# گزارش پروژه‌ی پایانی دوره‌ی کارشناسی

**محمدحسین اسدی**
**۹۷۵۳۶۲۲۰۲**
**تابستان ۱۴۰۲**

## مقدمه

برای پروژه‌ی پایانی دوره‌ی کارشناسی قصد داشتم روی چیزی کار کنم که متناسب با مهارت‌هایم باشد. و مهارت‌های من بیشتر در زمینه‌های مربوط به مدیریت سیستم (System Administration) بود. از مدت‌ها پیش دامنه، آی‌پی ثابت و سرور خانگی (یک مینی‌کیس دست دوم!) برای خودم تهیه کرده بودم و هنوز هم از آن برای کارهای روزمره‌ی خودم استفاده می‌کنم. مثلاً سرویس ابری (Cloud Service) شخصی خودم را با Self-Host کردن Nextcloud راه انداخته‌ام یا برای پروژه‌های شخصی (مثل همین پروژه) از ورژن‌کنترل شخصی خودم که با استفاده از Gitea راه‌اندازی شده استفاده می‌کنم. از آنجایی که سرور خانگی امکانی بود که از قبل درگیر شدن با پروژه‌ی کارشناسی در اختیار داشتم و البته تجربه‌ی استفاده از داکر (docker) برای راه‌اندازی سرویس‌های مختلف روی آن را هم داشتم، تصمیم گرفتم پروژه‌ی کارشناسی را هم به نحوی به آن ربط بدهم.

از طرفی از آنجایی که شرکتی که در آن کار می‌کنم یک شرکت مبتنی بر اینترنت اشیاء بوده و با کامپیوترهای تک‌بردی (Single Board Compute – SBC) زیاد سروکار داشتم دوست داشتم پروژه‌ام کمی هم چاشنی سخت‌افزاری داشته باشد و از دستگاه‌هایی مثل رزبری‌پای در پروژه‌ام استفاده کنم.

طبق صحبتی که اوایل ترم با استاد محترم صورت گرفت ابتدا تصمیم بر این بود که من لیستی از لوازم مورد نیاز برای پروژه (مثل رزبری‌پای، ماژول دوربین و ...) تهیه کنم و بعد پروژه را شروع کنیم و من در محل دانشگاه روی پروژه کار کنم. از آنجایی که مدت کوتاهی پس از صحبت با استاد گرامی درگیر کنکور کارشناسی ارشد بودم و بعد از آن هم کارهای شرکت بیشتر شد و بیشتر وقتم در شرکت می‌گذشت و به دانشگاه فقط در حد حضور در کلاس‌ها می‌رسیدم تصمیم گرفتم مسیر پروژه را کمی تغییر دهم تا انعطاف لازم برای هماهنگی با برنامه‌ی جدیدم را داشته باشد. بنابراین از آنجایی که خودم هم علاقه داشتم، یک دستگاه BananaPi M2U خریداری کردم تا در وقت آزاد خودم روی پروژه کار کنم. انتخاب BananaPi برای کاهش هزینه‌ها بود. زیرا ارزان‌ترین دستگاه رزبری‌پای بیش از ۳ برابر این بناناپای قیمت داشت و البته این بناناپای امکانات بهتری به نسبت قیمت ارائه می‌کرد. و دوباره برای کاهش هزینه‌ها به جای خرید ماژول دوربین از یک وبکم قدیمی استفاده کردم. هرچند به دلیل کمتر شناخته‌شده بودن بناناپای مدت زمان زیادی درگیر boot و راه‌اندازی موفقیت آمیز آن بودم، ولی خود این مسیر هم ارزش آموزشی داشت؛ و مگر فلسفه‌ی انجام این پروژه چیزی غیر از آن است؟

