovací jazyk Python s framework-om Flask, ktorý uľahčuje prácu s klient-server architektúrou a web socketmi. Súčasťou serveru sú tzv. end-pointy, pomocou ktorých je možné pristupovať k funkcionalitám, podstránkam a záznamom dát. Okrem toho boli použité knižnice Plotly na zobrazovanie grafov, Gauge na zobrazenie ciferníka, jQuery na ľahkú prácu s elementmi HTML dokumentu a Bootstrap pre vizuálne úpravy stránky.

## 2.3 Use Case Diagram

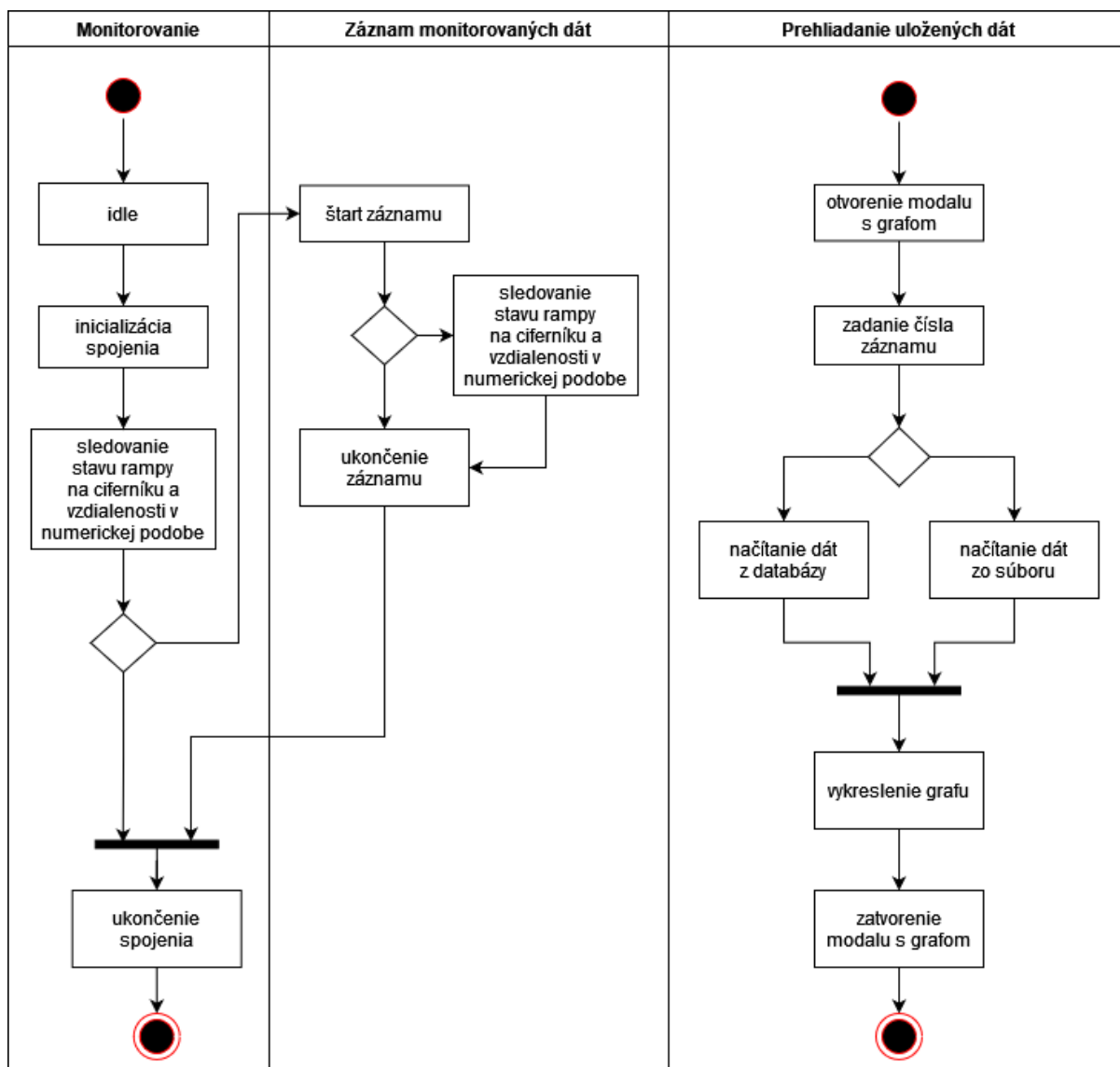


