گزارش پروژه درس نهفته

پروژه‌ی شماره ۹: کنترل مکان خودرو

اعضای تیم

امیرحسین عباسی - ۹۷۱۰۲۰۴۴

پرهام چاوشیان - ۹۸۱۰۰۱۱۸

محمد طه جهانی نژاد - ۹۸۱۰۱۳۶۳

بهار ۱۴۰۱-۱۴۰۲

تعریف پروژه

در این پروژه قصد داریم تا یک سیستم نهفته طراحی کنیم که بر روی خودروها نصب می‌شوند و به مالکان خودرو اجازه‌ی کنترل مکان خودرو و در صورت نیاز، خاموش و یا روشن کردن آن‌را می‌دهد.
به صورت دقیق‌تر نیازمندی‌های پروژه به شرح زیر هستند:

- موقعیت مکانی خودرو همواره قابل رصد و بررسی باشد.
- در صورت دور شدن خودرو از محدوده‌ی اولیه، زنگ مربوطه فعال شود و اطلاع رسانی دهد.
- در صورت دور شدن خودرو از محدوده‌ی ثانویه، تصویری از خارج خودرو برای مسئول ارسال شود و همچنین موتور خودرو به صورت خودکار خاموش شود.
- در صورت نیاز، مسئول بتواند موتور خودرو را از راه دور کنترل کند.

نیازمندی‌های فنی پروژه

طبق کاربردهایی که بالاتر ذکر شد، برای پیاده‌سازی این پروژه به تجهیزات فنی زیر نیاز خواهیم داشت:

- یک بورد Raspberry pi 3، که به عنوان پردازشگر اصلی عمل کند. همچنین مدل pi 3 بلوتوث و وای‌فای را به صورت Built-in خواهد داشت.
- یک سنسور GPS، که برای بررسی لحظه‌ای موقعیت خودرو به کار می‌رود.
- یک ماژول Buzzer، که در صورت نیاز فعال شده و زنگی را پخش می‌کند.
- یک ماژول Relay، که برای خاموش و روشن کردن موتور خودرو استفاده خواهد شد. همچنین در حالت تست، از یک Fan به عنوان موتور خودرو استفاده خواهیم کرد.
- یک ماژول دوربین، که برای تصویربرداری استفاده خواهد شد.

طراحی پروژه - کلاینت

کدهای مربوط به کلاینت، در ریپوی گیت، داخل پوشه‌ی Client قرار دارند. هر فایل پایتون داخل این پروژه مربوط به فعالیت یکی از ماژول‌های توضیح داده شده در بخش قبلی می‌باشد. در ادامه توضیحات مربوط به هر فایل به صورت کلی آورده شده است.

ماژول gps

فایل gps.py

فایل gps.py برای فعالسازی ماژول gps و دریافت موقعیت مکانی به کار می‌رود. تابع get_location عملیات اصلی این ماژول را به عهده دارد.
ماژول gps، در هر لحظه، تعداد زیادی پیام‌های مختلف که به نام NMEA messages شناخته می‌شوند را روی پورت serial رزبری ارسال می‌کند. همه‌ی این پیام‌ها مورد استفاده‌ی ما نیستند و تنها پیام‌هایی را نیاز داریم که حاوی موقعیت مکانی می باشند. در نتیجه روی نوع پیام - که ۶ کاراکتر اول هر پیام می‌باشد - یک شرط گذاشته‌ایم و به محض دریافت مسیج‌های از نوع $GNRMC، آن‌را parse می‌کنیم و به صورت یک آبجکت Location (که فقط دو فیلد Lat و Lon دارد) بازمی‌گردانیم.
تابع دیگر، یعنی gps_thread، ابتدا منتظر موقعیت اولیه می‌ماند. پس از دریافت موقعیت اولیه، دائما موقعیت هر لحظه‌ی دستگاه را دریافت می‌کند و با استفاده از توابع ریاضی کمکی تعریف شده در فایل location_utils، فاصله‌ی خودرو را از مکان اولیه آن محاسبه کرده و به کلاس Manager اطلاع رسانی می‌کند.