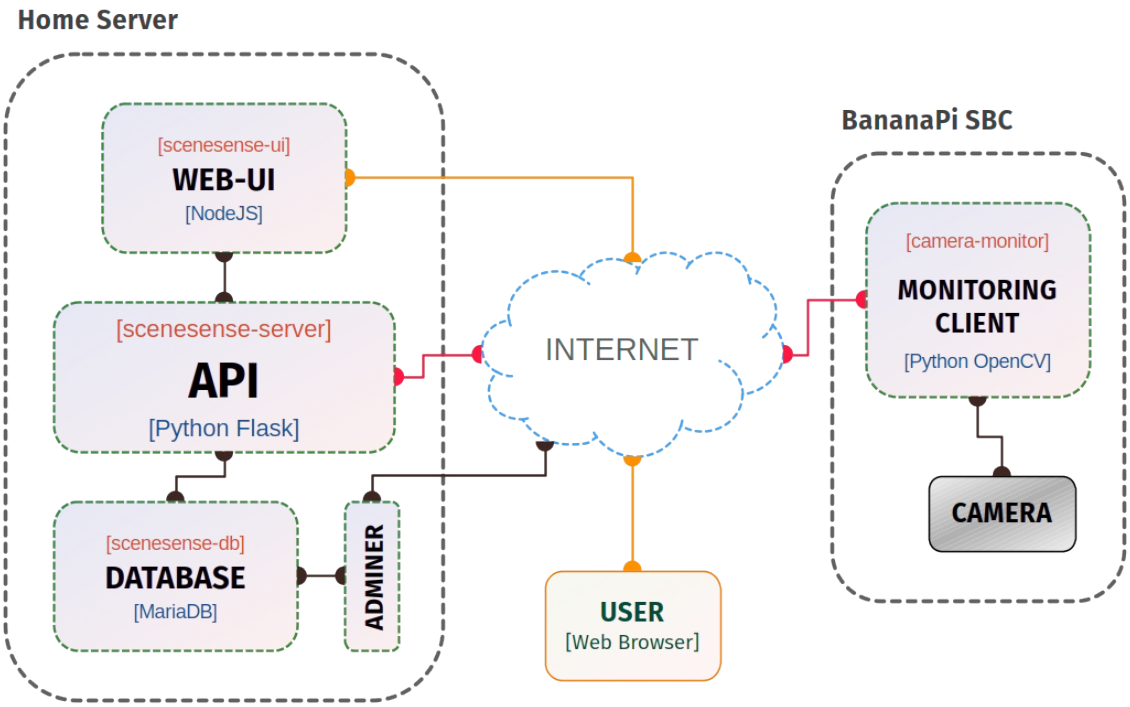
در این پروژه، که نام آن را SceneSense گذاشته‌ام هدف کلی این است که با استفاده از یک کامپیوتر کوچک تک‌بُردی و یک دوربین بتوان محیط را پایش کرد و تغییرات به وقوع پیوسته در محیط را با برچسب زمانی و اگر لازم بود توضیحات اضافی، در یک سرور و دیتابیس خارجی ذخیره کرد.

## شمای کلی پروژه

این پروژه از دو بخش کلی تشکیل می‌شود: کلاینت و سرور. در سمت سرور چهار سرویس مختلف داکر اجرا می‌شوند، یک برنامه‌ی اصلی سرور که نقش API را دارد و با زبان پایتون نوشته شده است و

وظیفه‌ی اضافه کردن رکورد به دیتابیس و خواندن رکوردها از دیتابیس را دارد. دیتابیسی که رکوردها در آن ذخیره می‌شوند به عنوان یک کانتینر داکری جداگانه وجود دارد. بخش سوم وب‌اپلیکیشنی است که برای مشاهده‌ی رکوردهای دیتابیس از طریق وب وجود دارد و با استفاده از NodeJS نوشته شده است. مورد چهارم یک ابزار مدیریت دیتابیس مشابه phpMyAdmin است، البته به مراتب سبک‌تر و کم امکانات‌تر به نام Adminer که جهت دسترسی مستقل به دیتابیس در شرایط خاص ایجاد شده است.

در بخش دوم، کلاینت، دوربین یا وبکم از طریق پورت USB به بناناپای متصل می‌شود و برنامه‌ای پایتونی روی آن اجرا می‌شود.




### بخش API

در این بخش با استفاده از زبان پایتون و کتابخانه‌ی flask یک وب‌سرور ساده طراحی شده و مسیرهای مختلف (api endpoint) برای ایجاد رکورد جدید، دریافت آخرین رکورد، دریافت همه‌ی رکوردها و یا دریافت رکوردها در بازه‌ی زمانی خاص از دیتابیس وجود دارد. برای روابط بین برنامه‌ی flask و دیتابیس هم از کتابخانه‌ی flask_sqlalchemy استفاده شده است که یک لایه بین دیتابیس و برنامه اضافه می‌کند و کار کدنویسی را به مقدار زیادی راحت‌تر می‌کند.

برای استفاده از این API باید API Key مناسب در سرآمد (Header) درخواست به سمت سرور ارسال شود. دو سطح دسترسی تعریف شده: RO و RW؛ اولی فقط دسترسی خواندن دارد و فقط می‌تواند به مسیر(route)های مربوط به خواندن اطلاعات از دیتابیس درخواست دهد و دومی دسترسی خواندن و

نوشتن دارد و می‌تواند از تمام مسیرهای موجود برای خواندن از دیتابیس و نوشتن به دیتابیس استفاده کند.

برنامه‌ی اجرایی این بخش در فایل‌های پروژه در مسیر server/src/server_api.py قابل مشاهده است، برای استقرار (deploy) این برنامه از Dockerfile ای که در مسیر server/Dockerfile وجود دارد استفاده شده است. ایمیج برنامه بر روی ایمیج python:3.10-slim-bookworm ساخته شده و هنگام ایجاد کانتینر داکر -در کنار سرویس‌های دیگر- از آن استفاده می‌شود. (ر. ک. فایل docker-compose.yml ای که در پوشه‌ی container قرار دارد.)