Obrázok č. 1: Use Case diagram

## 2.4 Popis rolí

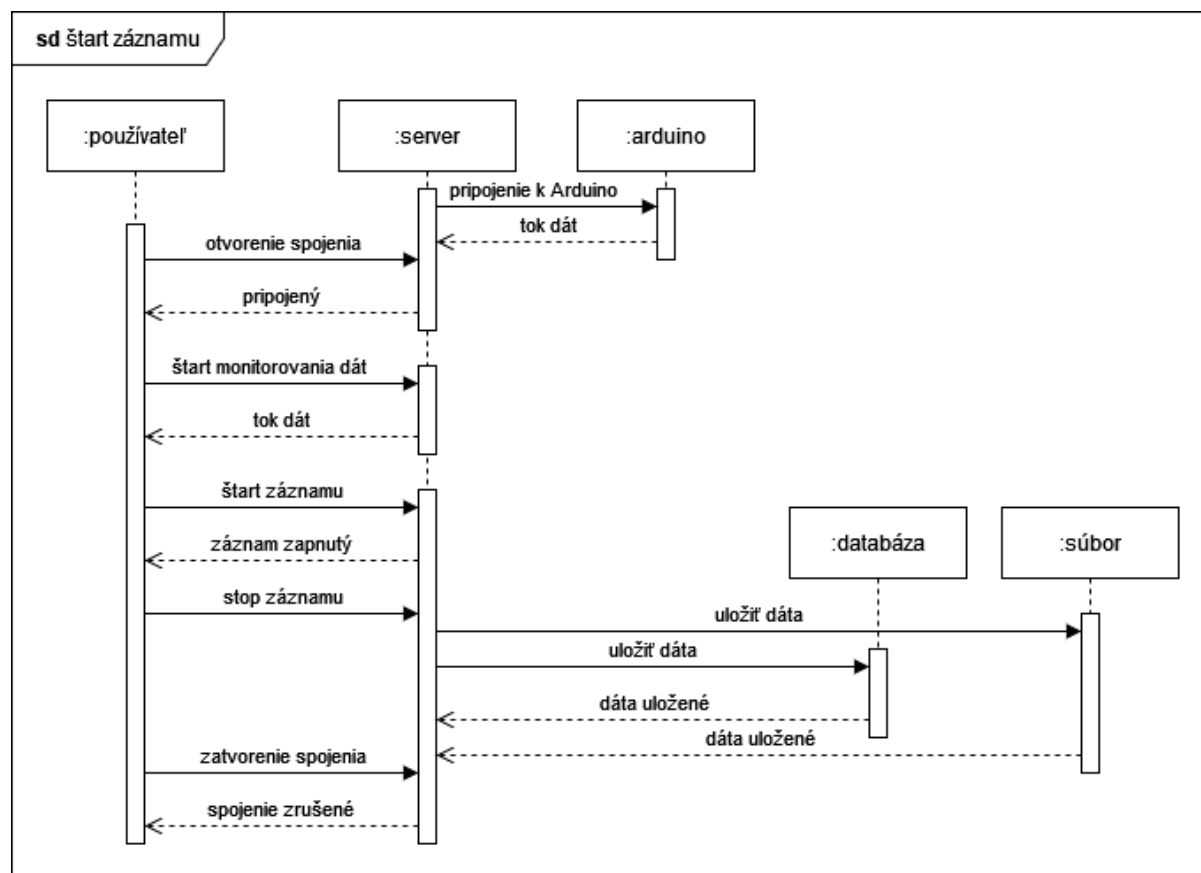
Jedinou rolou softvéru je používateľ- klient. Používateľ sa cez webový prehliadač pripojí na webovú aplikáciu, kde môže v reálnom čase sledovať monitorované hodnoty a stav akčného členu- rampy. Okrem real-time monitorovanie môže taktiež prehliadať uložené záznamy z databázy a z textového súboru a to vykreslené v grafe a vypísané v číselných hodnotách.

