OpenBus Navigator

Giancarlo Marín Hernández
giancarlo.marin@ucr.ac.cr

Stuart Leal Quesada
stuart.leal23@gmail.com

Paula Hernandez Góchez
paulagochez.h@gmail.com

I. Resumen

Propuesta para el desarrollo de una aplicación que mejore la experiencia de viaje buses del transporte público en Costa Rica. Podiendo determinar las rutas y paradas de un autobús de transporte público en el cual la adquisición de datos sea proporcionada por los mismos usuarios, para evitar el contacto directo con alguna empresa de autobúses o el CTP, ytener autonomía, independencia y generalización a la hora del desarrollo.

II. Introducción

En Costa Rica la deficiencia de información con respecto a rutas y paradas del transporte público provoca a los usuarios la pérdida de tiempo, dinero dado que en ocasiones deben tomar otros tipos de transporte o desplazarse en otra ruta sin seleccionar la más óptima para su viaje por el simple desconocimiento de donde se toma y por donde pasa determinado bus.

Una de las formas en que se suelen solucionar estas dudas por parte del usuario es consultándole personalmente al chofer del bus o a otra persona que se encuentre en determinada parada, por lo que surge la siguiente pregunta: ¿De qué manera más eficiente se podría obtener los datos de las rutas y paradas de los buses de transporte público en tiempo real y desde cualquier sitio con el uso de internet?

Es por esto que se plantea una solución en el cual se tiene como fin la mejorar la información proporcionada al usuario final sobre las paradas oficiales, rutas proporcionadas según su posición y destino, además de la ruta del bus en tiempo real, con una proyección a determinar los horarios por los que cierto bus se localiza en cierto punto de la ruta sin la utilización de equipos físicos adicionales alojados en el bus.

III. Nota Histórica

En Costa Rica los intentos por arreglar el problema descrito han sido abarcados parcialmente, por la aplicación Cazadora UCR y Busmaps, ambas disponibles en la App Store de Android. Sin embargo estas aplicaciones poseen horarios fijos, que no toman en cuenta la variabilidad de horas que se da por las presas en horas pico. Junto con esto está su alta dependencia de las empresas de transporte ya que la base de datos de ambas está basado en lo proporcionnado por las respectivas empresas de buses.

IV. Objetivo general

- Desarrollar una aplicación para mejorar la experiencia de viajar por medio del sistema de buses costarricense bajo una estructura de datos definida

V. Objetivos específicos

- Analizar el tipo datos necesarios.
- Conceptualizar la estructura de datos.
- Determinar la estructura de datos adecuada para almacenar estos datos.
- Brindar la información de rutas de interés para el usuario.

VI. Metodología

Para el desarrollo de este proyecto, se dividirá en etapas que nos llevaran al desarrollo optimo de la aplicación.

Etapa I. Análisis del tipo de datos necesarios En esta etapa se recopilaran datos de rutas ejemplo con variabilidad suficiente y para poder contemplar posibles error. Primeramente se encontrará una aplicación disponible en la tienda de Android (*realizado por Stuart Leal*) que permita mapear una ruta, y recopilar el archivo con los puntos ó $waypoints$ de manera que se pueda implementar dicho archivo en un programa que procese dichos puntos. Simultaneamente con esto, se investigará posibles fuentes de error causados por los GPS utilizados regularmente y como esto podría afectar la recopilación de datos.(*realizado por Paula Gochez*)

Etapa II. Conceptualización de la estructura de datos En esta etapa se definirá la estructura de datos más adecuada para almecenar estos datos por lo que mediante z(*realizado por Giancarlo Marín*). Así mismo se definirá como el programa que va a procesar dichos puntos y como asociarlos a un mapa, así como analizar la mejor manera de representar las rutas encontradas para crear en base a ellos una ruta, que se pueda realimentar con otro archivo de medición posteriormente, y optimize dicha ruta con base a toda la información admnistrada (*realizado por los tres estudiantes*). Posterior a esto, se buscará la manera en implementar la ruta creada, a un mapa del sitio. Probablemente dicho mapa será realizado como parte del proyecto, pero también existe la posibilidad de utilizar OpenStreetMap o algun otro mapa libre disponible que nos facilite la tarea (*realizado por Paula Góchez*).