**بخش WEB-UI**

این بخش شامل یک برنامه‌ی NodeJS می‌شود که یک رابط تحت وب برای دریافت اطلاعات و نمایش آن به کاربر ارائه می‌کند. API Key کاربر را از ورودی می‌گیرد و با استفاده از آن، آخرین رکورد، تمام رکوردها و یا درصورت دریافت بازه‌ی زمانی از کاربر، رکوردهای مربوط به آن بازه‌ی زمانی را به کاربر نمایش می‌دهد.

برنامه‌ی اجرایی این بخش در مسیر webui-client/src/main.js قابل مشاهده است، برای استقرار این برنامه نیز از Dockerfile ای که در مسیر webui-client/Dockerfile وجود دارد استفاده شده است. ایمیج برنامه بر روی ایمیج node:20-slim ساخته شده و مشابه بخش قبل هنگام ایجاد کانتینر داکر -در کنار سرویس‌های دیگر- از آن استفاده شده است. (ر. ک. فایل docker-compose.yml ای که در پوشه‌ی container قرار دارد.)

**بخش DATABASE**

برای این پروژه، به دلیل آشنایی قبلی، از دیتابیس MariaDB که fork ای از دیتابیس MySQL است استفاده کرده‌ام. دیتابیس را به طور مستقیم با استفاده از داکر و ایمیج رسمی MariaDB راه‌اندازی کرده‌ام. دیتابیس شامل جدولی به نام records است و این جدول شامل سه ستون id, timestamp, description است. id به عنوان private key انتخاب شده است و در هر رکورد افزایش پیدا می‌کند و همیشه منحصر به فرد است. مقدار timestamp همیشه به صورت Unix-Timestamp و بدون در نظر گرفتن منطقه‌ی زمانی در دیتابیس ذخیره می‌شود. در بخش توضیحات یا description فعلاً صرفاً یک جمله‌ی ساده مثل «تغییری در محیط در زمان [زمان] به وقوع پیوست» البته به انگلیسی نوشته می‌شود. در این بخش به جز یک اسکریپت ساده‌ی sql ای که برای ایجاد جدول در دیتابیس استفاده می‌شود و هنگام راه‌اندازی کانتینر دیتابیس برای اولین بار نیاز است و در مسیر زیر قرار دارد، من کار خاصی نکرده‌ام.

container/docker-entrypoint-initdb.d/scenesense_db_init.sql

**بخش MONITORING**

تقریباً مهم‌ترین بخش این پروژه که کار اصلی را انجام می‌دهد بخش مانیتورینگ است. در برنامه‌ی پایتونی که برای این کار نوشته شده و در مسیر camera-client/camera_monitor.py در دسترس است از کتابخانه‌های OpenCV و NumPy و چند کتابخانه‌ی built-in دیگر برای این منظور استفاده شده است. ابتدا یک کلاس به اسم CameraMonitor تعریف کرده‌ام که متدهای مختلفی برای وظایف مورد نظر در آن

تعریف شده. مثلاً، متد initialize_camera برای راه‌اندازی دوربین استفاده می‌شود. متد detect_scene_change برای تشخیص میزان تغییر دو فریم متوالی و مقایسه‌ی آن با میزان threshold ای که کاربر مشخص کرده و در نهایت تشخیص اینکه تغییر مد نظر اتفاق افتاده یا خیر استفاده می‌شود. متد log_scene_change برای ایجاد رکورد جدید جهت ثبت وقوع تغییر در دیتابیس از طریق API Call استفاده می‌شود. و در نهایت متد run که حلقه‌ی اصلی برنامه است و به طور دائمی اجرا می‌شود و فریم‌ها را می‌خواند و برای هر دو فریم متوالی متد detect_scene_change را فراخوانی می‌کند.

در برنامه سعی شده تا حد امکان ویژگی logging وجود داشته باشد و همچنین در مقابل اتفاقات مورد انتظاری مثل قطع و از دسترس خارج شدن دوربین، fail safe باشد. در صورت وقوع برنامه به طور مداوم retry می‌کند و در صورت اتصال مجدد دوربین کارش را از سر می‌گیرد و نیاز به راه‌اندازی مجدد ندارد.

در این برنامه از فایل کانفیگ خارجی برای اعمال تنظیمات استفاده شده که یک نسخه به عنوان نمونه در فایل camera-client/example.config.json وجود دارد. این برنامه برای درست کار کردن حتماً به API Key با سطح دسترسی RW نیاز خواهد داشت.