## 2.5 Diagram aktivít



Obrázok č. 2: Diagram aktivít

## 2.6 Sekvenčný diagram



Obrázok č. 3: Sekvenčný diagram štartu záznamu

## 3. Príručka

Táto kapitola obsahuje podkapitoly zamerané na hlavné časti systému. Obsah príručky by mal používateľovi bližšie priblížiť dostupné funkcionality a poskytnúť jednoduchý návod na používanie systému. Skúsenejší používateľ by mal byť schopný na základe príručky upraviť systém podľa vlastných požiadaviek.

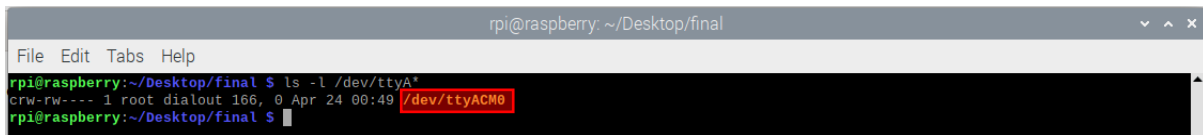
### 3.1 Arduino Uno

Arduino Uno slúži na čítanie hodnôt z ultrazvukového senzoru HC-SR04, ktoré následne prepočíta na vzdialenosť v centimetroch a na základe zistenej vzdialenosti ovláda servo motorček- rampu. Arduino Uno stačí pripojiť k fyzickému Raspberry Pi alebo k PC s bežiacim virtuálnym RPi OS.

### 3.2 Raspberry Pi

V prípade, že má používateľ k dispozícii fyzické Raspberry Pi, možno bude nutné v zdrojovom kóde serveru **app.py** zmeniť adresu portu pre komunikáciu s Arduino Uno. Toto je možné upraviť jednoducho priamo na Raspberry Pi, ako je znázornené na nasledujúcom postupe:


Ako prvé je potrebné zistiť adresu pripojeného portu zadáním príkazu `ls -l /dev/tty*` do príkazového riadku. Vo výpise hľadáme riadok v tvare `/dev/ttyUSBx` alebo `/dev/ttyACMx`, kde `x` je ľubovoľné číslo. V našom prípade pôjde o `/dev/ttyACM0` ako je zvýraznené v červenom obdĺžniku:



```
rpi@raspberrypi: ~/Desktop/final
File Edit Tabs Help
rpi@raspberrypi:~/Desktop/final $ ls -l /dev/ttyA*
crw-rw---- 1 root dialout 166, 0 Apr 24 00:49 /dev/ttyACM0
rpi@raspberrypi:~/Desktop/final $
```

Obrázok č. 4: Výpis pripojených portov

Zistenú adresu portu následne napíšeme do **app.py** na zelenou zvýraznené miesto namiesto pôvodnej adresy:



```
app.py x
1 import serial
2 from threading import Lock
3 from flask import Flask, render_template, session, request, jsonify, url_for
4 from flask_socketio import SocketIO, emit, disconnect
5 import MySQLdb
6 import time
7 import configparser as ConfigParser
8 import json
9
10 async_mode = None
11
12 app = Flask(__name__)
13
14
15 ser = serial.Serial('/dev/ttyACM0', 9600) #Spojenie s Arduino cez seriovu komunikáciu
16 config = ConfigParser.ConfigParser()
```

Obrázok č. 5: Definícia portu v zdrojovom kóde

Spustenie serveru je možné v príkazovom riadku príkazom **sudo python3 app.py** (je nutné nachádzať sa v adresári ktorý obsahuje súbor **app.py**). Ak sa nenachádzame v danom adresári, presunieme sa doň pomocou príkazu **cd**.

