Proyecto Final Diplomado Programación en Python y DevNet

Andres Guillermo Castellanos Alarcon
Fundación Universitaria del Área Andina
Bogotá D.C, Colombia
e-mail: agcastellanosa21@gmail.com

***Resumen*. Se pretende explicar de forma general la elaboración del dispositivo GPS TRACKING, para el seguimiento de ubicación de vehículos (Transporte de mercancía, transporte intermunicipal, etc) basado en coordenadas geográficas (Latitud y Longitud), el desarrollo de este dispositivo está conformado por el microcontrolador ESP32 y el modulo NEO6M GPS, el propósito de este prototipo tiene como finalidad poner en práctica los conocimientos adquiridos en el lenguaje de programación Python.**

***Abstract.* It is intended to explain in a general way the development of the GPS TRACKING device, for tracking the location of vehicles (Transport of merchandise, inter-municipal transport, etc.) based on geographical coordinates (Latitude and Longitude), the development of this device is made up of the microcontroller ESP32 and the NEO6M GPS module, the purpose of this prototype is to put into practice the knowledge acquired in the Python programming language.**

I. Introducción

Actualmente el tiempo y la seguridad es muy importante para las personas y las empresas, por lo tanto el conocer la ubicación aproximada de un vehículo de reparto o transporte puede ayudar a ahorrar tiempo, generar confianza y mejorar diferentes proceso logísticos de una compañía, esto puede lograr por medio del análisis de las diferentes ubicaciones que reporta las diferentes unidades monitoreadas. GPS TRACKING es un prototipo para monitorear las diferentes unidades, permitiendo conocer la ubicación aproximada (casi exacta) de las mismas.

II. Funcionamiento

GPS TRACKER, funciona con una **ESP32**, un módulo **GPS-NEO6M** y una fuente de energía de 6v, una vez se conecta a la fuente de energía el prototipo abre hilos para conectarse simultáneamente a la red Wi-Fi configurada al momento de la programación y para conectarse por medio el módulo GPS a satélites cercanos, para obtener las coordenadas actuales de la unidad.

Una vez se establezca conexión con el GPS, cada 30 segundo se verifica el estado de conexión Wi-Fi, si está conectado se reporta la ubicación obtenida a una base de datos en un servidor remoto que se haya configurado durante el proceso de desarrollo.

El prototipo cuenta con una pantalla **OLED** integrada en la **ESP32**, la cual muestra cada 10 segundos el estado de conexión de la red Wi-Fi y el GPS:

Si no hay conexión con Wi-Fi o a GPS se muestra el siguiente mensaje:

Fig 1. Mensaje conectando a red Wi-Fi

Fig 2. Mensaje conectando a GPS

Una vez se establezca conexión a la red Wi-Fi se muestra en la pantalla OLED un mensaje con la dirección IP obtenida:

Fig 3. Mensaje Wi-Fi conectado

Una vez se establezca conexión con GPS la pantalla OLED muestra cada 10 segundos las coordenadas obtenidas con un delay de 5 segundos entre longitud y latitud:

Fig 4. GPS Conectado

Otra forma de conocer el estado de la conexión Wi-Fi y GPS es por medio de luces led así:

a.) Cuando el led de color verde parpadea de forma lenta, quiere decir que está intentando conectarse a una red, si el led de color verde parpadea de forma rápida, significa que ya está conectado a una red Wi-Fi.

b.) Cuando el led de color azul parpadea cada 30 segundos, indica que está conectado a algún satélite y está enviando datos al servidor.

Adicional se tiene un sitio web donde se puede hacer un monitoreo en tiempo real de las unidades en total o detallada mediante de un mapa con el API HERE.

Fig 5. Unidades localizadas

III. COMO SE HIZO

El dispositivo está construido con un microcontrolador **ESP32** con pantalla **OLED** de 0.96, un **chasis-2WD-B**, 2 servomotores, módulo NEO6M, un controlador servo **L298N**, luces led azul, luces led verde y una protoboard

Fig 6. Modulo ESP32 OLED, ref: mercadolibre

Fig 7. Chasis-2WD-B

Fig 8. Modulo GPS NEO-6M

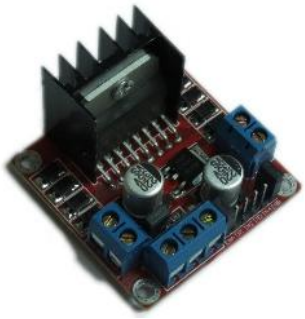

Fig 9. Controlador servo L298N

Se tienen dos (2) proyectos con código fuente:

FUAASEGUIMIENTO-WEB:

Este proyecto contiene el código fuente de la aplicación web, para el monitoreo de cada uno de las unidades de GPS. Este proyecto está

desarrollado bajo el lenguaje de programación Python, con programación orientada a objetos, utiliza flask para la creación de servicios rest api.