★ توجه کنید که برای فعال‌سازی ماژول gps، نیاز داریم تا پورت‌های سریال را فعال کنیم. به همین منظور، فایل‌های boot/config.txt و boot/cmdline.txt را طبق توضیحات داده شده در این لینک تغییر می‌دهیم. با این کار، ماژول gps روی پورت سریال ttyAMA0 فعال خواهد شد.

اتصال به رزبری‌پای

برای اتصال ماژول GPS به رزبری‌پای، نیاز به سه کابل داریم. اتصالات را به صورت زیر برقرار می‌کنیم:
پایه‌ی Vcc جی‌پی‌اس ← پورت Vcc رزبری
پایه‌ی Tx جی‌پی‌اس ← پورت Rx رزبری (پایه‌ی GPIO 15)
پایه‌ی Gnd جی‌پی‌اس ← پایه‌ی GND رزبری

ماژول بلوتوث

فایل bluetooth_server.py مربوط به فعالسازی این ماژول می‌باشد. داخل این فایل سه تابع اصلی وجود دارد. تابع connect، منتظر یک اتصال از سمت سرور به بلوتوث باقی می‌ماند. تابع send_message یک رشته‌ی متنی را تبدیل به یک آرایه‌ای از بایت‌ها کرده و داخل سوکت باز شده می‌نویسد. تابع read_message مداوما از روی سوکت اطلاعات ارسالی را می‌خواند و مطابق دستور ارسال شده، عملیاتی را انجام می‌دهد. طبیعتا این تابع باید روی یک ترد جداگانه اجرا شود و این کار داخل فایل setup.py انجام می‌شود.

★ توجه کنید که به علت فعال‌سازی پورت serial برای ماژول gps، ماژول بلوتوث غیر فعال می‌شود. برای فعال‌سازی آن، باید داخل فایل config.txt خط زیر را اضافه کنیم:

dtoverlay=pi3-miniuart-bt

با انجام این کار، ماژول بلوتوث روی سریال ttyS0 فعال خواهد شد و تداخلی با gps نخواهد داشت.

ماژول buzzer

فایل buzzer.py و machine.py

چونکه در این پروژه از یک بازر passive استفاده کردیم، امکان ایجاد طیف وسیعی از نت‌ها و فرکانس‌های صوتی را داشتیم. بنابراین با اندکی جستجو، یک کتابخانه‌ی متن‌باز داخل github پیدا کردیم که می‌توان با استفاده از آن، روی ماژول passive buzzer یک آهنگ دلخواه را پخش کرد.
تابع play_song، از داخل لیست آهنگ‌ها، یکی را انتخاب کرده و پس از راه‌اندازی پورت مربوطه، شروع به پخش آهنگ می‌کند.
فایل machine.py نیز برای compatibility برای رزبری ۳ استفاده شده و کاملا حاوی توابع این کتابخانه می‌باشد.

اتصال به رزبری

ماژول بازر تنها سه پایه‌ی ورودی دارد. آن‌ها را به این صورت متصل می‌کنیم:
پایه‌ی Vcc ← پین Vcc رزبری
پایه‌ی GND ← پین GND رزبری
پایه‌ی I/O ← پین GPIO 26 رزبری

ماژول فن و رله

فایل fan.py

تابع setup، تنظیمات مربوط به پایه‌ی رزبری را انجام می‌دهد؛ یعنی پایه‌ی GPIO 16 را به عنوان خروجی قرار داده و مقدار ولتاژ آن‌را نیز تعیین می‌کند.
توابع turn_on و turn_off هم تنها ولتاژ این پایه را بین ۰ و ۳ ولت تغییر می‌دهند.