Po úspešnom spustení serveru uvidíme v príkazovom riadku výstup podobný tomu na nasledujúcej snímke:



```
rpi@raspberrypi: ~/Desktop/final
File Edit Tabs Help
rpi@raspberrypi:~/Desktop/final $ sudo python3 app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 301-450-911
```

Obrázok č. 6: Výstup po spustení serveru

Úlohou Raspberry Pi v našom systéme je čítanie dát od Arduino Uno a následné poskytovanie týchto dát používateľovi, v prípade štartu záznamu aj zápis dát do databázy a do textového súboru. Okrem komunikácie poskytuje aj tzv. endpointy, cez ktoré je možné dostať sa ku konkrétnym záznamom v databáze či textovom súbore.



### 3.3 Python-Flask webová aplikácia

Webová aplikácia bola vytvorená pomocou Flask framework-u pre Python. Tento framework zjednodušuje prácu s klient-server architektúrou, web socketmi a s API endpoint.

Po pripojení klienta na server a načítaní stránky je inicializovaný socket pre dané spojenie. Po stlačení tlačítka Open sa na pozadí spustí `background_thread`, ktorý zabezpečuje vypisovanie hodnôt v numerickej podobe, ovládanie ciferníka na signalizáciu stavu rampy-  
Otvorená/Zatvorená a zbieranie a zápis dát do databázy a súboru.

Po opätovnom stlačení tlačítka Open je komunikácia inicializovaná a dáta sa začnú vypisovať na stránke. Taktiež sa na ciferníku prejavujú zmeny stavu servo motorčeka v prípade, že k nim príde. Medzi reálnym stavom a stavom u klienta je pár sekundový rozdiel spôsobený prenosom a spracovaním na strane serveru a klienta.

Po stlačení tlačítka Start sa zmení text tlačítka na „Stop“ a začne sa záznam dát. Po stlačení tlačítka Stop sa text zmení späť na „Start“ a dáta sú zapísané do databázy a do súboru.

Stlačením tlačítka Close sa spojenie ukončí a výpis dát je prerušený.

Vykreslenie dát do grafu je možné stlačením tlačítka Graph, otvorí sa nám modal s riadkom pre zadanie identifikátora riadku. Po zadaní čísla a následnom kliknutí na jedno z tlačítok (**from DB** alebo **from file**) sú dáta načítané a vykreslené do grafu a vypísané po riadkoch pod grafom. Toto okno je možné zatvoriť buď kliknutím na **X** v pravom hornom rohu modal okna, alebo kliknutím kdekoľvek mimo tohto okna. Po opätovnom stlačení tlačítka Graph je graf vykreslený ako bol pred posledným zatvorením.

Okrem vyššie spomenutých tlačítok má používateľ k dispozícii nasledujúce API endpointy:

- **/db** pre výpis všetkých záznamov databázy
- **/file** pre výpis obsahu celého textového súboru
- **/dbdata/<číslo>** pre výpis daného záznamu databázy v JSON formáte
- **/read/<číslo>** pre výpis daného záznamu textového súboru v JSON formáte

## 4. Verzionovací systém

Pri vývoji bol použitý verzionovací systém GitHub pre jednoduché sledovanie zmien, možnosť návratu k predošlému stavu a dostupnosť súborov naprieč všetkých platforiem.

Link na GitHub repozitár so súbormi: [https://github.com/xzatkot/POIT\\_final](https://github.com/xzatkot/POIT_final)