در فایل تنظیمات می‌توان device مربوط به دوربین، میزان threshold یا حساسیت، آدرس سرور API، تأخیر detection_delay که برای محدود کردن میزان ثبت اطلاعات در سرور استفاده می‌شود (مثلاً اگر به مقدار 10 تنظیم شده باشد باید بین دو بار ثبت رکورد در سرور حداقل 10 ثانیه فاصله باشد)، API Key مورد استفاده و همچنین مقدار failure_delay که مدت زمان صبر کردن قبل از تلاش مجدد است را تعیین کرد.

## بخش عمومی

در این بخش فقط به صورت تیتروار به یک سری فعالیت‌های جانبی مربوط به پروژه اشاره می‌کنم:

- راه‌اندازی سرور خانگی با سیستم‌عامل لینوکس، تهیه‌ی نام دامنه و آی‌پی ثابت (شرح در مقدمه)
- راه‌اندازی نرم‌افزار داکر روی سرور
- راه‌اندازی وب‌سرور Nginx
- ایجاد subdomain های لازم و تنظیم آی‌پی مناسب برای سرویس‌ها مختلف پروژه
- ایجاد systemd unit برای camera_monitor که یک نسخه به عنوان نمونه در فایل‌های پروژه در آدرس camera-client/example.camera_monitor.service وجود دارد.
- تنظیمات مربوط به وب‌سرور Nginx که روی سرور خانگی‌ام انجام شده است.
- راه‌اندازی سرویس‌های پروژه با استفاده از docker-compose

## استفاده از برنامه و لینک‌های مفید

برای استفاده از برنامه فقط به یک API Key با دسترسی مناسب نیاز خواهید داشت. API Key را می‌توان در سمت سرور به صورت متغیر محیطی تعریف کرد و در اختیار کاربر قرار داد. در حال حاضر دو کلید برای این پروژه تعریف شده که یکی دسترسی RW دارد و دستگاه بناناپای با استفاده از آن رکوردها را روی

دیتابیس سرور می‌نویسد و دیگری با دسترسی RO برای تست کنندگان در نظر گرفته شده است که در همین گزارش قابل مشاهده است:

Read-Only API KEY: **8Pr8IbWn28vxVJ8ofDF4zrGe**

با استفاده از این کلید و از طریق رابط کاربری تحت وب یا با request زدن به طور مستقیم به API Server می‌توان رکوردهای دیتابیس را مشاهده کرد. علاوه بر این‌ها نرم‌افزار مدیریتی Adminer نیز در دسترس است هرچند نام کاربری و رمز عبور برای استفاده از آن جهت تست ارائه نشده است.

لینک‌ها:

| | |
|---|---|
| API Server | api.ss.mhasadi78.ir |
| Web UI Client | ui.ss.mhasadi78.ir |
| DB Adminer | dbadmin.ss.mhasadi78.ir |
| Git Repository | git.mhasadi78.ir/mhasadi78/SceneSense |

مسیرهای API:

| Route | Method | Description |
|---|---|---|
| /add_record | POST | Add a new record to the database |
| /get_last_record | GET | Get the last recorded entry from the database |
| /get_all_records | GET | Get all records from the database |
| /get_records_since/<timestamp> | GET | Get records since a given timestamp |
| /get_records_between/<start_timestamp>/<end_timestamp> | GET | Get records between two timestamps |
| /get_records_until/<timestamp> | GET | Get records until a given timestamp |

## جمع‌بندی

در این پروژه سعی شد با استفاده از امکانات محدود و بدون تحمیل هزینه‌های غیرلازم، پروژه‌ای پیاده‌سازی شود که پتانسیل استفاده در موارد پرکاربردتر را هم داشته باشد. قطعاً در این پروژه چیز جدیدی به حوزه‌ی علوم کامپیوتر اضافه نشده و همه‌ی این موارد در اشکال مختلف قبلاً به کرات انجام شده بودند و این پروژه صرفاً یک سرهم‌بندی از تکنولوژی‌های مختلف و با اهداف آموزشی بود تا بتوانم مهارت‌هایی که آموخته‌ام را در پیاده‌سازی و راه‌اندازی یک پروژه‌ی نه‌چندان ساده و نه‌چندان پیچیده محک بزنم. نکته‌ی قابل توجه در این پروژه این است که در نرم‌افزار مانیتورینگ که از دوربین استفاده می‌کند، با کمی تغییر در کد می‌توان منطق تشخیص وقوع تغییر را عوض کرد و مثلاً از مدل‌های train شده برای تشخیص اشیاء یا چهره‌های خاص استفاده کرد. بنابراین با تغییر داخل یک متد می‌توان از بقیه‌ی این پلتفرم برای کاربردهای جدی‌تری استفاده کرد.

با احترام
محمدحسین اسدی
**دانشکده‌ی مهندسی برق و کامپیوتر**
دانشگاه تبریز