

Synchronizace obrazových a akustických dat

programátorská příručka

ENC-NSS: Nasazení software a služeb

Prázdný řetězec

Otevřená informatika (N-OI)

Brno, 15. 5. 2025

Obsah

1	Úvod	1
1.1	Architektura	1
1.2	Použité technologie	1
2	Lokální vývoj	2
2.1	Databáze	2
2.2	Backend	3
2.3	Frontend	3
2.4	Nastavení environmentálních proměnných	4
2.5	docker-compose	4
2.6	LabVIEW	5

1 Úvod

Tento dokument slouží jako programátorská dokumentace k softwarovému dílu SOAD (Synchronizace obrazových a akustických dat). SOAD je laboratorní aplikace, která slouží ke sběru, agregaci a uchovávání naměřených údajů z různých senzorů.

Systém umí číst data z akustické emise (zařízení ZEDO). Dále pak umí komunikovat s RGB kamerou firmy Basler a touto komunikací je schopný pořizovat a ukládat snímky. Nakonec také umí komunikovat s multispektrální kamerou od firmy Basler a to pomocí technologie LabVIEW.

Výsledná data, která je systém schopný ze senzorů vyčíst či je pomocí nich pořídit, jsou komprimována a agregována pomocí archivace do jednoho souboru. Výsledný archivní soubor je nakonec nahráván na cloudové úložiště. Celý software je možné ovládat pomocí webového rozhraní (viz [Uživatelský manual](#))

1.1 Architektura

Projekt byl vytvořen pomocí struktury **monorepo**. V jediném GitHub repozitáři se nachází veškerý zdrojový kód projektu. Odkaz na repozitář: <https://github.com/Prazdny-retezec/soad>

Byla použita klasická architektura klient-server. Klient slouží k ovládání celého systému prostřednictvím grafického rozhraní a komunikuje se serverem. Server obsahuje většinu výpočetní logiky:

- Komunikuje s jednotlivými senzory. Získává z nich data, případně je jinak ovládá.
- Získaná data agreguje a komprimuje.
- Zároveň rovněž ukládá všechna data pro obsluhu měření do centrální databáze systému.
- Nakonec také komunikuje s cloudovým úložištěm, autentizuje se u něj a nahrává na něj výsledné archivy dat.

1.2 Použité technologie

Celá struktura systému je zpracována ve třech kontejnerech pomocí [Docker Compose](#):

- [Vue.js](#) frontend – SPA
- [FastAPI](#) backend – REST API s vlastní dokumentací formou OpenAPI a Swagger
- [Postgres](#) db – ukládání naplánovaných úloh

Další použité technologie:

- [LabVIEW](#) – přijímá a zpracovává data z multispektrální kamery (MS). Musí běžet na PC v laboratoři. Využívá se protokol GigE Vision a rozhraní **Gigabit Ethernet**
- [Pylon](#) – pro plné fungování RGB kamery (v tomto projektu byla použita verze 8.1.0). Musí být nainstalováno na laboratorním PC

- [Pypylon](#) – ovládání RGB kamery přes **USB 3**
- [APScheduler](#) – plánování úloh (měření)
- [SQLAlchemy](#) – SQL toolkit a ORM pro Python
- [Pytest](#) – testování BE
- [Google Drive](#) – nahrávání zazipovaných dat (akustická emise a fotky)
- [AlwaysData](#) – nasazení
- [ZEDO](#) a webové sockety – akustická emise (AE)

2 Lokální vývoj

Celý systém je možné spustit dvěma způsoby. Buďto prostřednictvím technologie Docker (respektive docker-compose) a nebo nativně. Systém je možné spustit v ostrém režimu nebo v tzv. mock režimu. To v jakém režimu se systém spustí lze nastavit pomocí enviromentálních proměnných.

V mock režimu bude systém produkovat pouze nepravdivé údaje. Tyto údaje ovšem vychází z opravdových dat, jsou tedy naprosto realistické. Tento režim se obzvlášť hodí v případě dalšího vývoje, kdy nejsou opravdové senzory k dispozici.

