**Traffic Analysis System**

**28/05/2021**

**Software Requirements and Installation**

1. **OS: Ubuntu 20.04 LTS**
2. **Python version 3.8**

apt install python3.8

apt install python3-pip

1. **FFmpeg version 4 or newer**

apt install ffmpeg

1. **PostgreSQL version 12 or newer**

apt install postgresql

create custom user, password, database

1. **Web server (Apache or Nginx)**
2. **Python libs from standard repo (pip install -r requirements.txt)**

ffmpeg-python

psutil

psycopg2-binary

pandas

paho-mqtt

norfair

pandarallel

1. **Special Python libs**

* PyTorch for system CUDA version (<https://pytorch.org/>)
* OpenCV 4

apt update

apt install libopencv-dev python3-opencv

1. **Traffic Analysis System**

8.1. Install software (<https://github.com/DRR-IGI/workflow.git>)

8.2. Setup local configuration file (bin/config.ini)

8.3. Make data dirs

python makedirs.py

8.3. Setup database

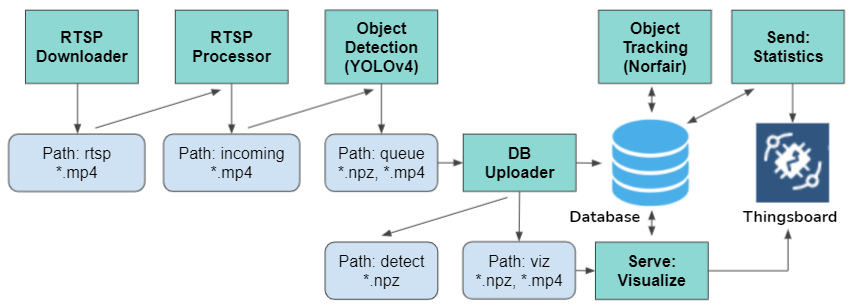
psql -d (db\_name) < ../sql/schema.sql

8.4. Start/ Stop the system

bin/start.sh

bin/stop.sh

**Main Processing Workflow**



1. **RTSP Downloader**

Download segment of RTSP video into path: rtsp

1. **RTSP Processor**

Check and move complete video segments into path: incoming

1. **Object Detection (YOLOv4)**

Detect objects in video, write detection results to a npz file. Move complete data into path: queue

1. **DB Uploader**

* Upload detection results, move the npz file to path: detect
* Generate frame-no and timestamp mapping to support visualization, move the npz file with video into path: viz

1. **Object Tracking (Norfair)**

In the database, wait for new detection data and create tracking data

1. **Serve: Visualize**

Periodically, generate visualization and send a trigger to Thingsboard

1. **Send: Statistics**

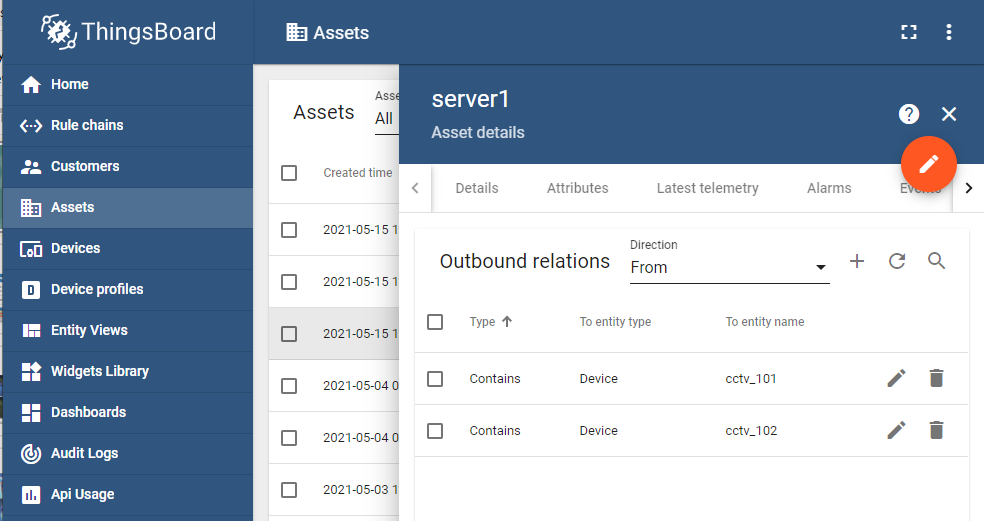
Periodically, send statistic to Thingsboard