Etapa III. Diseño y prueba del programa base En esta etapa se implementará la estructura de datos analizada en las

etapas anteriores y se diseñará el código base para realizar funciones que manejen esos datos y los metan directamente en esas estructuras de datos además que extraigan la informacion importante de los datos dados.
Se programará y probará nuestro sistema para encontrar pulgas, de manera que podamos optimizar el código y a su vez hacer un sistema más robusto (*realizado por los tres estudiante*).
Finalmente, se buscaría la forma de automatizar el proceso, de manera que no haya que exportar los archivos manualmente, convirtiendo la tarea cada vez más agradable y sencilla para el usuario.

VII. Desarrollo

Etapa I Se recopilaron datos de rutas ejemplo con variabilidad suficiente y para poder contemplar posibles error. Para esto se utilizó la aplicación Open GPS Tracker, encontrada en la App Store de Android. Esta aplicación guarda la información de la ruta como un track con distintos $waypoints$ en archivo tipo .gpx que tiene información como nombre del track,fecha,hora,velocidad,exactitud, latitud y longitud. Se tomaron aproximadamente 25 datos de la ruta del bus Universitario hacia la ciudad de la investigación partiendo de las distintas paradas encontradas en Finca 1.
Para el desarrollo del proyecto se encontró que los GPS tipicamente tienen 6 clases de errores [1].

- Error por efemérides, estos son errores de transmisión por la ubicación del satélite
- Error por el reloj de satélite, errores en el reloj de transmisión
- Ionosfera, la señal transmitida por el GPS se ve afectada por los efectos ionosféricos(como electrones libres)
- Troposfera, la señal transmitida por el GPS se ve afectada por los efectos de la troposfera (variaciones de presión,temperatura, humedad, que varían la rapidez de las ondas de radio)
- Múltiples trayectorias, errores causados por las señales reflejadas entran en la antena receptora
- Errores en el receptor,errores en la medición del receptor de la distancia causado por el ruido térmico, la precisión del software, y sesgos entre canales

Por esto se decidió que se utilizaría un radio de aproximadamente 2 metros para poder medir los waypoints que son vecinos, ya que en ese radio los datos siguen sobre la misma ruta y se minimizan los errores causados por cualquiera de las fuentes antes mencionadas.

Etapa II. Diseño de la estructura de datos Durante la etapa de recolección de datos, se comienza con el diseño de una estructura que contendrá los datos antes de de ser procesados en el programa. La estructura se basa en objetos de tipo *"parada"* que poseen las posiciones de latitud y longitud de los *waypoints* encontrados por medio del algoritmo descrito en la Fig.1)

Además se realiza un diseño de clases para el tratamiento de los waypoints vecinos. Finalmente, se realiza un diseño de grafos por medio de listas enlazadas que será utilizada para determinar las paradas siguiente y anterior. Estructuras basadas en [2]

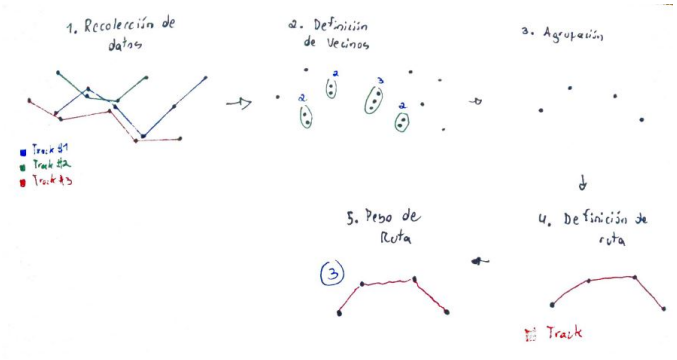

Figura 1: *Algoritmo para procesamiento de rutas*

Etapa III. Diseño y prueba del programa base Posterior a la recolección de datos se ingresó estos en una plataforma de software libre para mapear los datos encontrados, encontrando la Figura 1. la cual corresponde a la ruta del bus interno de la Universidad de Costa Rica. Ver Fig.2