2.1 Databáze

Nejprve je nutné sestavit a spustit Postgres databázi. Lze to udělat dvěma způsoby: [lokálně nainstalovat](#) nebo použít Docker. V obou případech je nutné nastavit proměnné PGDATA, POSTGRES_DB, POSTGRES_USER a POSTGRES_PASSWORD (jejich hodnoty jsou v souboru docker-compose.yml).

V případě Dockeru je nutné použít následující příkaz:

```
docker run --name soad-db -p 5002:5432 \
-e PGDATA=/var/lib/postgresql/data/db-files/ \
-e POSTGRES_DB=soad \
-e POSTGRES_USER=api \
-e POSTGRES_PASSWORD=changeit \
-v database_volume:/var/lib/postgresql/data/ postgres:17-alpine

# V případě, že neexistuje volume
docker volume create database_volume
```

Je možné také jít cestou docker-compose:

```
# Pozor: udělá build všech služeb z .yaml (BE, FE, db)
docker-compose build
docker-compose up database
```

Po spuštění výše uvedených příkazů by se měla vytvořit databáze s názvem soad, kam se budou ukládat veškerá měření. Postgres by měl běžet na portu 5002 a být připraven k akceptaci dotazů.

2.2 Backend

Po spuštění databáze můžeme připravit backend. Je nutné mít nainstalovaný interpret jazyka **Python** (verze ≥ 3.11). Lze použít i docker-compose, ale po každé změně ve zdrojovém kódu bude potřeba vytvořit nový image backendu.

Zpočátku vytvoříme virtuální prostředí a nainstalujeme závislosti. Na macOS/Linux:

```
# soad/backend/  
python3 -m venv venv  
  
source venv/bin/activate  
  
python3 -m pip install --upgrade pip  
  
pip3 install -r requirements.txt
```

Na Windows:

```
# soad/backend/  
# Občas py nefunguje (záleží na instalaci), zkusit python3  
py -m venv venv  
  
venv\Scripts\activate  
  
py -m pip install --upgrade pip  
  
pip install -r requirements.txt
```

Dále je nutné přidat soubor `.env`. Záleží na přání vývojáře, jak ho vyplní, ale ukázkové nastavení environmentálních proměnných je možné nalézt v `soad/backend/.env.example`. Poté můžeme spustit soubor `main.py`:

```
fastapi dev src/main.py
```

Po krátké době se vytvoří FastAPI aplikace a poběží na adrese <http://localhost:5001>. Na <http://localhost:5001/docs> poběží Swagger dokumentace s popisem endpointů. Je důležité zmínit, že celá backendová služba je zabezpečena pomocí **Basic Auth**.

2.3 Frontend

Nutnou podmínkou pro FE je nainstalovaný **Node.js**. Dále je nutné přidat nezbytné závislosti:

```
# soad/frontend/  
npm install
```

Podobně jako u backendu je potřeba přidat soubor `.env` a vyplnit jej. Ukázkové nastavení se nachází v `soad/frontend/.env.example`. Následně je potřeba spustit Vite dev server:

```
npm run dev
```

Aplikace by měla běžet na adrese <http://localhost:5000>.

2.4 Nastavení environmentálních proměnných

Jak bylo uvedeno v předchozích částech, je nutné nastavit proměnné prostředí pro FE i BE. Níže je popis všech proměnných:

- FE:
 - VITE_API_URL - odkaz na BE
 - VITE_ADMIN_USERNAME - přihlašovací jméno na FE
 - VITE_ADMIN_PASSWORD - přihlašovací heslo na FE
- BE:
 - DATABASE_URL - adresa Postgres databáze
 - OUTPUT_DIR - absolutní cesta na adresář, kde se budou ukládat mezi-výsledky měření
 - AE_IP_ADDRESS - IP adresa akustické emise
 - AE_PORT - port akustické emise
 - MOCK_DATA_DIR - absolutní cesta na adresář, který obsahuje modelová data
 - GDRIVE_CREDENTIALS_DIR - absolutní cesta na adresář obsahující přihlašovací údaje do Google Drive
 - MOCK_ACOUSTIC_EMISSION - pokud je TRUE, nastaví se mock režim pro AE (více o tom v Úvodu)
 - MOCK_RGB_CAMERA - pokud je TRUE, nastaví se mock režim pro RGB kameru (více o tom v Úvodu)
 - MOCK_MULTI_SPECTRAL_CAMERA - pokud je TRUE, nastaví se mock režim pro MS kameru (více o tom v Úvodu)

