به نام خدا

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

تیم شماره ۱

- یاشار ظروفچی
- سپهر صفری

گزارش‌کار اسپرینت ۴

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

۱. پیاده‌سازی کد سرور رزبری
۲. طراحی جدید برنامه پردازش تصویر به منظور اضافه کردن قابلیت تشخیص تابلوها در عکس گرفته شده از خیابان
۳. پیاده‌سازی بخشی از اپ اندروید
۴. تست اندازه‌گیری سرعت
۵. رفع برخی باگ‌ها

مورد اول:

ما در اسپرینت قبلی سرور سمت رزبری را طراحی کردیم به شکلی که اپ اندروید بتواند با آن ارتباط برقرار کند. در این اسپرینت این طراحی را پیاده‌سازی کردیم که شامل نوشتن کد و تحلیل برخی جزئیات آن می‌شد.

طراحی کلی به این شکل است که سرور ۴ تا api از نوع GET در مسیرهای زیر دارد:

- /get-locations
- /get-speeds
- /get-signs
- /get-speed-limits

هر کدام از این ۴ api یک پارامتر به نام timestamp هم دارند. این پارامتر زمان ثبت آخرین داده‌ای اسست که اپ اندروید تا کنون از سرور دریافت کرده است. وقتی سرور یک ریکوئست از انواع بالا دریافت می‌کند، به کمک جستجوی دودویی داده‌ای که زمان ثبت آن با timestamp برابر است را در لیست متناظر پیدا می‌کند (مثلا لیست متناظر با مسیر get-locations/ برابر است با locations) و سپس تمام داده‌های بعد از داده مذکور را به اپ اندروید ارسال می‌کند. به این شکل اپ تمام داده‌های جدید را دریافت می‌کند. برخی توابع مهم کلاس MyHandler در زیر نشان داده شده‌اند:

تابع do_GET: تمام درخواست‌های GET که به سرور می‌آیند، ابتدا در اینجا بررسی می‌شوند:

```
def do_GET(self):
    self.__parse_get_params()
    if self.path.startswith('/get-locations'):
        self.__send_response(locations)
    elif self.path.startswith('/get-speeds'):
        self.__send_response(speeds)
    elif self.path.startswith('/get-signs'):
        self.__send_response(signs)
    elif self.path.startswith('/speed-limits'):
        self.__send_response(speed_limits)
    else:
        self.__send_400_response()
```

تابع send_response__: با توجه به لیست تعیین شده برای آن، داده‌های جدید را به کمک پارامتر timestamp تشخیص داده و برای اپ می‌فرستد.

```
def __send_response(self, arr):
    timestamp = int(self.params['timestamp'][0])
    data_lock.acquire()
    index = self.__find_timestamp_index(timestamp, arr)
    self.send_response(200)
    self.send_header('Content-type', 'text/html')
    self.end_headers()
    self.wfile.write(json.dumps(arr[index+1:len(arr)]).encode())
    data_lock.release()
```

تابع find_timestamp_index__: روی لیست داده شده به آن جستجوی دودویی انجام داده و اندیس اولین داده‌ای که زمان ثبت آن بزرگ‌تر یا مساوی با timestamp است را برمی‌گرداند.

```
def __find_timestamp_index(timestamp: int, arr: [tuple]):
    def bin_search(start, end):
        if start == end:
            return start
        mid = (start + end) // 2
        mid_timestamp = arr[mid][0]
        if mid_timestamp < timestamp:
            return bin_search(mid+1, end)
        else:
            return bin_search(start, mid)

    return bin_search(0, len(arr)-1)
```

مورد دوم:

تا قبل از این هوش مصنوعی پروژه صرفا توانایی تشخیص تابلوها در تصاویری را داشت که تابلو کل کادر تصویر را می‌پوشاند. مثلا اگر عکسی از خیابان می‌گرفتیم که در قسمتی از آن یک تابلو بود، هوش مصنوعی توانایی تشخیص آن را نداشت. در این اسپرینت برای حل این مشکل راه‌حلی طراحی کردیم و فقط باید آن را پیاده‌سازی کنیم. راه‌حل به این شکل هست که مثلا اگر تصویر ما ابعاد ۳۰۰ در ۳۰۰ دارد، یک مربع ۳۰ در ۳۰ در نظر می‌گیریم و آن را روی کل عکس حرکت می‌دهیم. به این شکل عکس اولیه به تعدادی عکس ۳۰ در ۳۰ تبدیل می‌شود (دقت شود این عکس‌ها می‌توانند همپوشانی هم داشته باشند). در نهایت هرکدام از این عکس‌ها را به هوش مصنوعی می‌دهیم تا معلوم شود تابلویی در آن هست یا نه. به محض اینکه تابلویی در یکی از این تصاویر پیدا کردیم (پیدا کردن تصویر تابلو یعنی که با احتمال بالایی، مثلا ۹۵ درصد، عکس مذکور با یک نوع تابلو تطبیق داده شود) از تابع تشخیص عکس خارج می‌شویم و تابلوی تشخیص داده شده را به عنوان خروجی برمی‌گردانیم.

مورد سوم:

همانگونه که پیشتر گفته شده بود، واسط کاربری بین راننده و رزبری از طریق اپلیکیشن اندروید است. برای این منظور، نیازمند موارد زیر در طراحی اپلیکیشن هستیم

- نمایش سرعت به صورت اعلان ثابت
- ارتباط از طریق REST Api با رزبری
- نمایش اعلان اضطراری (هنگام عبور از سرعت غیر مجاز)
- تعبیه‌ی یک fragment اصلی برای نمایش سوابق

برای این اسپرینت، دو مورد اول پیاده‌سازی شده‌اند. به صورت کلی با کمک توابع retrofit.http موجود در اندروید می‌توان امکان ارتباط بر بستر REST را مهیا کرد. همچنین با کمک NotificationManagerCompat نیز مدیریت اعلان‌ها در اندروید صورت می‌گیرد.

مورد چهارم:

در این هفته به صورت عملی، ماژول GPS حین حرکت خودرو بررسی شد و سرعت‌ها اندازه‌گیری شد که نتایج آن در گیت‌هاب ثبت شده‌اند.

مورد پنجم:

در اسپرینت قبلی کد بخش ماژول‌های رزبری تعدادی باگ داشت که مربوط به طول آرایه‌هایی بودند که داده‌های ثبت شده را در خود نگه می‌دارند. مشکل این بود که این آرایه‌ها در ابتدای راه‌اندازی برنامه طول صفر دارند اما در برخی از توابع از خانه‌های آخر و حتی یکی مانده به آخر این آرایه‌ها استفاده می‌شد. این توابع طول آرایه‌ها را چک نمی‌کردند به همین خاطر بروز خطای "اندیس خارج از محدود" برای این آرایه‌ها قطعی بود. در این اسپرینت این باگ‌ها رفع شدند.