★ توجه کنید که فن داده شده تنها امکان کار کردن با ولتاژ ۵ ولت را داشت و به همین دلیل به جای اینکه آن‌را مستقیما به خود رزبری متصل کنیم، از ماژول رله استفاده کردیم.

اتصال به رزبری

اتصالات به صورت زیر انجام می‌شوند:
ورودی Vcc رله ← پین Vcc رزبری
ورودی GND رله ← پین GND رزبری
ورودی Signal رله ← پین GPIO 16 رزبری
خروجی اول رله ← ورودی Vcc فن
خروجی دوم رله ← پین Vcc رزبری
ورودی GND فن ← پین GND رزبری
با انجام اتصالات فوق، هنگام روشن شدن رله، دو خروجی بهم متصل می‌شوند و این یعنی ورودی Vcc فن به پین Vcc رزبری وصل می‌شود و فن روشن می‌شود.

ماژول وای‌فای و دوربین

فایل wifi.py و camera.py

ماژول وای‌فای به صورت پیش‌فرض روی رزبری قرار دارد و برای تنظیم و اتصال آن کار خاصی انجام نمی‌دهیم.
داخل فایل wifi.py هم تنها یک تابع برای آپلود کردن تصویر به سرور وجود دارد؛ این تابع به این صورت عمل می‌کند که آدرس یک فایل تصویر را دریافت کرده و آن‌را به url مشخص شده برای سرور ارسال می‌کند.
برای استفاده از ماژول دوربین نیز، از دوربین موبایل استفاده می‌کنیم. برای اینکه دوربین موبایل روی رزبری قابل استفاده باشد، برنامه IP Webcam را روی موبایل نصب می‌کنیم. با این کار، دوربین گوشی از طریق وای‌فای در دسترس رزبری خواهد بود.
داخل فایل camera.py نیز تنها یک تابع وجود دارد که برای تصویربرداری از دوربین گوشی می‌باشد. این تابع نیز همانند تابع داخل wifi.py، تنها یک درخواست Http به سرور دوربین (همان موبایل) ارسال می‌کند و پس از دانلود تصویر، آن‌را داخل آدرس داده شده در ورودی ذخیره می‌کند.

دیگر فایل‌های کلاینت

فایل bcolors.py

این فایل تنها برای نمایش خروجی رنگی داخل کلاینت استفاده شده و کاربرد خاصی ندارد. کاراکترهای کنترلی تعریف شده داخل این فایل اگر در ابتدای یک رشته قرار بگیرند، آن رشته به صورت رنگی در ترمینال چاپ خواهد شد.

فایل distance_utils

داخل این فایل کلاس Location قرار دارد که تنها به عنوان یک wrapper برای اطلاعات lat و lon عمل می‌کند و هیچ کاربرد خاصی ندارد.
تابع calculate_spherical_distance نیز برای محاسبه‌ی فاصله‌ی فضای دو Location به کار می‌رود و از آن استفاده می‌کنیم تا فاصله‌ی کنونی خودرو را با مکان اولیه‌ی آن بسنجیم.

فایل manager.py

این فایل مسئولیت انجام کارهای مختلف را بر حسب فاصله‌ی خودرو از مبدا دارد. اگر خودرو همچنان داخل محدوده باشد، هیچ عملیاتی انجام نمی‌شود. اگر خودرو خارج محدوده‌ی اول باشد، تنها با بلوتوث به ادمین اطلاع رسانی می‌شود و زنگی از بازر پخش می‌شود.
اگر خودرو از محدوده‌ی دوم نیز خارج شود، موتور خودرو به صورت خودکار خاموش شده و تصویری برای ادمین ارسال می‌شود.

فایل setup.p

این فایل همان فایل اصلی است که بر روی کلاینت اجرا می‌شود. مسئولیت آن شروع اولیه موتور خودرو، راه اندازی انواع ماژول‌های مختلف، دریافت مکان اولیه و اجرای تردهای مخصوص gps و bluetooth می‌باشد.