Figura 2: *Representación de la ruta del bus universitario*

Luego, con la estructura antes mencionanda se realizaron distintos scripts y librerías en python, representados básicamente en el diagrama de la Figura 2.
La primera parte del programa se encarga de tomar los datos del servidor (por el momento la carpeta de dropbox del usuario) y parsearlos para generar datos "amigables" en archivos de texto para procesarlo posteriormente. Sin embargo el siguiente paso para desarrollar nuestro software es usar una librería que en el momento haga la conversión para no gastar el doble de espacio en el disco duro. Esto se puede lograr con *gpxpy*, una librería que se encarga de procesar y producir archivos gpx.
Esta primera parte del programa fue desarrollada completamente en esta primera etapa del proyecto. Hay tres .py principales: *request_acces, dropbox_download, y gpx_parser.* Su funcionalidad se muestra en la Figura 2 donde se muestra lo que genera cada uno de ellos. A grandes rasgos entre los tres realizan las tareas descritas anteriormente.
La segunda etapa del programa se encarga de la parte interesante: procesa la información que se encuentra en los archivos .gpx y lo traduce a puntos de parada. Este algoritmo se encuentra bien explicado en el código de *find_stops.py* en

la página de github del proyecto.
En este punto necesitamos la creación de un nuevo método que implemente la librería *gpxpy* para poder generar *waypoints* a partir de esos puntos encontrados por *find_stops.py* y probar que efectivamente encontró las paradas correctamente.

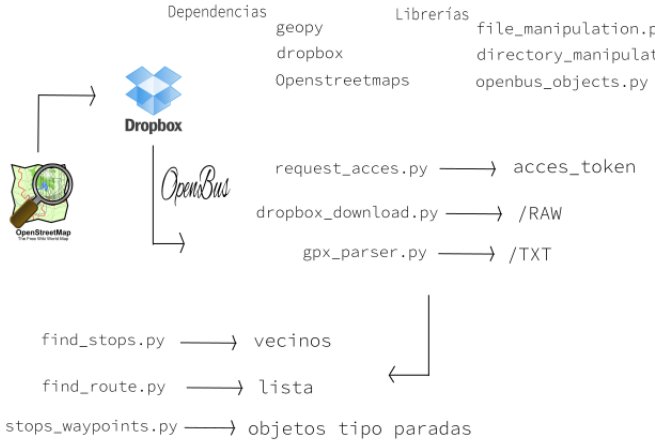

Figura 3: *Diagrama del programa*

Los próximos pasos de nuestros proyecto serán la utilización de la clase que contiene objetos tipo parada, para almacenar la información ya encontrada en el disco duro. Encontrar un algoritmo que clasifique rutas diferentes, ya que *find_stops.py* asume que la lista de archivos .gpx que le entran son de la misma ruta. Además encontrar un algoritmo de retroalimentación, para no tener que volver a calcular todo desde cero cada vez que se añada un nuevo waypoint al servidor, si no que compare el waypoint nuevo con la información encontrada anteriormente.
También lograr introducir información de links en la clase parada, y crear un método que encuentre dichos links. Una vez que todo esto se encuentre funcional podríamos agregar funciones más interesantes a la aplicación, como por ejemplo usar Dijkstra para encontrar entre las rutas de buses existentes, cuál es la mejor para llegar a un punto destino.
La documentación de las dependencias y cómo instalarlas se encuentran en la página de github en el README del repositorio, y el código mismo se encuentra en github también.

VIII. ANEXOS: CRONOGRAMA

Cuadro I: *Cronograma*

Tarea	Se 11-1
Recopilar los datos de las ruta ejemplo por medio ”waypoints”GPS	
Definir la estructura que albergará los datos	
Determinar los posibles errores GPS a la hora de captura de datos	
Programar la forma de agregar los datos obtenidos a la estructura definida	
Investigar el algoritmo para el reconocimiento de nodos (paradas) y ruta principal	
Automatizar el reconocimiento de paradas y rutas por medio del algoritmo y determinar como se implementará	
Corrección a errores en el programa y definición de recomendaciones ”pasos a seguir”para continuar del proyecto	

REFERENCIAS

[1] Mohinder G., Laurence R. Angus A.,2007.Global Positioning Systems, Inertial Navigation and Integration New Jersey, U.S.A

[2] Cormen, T., Leiserson, C (2009). *Introdction to Algorithms* M.I.T, MA, USA. Third Edition