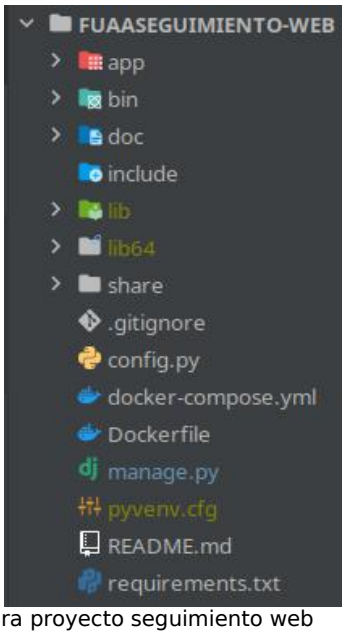

Fig 10. Estructura proyecto seguimiento web

En la figura enterior podemos ver la estrutura del proyecto seguimiento web:

a) **app:** Contiene todas las clases tipo repository, configuraciones y templates con estructura de proyecto flask.

b) **doc:** Contiene documentación del proyecto, como colecciones de postman y documento IEEE.

c) **config.py:** Archivo de configuración, con métodos y variables globales para uso en el back-end, aquí se puede configurar el tipo de debug, conexiones a bd, entre otras.

d) **manage.py:** Este archivo es el archivo principal, el cual permite configurar nombre del host, puerto y comandos.

e) **app/config:** e)Este paquete contiene clases de conexión a base de datos y parseo de datos para Mongo DB.

f) **app/config/ConnectionDB.py:** Esta clase es la encargada de realizar conexión a base de datos mongo mediante el método **connect**.

g) **app/config/MongoJSONEncoder:** Esta clase se encarga de serializar los objetos de colección a formato JSON para ser retornados en los servicios rest.

h) **app/config/ObjectIdConverter:** Esta clase se encarga de convertir los objetos tipo Id de mongo a string, esto resuelve error en la serialización de objetos de colección retornados por MongoDB.

i) **app/repository**: Contiene todas las clases tipo repository encargadas de gestionar los CRUDS a cada una de las colecciones, cada clase tiene el nombre de la colección que va afectar:

Fig 11. Clases tipo repository

Un ejemplo es la clase **LocalizationRepository** se encarga de Crear, Listar la colección localization en MongoDB.

j) **app/static**: Este paquete contiene los recursos estáticos como: imagenes, javascript y css.

k) **app/static/script/shere-map.js:** Este archivo contiene las funcionalidades javascript para la inicialización del mapa, actualización del punto de ubicación, en la herramienta HERE.

l) **app/static/web-services.js:** Este archivo javascript contiene dos métodos:

localizeAllCarts: Se encarga de consumir el servicio rest para obtener ubicación actual de las unidades.

localizeCart: Se encarga de consumir el servicio rest api para ubicar una unidad especifica.

m) **app/templates:** Directorio donde se encuentra las vistas html de cada una de las opciones del monitor, estas vistas tiene una plantilla general que contiene la importación de recusos estáticos llamada **base.html**

Para ver repositorio haga clic aquí.

FUAASEGUIMIENTO-MICROPYTHON

Este proyecto esta desarrollado con Micropython, el código fuente de este proyecto es el encargo de controlar el prototipo para la ubicación GPS.