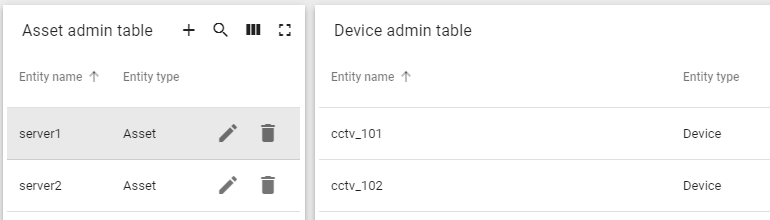
**System Configuration**

1. **Thingsboard Relationship and Attributes**

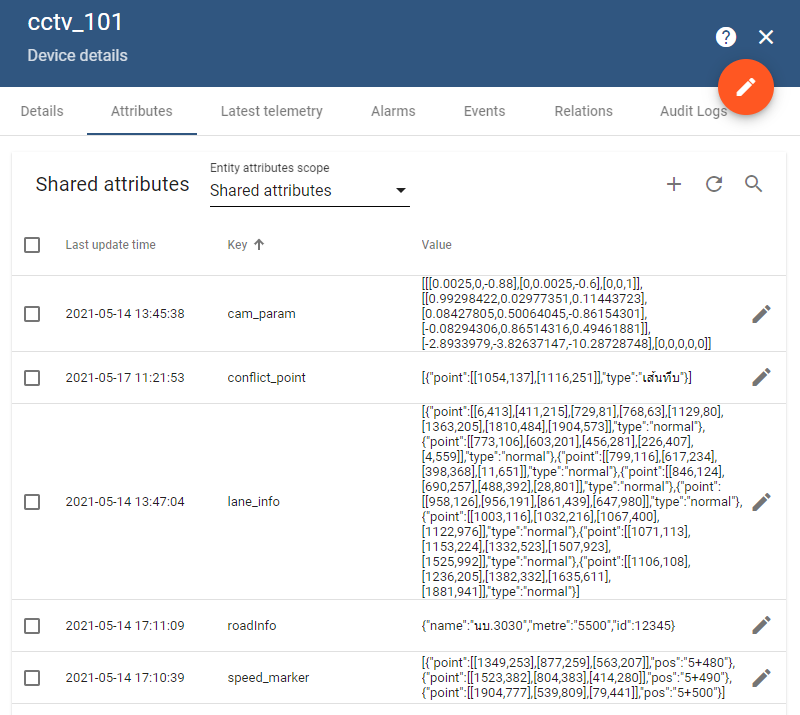
**Asset: server**

contains devices: **cctv**

****

****

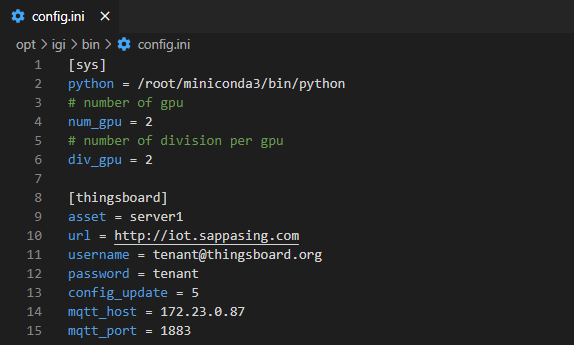
**Device: cctv**

****

**Shared Attributes:**

|  |  |
| --- | --- |
| Attributes | Description |
| cam\_param | Perspective adjustment matrix |
| lane\_info | Lanes Informaton |
| rtsp\_src | RTSP source (rtsp://host:port/stream) |
| speed\_marker | Zone of LOS checking |
| conflict\_point | Define conflict line that no car should cross |
| ai\_disable | Disable AI processing (if true) |