★ توجه کنید که در ابتدای این برنامه، mode اجرا دریافت می‌شود. به علت عدم امکان جا به جایی دستگاه هنگام اجرای آزمایشی، با استفاده از مود debug می‌توانیم فاصله‌ها را دستی به manager اعلام کنیم.

طراحی پروژه - سرور

سرور این برنامه بسیار ساده و مینیمال بوده و تنها دو مسئولیت به عهده دارد:

- دریافت و ذخیره سازی تصاویر آپلود شده داخل دیتابیس
- اتصال به کلاینت با استفاده از بلوتوث و امکان دریافت و ارسال پیام از طریق آن.

لازم به ذکر است که این پروژه یک webserver ساده بوده و از فریم‌ورک Django برای پیاده‌سازی آن استفاده شده.
داخل پوشه‌ی uploader، فایل views.py، تابع مربوط به دریافت تصاویر آپلود شده قرار دارد. این تابع پس از تایید صحت فایل ارسال شده، آن‌را به صورت یک مدل داخل دیتابیس ذخیره می‌کند.
داخل پوشه‌ی scripts نیز، فایل bluetooth_client.py ابتدا با استفاده از MAC address به بلوتوث کلاینت متصل شده و سپس در دو ترد مجزا، امکان دریافت و ارسال پیام را به صورت جداگانه مهیا می‌کند. محتوای این

فایل بسیار شبیه فایل بلوتوث کلاینت می‌باشد؛ به همین دلیل از توضیحات تکمیلی و اضافه برای این فایل پرهیز می‌کنیم.

نتایج و دستاوردها

در طی پیاده‌سازی این پروژه، با فرایندهای مختلفی در سیستم‌های نهفته آشنا شدیم.
نصب سیستم عامل و راه‌اندازی رزبری به طوری که روی یک شبکه‌ی داخلی قابل دسترسی و SSH زدن داشته باشد، یکی از مهارت‌های مهمی بود که در طی راه‌اندازی این پروژه یاد گرفتیم.[1]
هنگام اتصال ماژول neo 6 GPS به رزبری، متوجه شدیم که این ماژول در اصل برای بوردرهای آردوییِنو ساخته شده و برای اتصال آن به رزبری، نیازمند اعمال تغییراتی در تنظیمات رزبری و یا استفاده از یک مترجم هستیم. بدون اعمال این تغییرات، پیام‌های ارسال شده از سمت این ماژول کاملا بی‌معنی و غیر قابل استفاده بودند.[2]
پس از اتصال GPS و فعال‌سازی پورت‌های سریال، ماژول بلوتوث که روی همین پورت‌ها فعال بود از دسترس خارج شد. طی تحقیقاتی متوجه شدیم که بوردهای رزبری نسل ۳ به علت اضافه کردن بلوتوث، دو پورت سریال دارند و باید پورت سریال بلوتوث را روی پورت دیگری تنظیم کرد تا امکان استفاده از هر دو ماژول ممکن باشد.[3]
برای استفاده از بازر passive، متوجه شدیم که می‌توان با استفاده از Pulse Width Modulation یا PWM فرکانس‌های مختلفی را تنها با یک سیگنال صفر و یک تولید کرد. هم‌چنین یک کتابخانه برای تبدیل نت‌های مختلف موسیقی به سیگنال مورد نیاز برای تولید آن یافتیم.[4]

منابع

1. https://www.raspberrypi-spy.co.uk/2017/04/manually-setting-up-pi-wifi-using-wpa_supplicant-conf/
2. https://sparklers-the-makers.github.io/blog/robotics/use-neo-6m-module-with-raspberry-pi/
3. https://spellfoundry.com/2016/05/29/configuring-gpio-serial-port-raspbian-jessie-including-pi-3-4/
4. https://github.com/james1236/buzzer_music