به نام خدا

گزارش‌کار اسپرینت دوم پروژه شماره ۱۹ – دستیار تشخیص حداکثر سرعت مجاز

تیم شماره ۱

- یاشار ظروفچی
- سپهر صفری

گزارش‌کار اسپرینت ۳

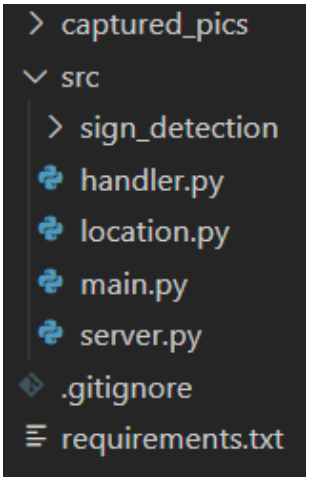
متاسفانه باید اعلام کنیم که یکی از هم‌تیمی‌های ما، اقای مقراضی، تصمیم گرفتند از تیم جدا بشوند. به همین خاطر تیم ما از این پس ۲ نفره خواهد بود.

در پایان دو هفته سوم، موارد زیر به انجام رسید:

۱. پیاده‌سازی کد رزبری برای مدیریت ماژول‌های آن
۲. طراحی اولیه شیوه تعامل و تبادل اطلاعات اپ اندروید و رزبری (raspberry server)
۳. نحوه بدست آوردن سرعت و مکان از اطلاعات خام gps

مورد اول:

کد پیاده‌سازی شده در گیت‌هاب آپلود شده است. پوشه‌ها و فایل‌های موجود به طورخلاصه به شکل زیر هستند:



پوشه captured_pics: جایی است که عکس‌های گرفته شده توسط دوربین، در آن ذخیره می‌شود. پوشه فایل‌های کد را در خود دارد.

پوشه sign_detection: همان کد پردازش تصویر برای پردازش تصاویر تابلوهای راهنمایی است که در اسپرینت‌های قبلی نوشته شده بود و صرفا کمی interfaceهای آن را تغییر دادیم تا کد فعلی بتواند از آن استفاده کند.

فایل main.py: فایل اصلی پروژه است که در ابتدا اجرا می‌شود و تردهای لازم برای برنامه را می‌سازد. در سمت رزبری کلا ۳ تا ترد اجرا می‌شوند. اولین ترد که در بدو شروع برنامه توسط خود سیستم‌عامل تولید می‌شود، وظیفه راه‌اندازی اولیه سیستم را برعهده دارد. مثلا کتابخانه‌های پردازش تصویر که بارگذاری آن‌ها زمان زیادی می‌برد را بارگذاری می‌کند. همچنین دو ترد دیگر برنامه را تولید می‌کند. ترد دوم، modules-thread است. این ترد به صورت دوره‌ای در دوره‌هایی به طول ۰٫۱ ثانیه با ماژول‌های دوربین، gps و buzzer تعامل می‌کند و کارهای مربوط به آن‌ها را انجام می‌دهد. ترد تابع handle_sensors در فایل handler.py را اجرا می‌کند. ترد سوم و آخر، view-thread است که سروری را بر روی رزبری بالا می‌آورد تا اپ اندروید درخواست‌های خود را به این سرور بزند و اطلاعات لازم را از آن بگیرد. این ترد تابع run_server در فایل server.py را اجرا می‌کند.

فایل handler.py: ترد modules-thread تابع handle_sensors این فایل را اجرا می‌کند که تصویری از آن در زیر آمده است:

```python
def handle_sensors():
    last_camera_clk = 0
    while True:
        start = get_time()

        handle_gps(start)
        if start - last_camera_clk > CAMERA_PERIOD:
            last_camera_clk = start
            handle_camera(start)
        handle_buzzer(start)

        now = get_time()
        sleep(CLK_PERIOD - (now - start))
```

همان طور که مشاهده می‌کنید این تابع یک حلقه while True است که به صورت دوره‌ای اجرا می‌شود و با تمام ماژول‌ها در آن تعامل صورت می‌گیرد. دو تابع handle_camera و handle_gps هم در زیر نمایش داده شده‌اند که به ترتیب برای تعامل با دوربین و gps استفاده می‌شوند:

```python
def handle_camera(cur_time):
    global speed_limits, signs, lock

    lock.acquire()
    pic_name = 'None'
    prediction = sign_detection.predict_pic(pic_name)
    signs.append((cur_time, prediction[0]))
    new_speed_lim = sign_detection.get_speed_limit(speed_limits, signs)
    if new_speed_lim != None:
        speed_limits.append((cur_time, new_speed_lim))
    lock.release()
```

```
def handle_gps(cur_time):
    global locations, speeds, prev_speed, lock

    lock.acquire()
    loc = location.get_location()
    locations.append((cur_time, loc))
    cur_speed = location.get_speed(locations, prev_speed)
    speeds.append((cur_time, cur_speed))
    prev_speed = cur_speed
    lock.release()
```