1. **Local configuration (config.ini)**

****

1. **Database: CCTV Table**

|  |  |
| --- | --- |
| Column | Description |
| id | ID of CCTV (Integer) |
| device\_id | Thingsboard: Device ID |
| name | Thingsboard: Name of CCTV |
| access\_token | Thingsboard: Access Token |
| attributes | Thingsboard: CCTV shared attributes |
| last\_update | Attributes last updated |
| active | Ready for processing (with valid attributes)  Required Attributes: 'cam\_param', 'lane\_info', 'rtsp\_src'  \*\* If “ai\_disable” = True, CCTV Active = False \*\* |

**Components**

1. **Config Updater (config\_update.py)**

Periodically update CCTV config from Thingsboard to DB

1. **CCTV Processor (cctv\_processor.py)**

Continuously process CCTV data

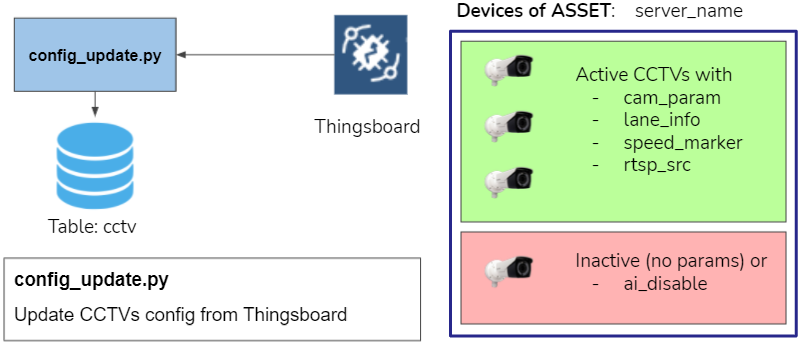
1. **Path Monitor (monitor.py)**

Watch and process files in the path

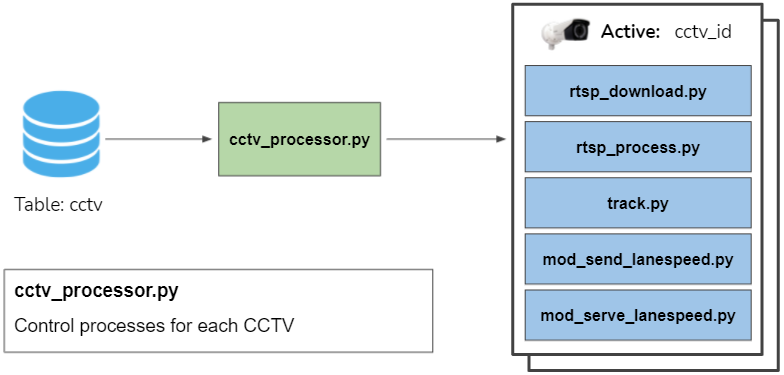
**Type of Processing**

|  |  |  |
| --- | --- | --- |
| System processing | Per CCTV processing | Concurrent processing |
| Config Updater | CCTV Processor   * rtsp\_download * rtsp\_process * track * mod\_send\_lanespeed * mod\_serve\_lanespeed | Path Monitor |

