Mendelova univerzita v Brně
Provozně ekonomická fakulta

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 Základní popis projektu

Systém SOAD slouží k synchronizaci a správě obrazových a akustických dat.

1.1 Funkcionalita

1. **Získávání dat ze senzorů** – čtení a sbírání dat z různých senzorů, konkrétně z akustických senzorů a kamer (RGB, multispektrální).
2. **Ukládání dat** – data jsou uchována lokálně a následně synchronizována do centrálního úložiště (cloud).
3. **Automatická transformace dat** – data jsou automaticky transformovaná do vhodných formátů (PNG, TIFF, …).
4. **Automatické spuštění a nastavení měření** – podpora automatického spuštění měření na základe definovaných kritérií (čas, událost).
5. **Webová administrace** – uživatelské rozhraní, které poskytuje možnost nastavení a konfiguraci systému a plánování měření.

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

2 Lokální vývoj

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 .yml (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 enviromentálních 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.

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 docker-compose

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

```
docker-compose build
docker-compose up

# Vypne kontejnery
docker-compose down
```

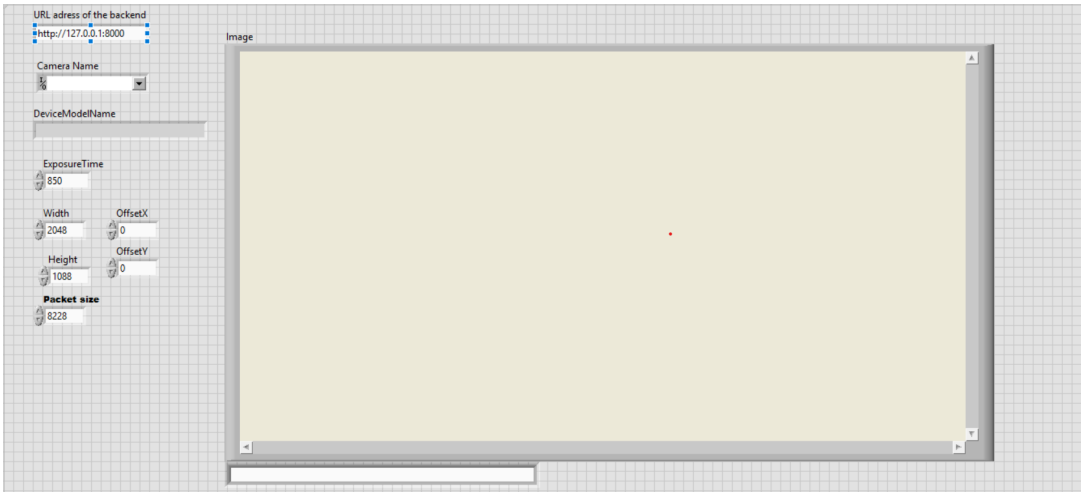
2.5 LabVIEW

Postačí jakákoliv stabilní verze LabVIEW, která poběží na laboratorním PC. MS musí být připojena přes **Gigabit Ethernet** a v operačním systému musí být na rozhraní síťové karty nastavena maximální velikost jumbo packetů.

Po připojení 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.