در بالای این فایل تعدادی متغیر سراسری تعریف شده‌اند که از نوع آرایه هستند. این آرایه‌ها اطلاعات مختلف را با زمان ثبت شدن آن‌ها در خود ذخیره می‌کنند. مثلا در آرایه speeds، هر خانه آن یک چندتایی (tuple) به طول ۲ است که دانه اول آن زمان ثبت سرعت خودرو و دانه دوم آن سرعت خودرو در آن لحظه است. همچنین برای جلوگیری از رقابت بین view-thread و modules-thread یک قفل هم تعریف کرده‌ایم. کد مربوط به این متغیرها و قفل در زیر قابل مشاهده است:

```
lock = threading.Lock()
locations = []
speeds = []
prev_speed = 0
signs = []
speed_limits = []
```

فایل location.py: این فایل توابع مربوط به gps را در خود دارد. دو تابع مهم آن get_location و get_speed است که به ترتیب برای گرفتن مکان و سرعت خودرو در لحظه فعلی استفاده می‌شود. تابع get_speed برای محاسبه سرعت فعلی از مکان خودرو در آخرین دو لحظه اخیر و سرعت قبلی آن استفاده می‌کند.

فایل server.py: این فایل سرور را بر روی رزبری بالا می‌آورد و توسط ترد view-thread اجرا می‌شود. پیاده‌سازی این فایل هنوز تمام نشده است و اما طراحی آن صورت گرفته است که در بخش‌های بعدی به آن می‌پردازیم.

مورد دوم:

نحوه ارتباط اپ اندروید و رزبری یکی از چالش‌های موجود بود به این خاطر که برای ما مهم است که علاوه بر دریافت اطلاعات از رزبری، بتوان زمان هر داده را هم دانست تا بتوان در اپ نمودارهای مربوط به آن را رسم کرد و پردازش‌های لازم را انجام داد. همچنین اپ باید توانایی این را داشته باشد که فقط اطلاعات جدید را از رزبری دریافت کند و مثلا لازم نباشد هر دفعه سرعت‌های خودرو در از لحظه شروع به کار رزبری تا حالا را بگیرد. برای رفع این مشکل راه‌حلی که در نظر گرفتیم به این شکل است که در کنار هر داده‌ای که در رزبری ذخیره می‌شود، زمان ثبت آن هم ذخیره شود و اپ وقتی دادگانی را تا لحظه فعلی دریافت کرده این زمان را در خود ذخیره کند. در دفعات بعدی وقتی اپ خواست از رزبری اطلاعات جدید بگیرد، زمان آخرین داده دریافت شده را به رزبری ارسال می‌کند و رزبری فقط دادگانی را به اپ می‌فرستد که زمان آن‌ها بزرگ‌تر از زمان ارسال شده هستند. لازم به ذکر است

همان طور که بخش قبلی گفتیم اطلاعات به صورت تعدادی آرایه ذخیره می‌شوند که هر خانه از آرایه زمان ثبت آن داده و خود داده را در خود جای داده است.

مورد سوم:

در اسپرینت پیشین، نحوه‌ی دریافت موقعیت با کمک cgps را دیدیم. در این گزارش با کمک کتابخانه‌ی عمومی پایتون و import کردن ماژول serial می‌توانیم از /dev/serial0 که محل نوشتن داده‌های gps است بخوانیم.
خروجی‌های ماژول gps یک استریمی از اطلاعات متفاوت (و بعضا تکراری) ارائه می دهد. با بررسی خروجی‌ها متوجه خواهیم شد که داده‌ای با سرآیند GPRMC مشخص شده حاوی اطلاعات مدنظر ما است. بنابراین در تابع get_location موجود در فایل location.py می‌توانیم این خط را بخوانیم. سپس با استفاده از تابع formatDegreesMinuets این سری داده را تبدیل به موقعیت مکانی (طول و عرض جغرافیایی) می‌کنیم.
اکنون در تابع calc_speed با استفاده از زمان سپری شده بین دو استعلام از gps و محاسبه‌ی مسافت بین این دو نقطه سرعت را بر حسب متر بر ثانیه به دست می‌آوریم. در نهایت این خروجی این تابع که به get_speed می‌رود بر حسب کیلومتر بر ساعت گزارش می‌شود.