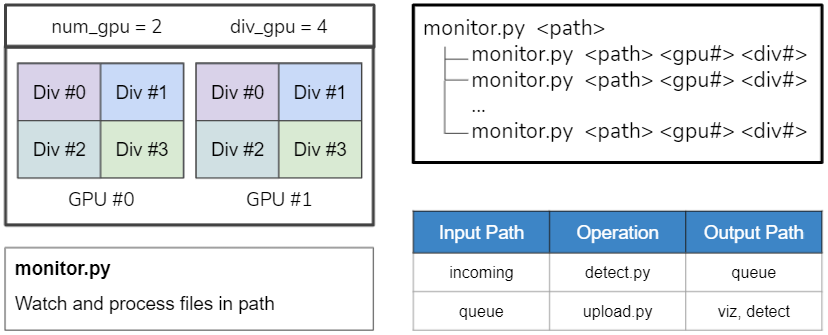
**Config Updater**

****

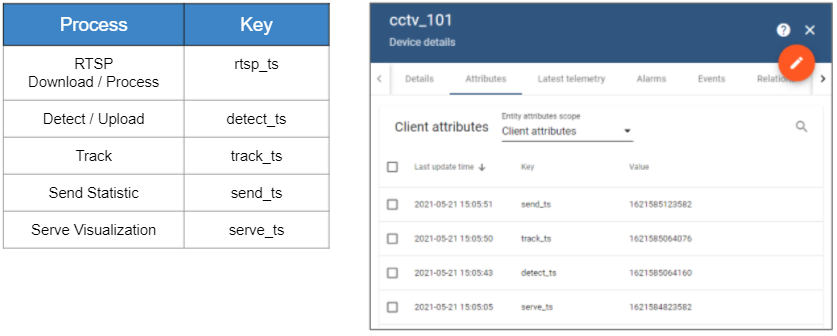
**CCTV Processor**



**Path Monitor**



**CCTV Processing Status**



**Programs and command-line arguments**

**[config\_update.py]**

Update CCTV config from Thingsboard

python config\_update.py [--sync sec]

**[detect\_opencv.py]**

Object detection - YOLOv4 OpenCV DNN

python detect\_opencv.py <source\_video> [--weights model\_weight\_file] [--config model\_config\_file] [--output out.npz]

**[detect.py]**

Object Detection - PyTorch YOLOv4

python detect.py <source\_video> [--weights model\_weight\_file] [--config model\_config\_file] [--output out.npz] [--device cuda\_device]

**[makedirs.py]**

Make all required data dirs

**[mod\_avg\_lanespeed.py]**

Calculate average lanespeed and number of cars

python mod\_avg\_lanespeed.py <cctv\_id> <start\_ts> <stop\_ts>

**[mod\_conflict\_report.py]**

Generate conflict report

python mod\_conflict\_report.py <cctv\_id> <start\_ts> <stop\_ts> <out.csv>

**[mod\_LOS.py]**

Generate LOS report

python mod\_LOS.py <cctv\_id> <start\_ts> <stop\_ts> <time\_step> <out.csv>

**[mod\_send\_lanespeed.py]**

Send statistic data to Thingsboard

python mod\_send\_lanespeed.py <cctv\_id> [last\_ts]

**[mod\_serve\_lanespeed.py]**

Serve visualize video and send trigger to Thingsboard

python mod\_serve\_lanespeed.py <cctv\_id> [last\_ts]

**[mod\_viz\_detect.py]**

Visualize object detection

python mod\_viz\_detect.py <cctv\_id> <start\_ts> <stop\_ts> <out.mp4>

**[mod\_viz\_lanespeed.py]**

Visualize Lane speed

python mod\_viz\_lanespeed.py <cctv\_id> <start\_ts> <stop\_ts> <out.mp4>

**[mod\_viz\_track.py]**

Visualize Object Tracking

python mod\_viz\_track.py <cctv\_id> <start\_ts> <stop\_ts> <out.mp4>

**[mod\_viz\_trajectory\_conflict.py]**

Visualize Lane speed with trajectory view

python mod\_viz\_trajectory\_conflict.py <cctv\_id> <start\_ts> <stop\_ts> <out.mp4>

**[monitor.py]**

Monitor and process files in the path

python monitor.py <queue|incoming> [<gpu\_id> <div\_id>]

**[rtsp\_download.py]**

Download video from RTSP source

python rtsp\_download.py <cctv\_id>

**[rtsp\_process.py]**

Check and move RTSP video to path: incoming

python rtsp\_process.py <cctv\_id>

**[start.sh]**

Start the workflow

**[stop.sh]**

Stop the workflow

**[track.py]**

Object tracking with Norfair

python track.py <cctv\_id> [last\_ts]

**[upload.py]**

Upload object detection result to DB

python upload.py <input\_video>