2.5 docker-compose

Pro otestování aplikace v kontejnerech je nutné spustit níže uvedené příkazy:

```
docker-compose build  
docker-compose up  
  
# Vypne kontejnery  
docker-compose down
```

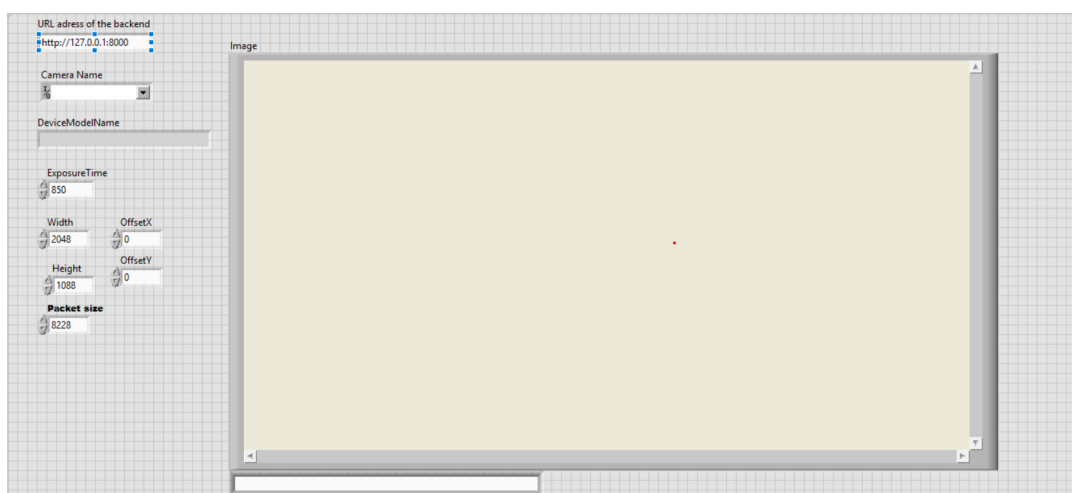
2.6 LabVIEW

LabVIEW slouží pro práci s multispektrální kamerou. Postačí jakákoli stabilní verze **LabVIEW**, která poběží na laboratorním PC. Dané PC musí disponovat alespoň jedním portem USB 3 (připojení RGB kamery) a dvěma Ethernet rozhraními, přičemž alespoň jedno z nich bude Gigabit Ethernet (připojení MS kamery) a v operačním systému musí být na rozhraní síťové karty nastavena **maximální velikost jumbo packetů**.

Po připojení MS kamery je potřeba otevřít labview/ v LabVIEW prostředí. V tomto adresáři se nachází:

- LabVIEW 2012/ (nutný toolkit)
- lv_gige/ (LabVIEW projekt)
- labview-control/ (FastAPI aplikace)

Před spuštěním je nutné nastavit parametry kamery. Níže je příklad:



Obrázek 1: Parametry MS v LabVIEW

- **URL address of the backend** - adresa BE, kam LabVIEW bude posílat GET dotaz
- **Path** – cesta k obrázku, který pořídí kamera
- **CameraName** – musí být vybrána připojená MS
- **Width & Height** – musí odpovídat hodnotám, které jsou v DeviceModelName (zde například D2048x1088)
- **Packet size** – musí být nastaven na hodnotu **8228**

Teprve teď můžeme spustit LabVIEW kód a FastAPI aplikaci v labview-control/. LabVIEW bude každou sekundu vysílat HTTP GET dotaz a pokud dostane hodnotu 1, tak se pořídí snímek, jinak čeká. Dotaz je vysílán na FastAPI aplikaci.