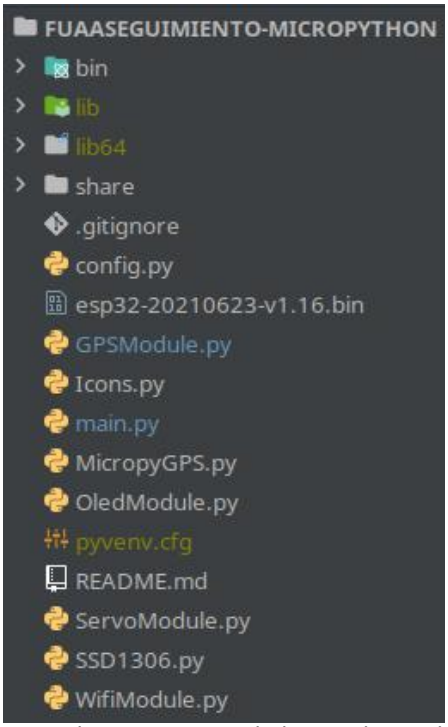

Fig 12. Estrutura de proyecto seguimiento micropython

En la figura anterior observamos la estructura del proyecto de seguimiento micropython:

a) **GPSModule.py**: Clase encargada de establecer conexión GPS al módulo **NEO6M** mediante conexion **UART**, esta clase contiene los métodos:

read: Este método es encargado leer el buffer devuelto por la conexión UART y a su vez hace el llamado del metodo **updateGeo** para actualizar la ubicación actual de la unidad.

connectingIndicator: Este método se encarga de enceder y apagar el led azul cada 500 ms, para indicar que esta estableciendo conexión GPS.

connectedIndicator: Este método se encarga de encender y apagar el led azu cada 50ms, esto para indicar que ya se establecio una conexión con GPS.

displayStatus: Metodo encargado de enviar datos al metodo **displayText** y **displayIcon** al controlador **SSD1306.py** para ser mostrados en la pantalla OLED.

updateGeo: Método para hacer llamado al api rest de flask, para actualizar la ubicación actual de la unidad, este es llamado por el método read, cada vez que se obtiene una ubicación geografíca.

b) **Icons.py**: Esta clase contiene dos métodos que devuelve array de bytes de los iconos para wifi y gps.

c) **MicropyGPS.py**: esta clase es encarga de serializar las respuestas del GPS obtenidas en el método **GPSModule.read**. Esta clase es obtenida del repositorio **inmcm/micropyGPS**.

https://github.com/inmcm/micropyGPS

d) **OledModule.py**: Esta clase se encarga mostrar texto e imagenes en la pantalla OLED, utiliza la clase **SSD1306.py**, esta clase es obtenida del repositorio **micropython/micropython**.

https://github.com/micropython/micropython

e) **ServoModule.py**: Esta clase se encarga de controlar el módulo **L298N**, en esta clase se encuentra los métodos:

forward: Mueve los servos permitiendo que el chasis se mueva hacia adelante.

back: Mueve los servos permitiendo que el chasis retroceda.

stop: Apaga los servos

f) **WifiModule.py**: Clase para controlar el módulo Wi-Fi integrado en la ESP32, esta clase contiene los métodos:

connect: Este metodo establece conexión con los datos de SSID y Clave establecidos en el archivo de configuración.

ipAddress: Devuelve la dirección IP asignada en la conexión con el router.

connectingIndicator: Enciende el led verde cada 500 ms, el cual indica que se esta intentando establecer conexión con el router.

connectedIndicator: Enciende cada 50 ms el led verde, indica que existe conexión entre el módulo Wi-Fi y el router.

displayStatus: muestra en pantalla OLED el icono y el estado de la conexión, si existe conexión muestra la direccion IP asignada.

IV. COMO SE USA

Tal como se menciono se tiene dos proyectos:

SEGUIMIENTO WEB:

1. Descargar el código fuente del repositorio.

2. Crear el entorno virtual en el directorio creado con la clonación del repositorio.

3. Activar el entorno virtual.

4. Instalar los paquetes definido en el fichero requiremets.txt.

```
click==8.0.1
dnspython==2.1.0
Flask==2.0.1
Flask-PyMongo==2.3.0
Flask-Script==2.0.6
importlib-metadata==4.6.1
isodate==0.6.0
itsdangerous==2.0.1
Jinja2==3.0.1
pymongo==3.12.0
PyMySQL==1.0.2
six==1.16.0
typing-extensions==3.10.0.0
Werkzeug==2.0.1
zipp==3.5.0
```

Fig 13. Dependencias seguimiento web

5. Modificar el archivo **ConnectionDB.py** con la URL y credenciales de la base de datos MongoDB.

```
def connect(self):
    return PyMongo(self.app, "mongodb+srv:
```

Fig 14. Conexion base de datos MongoBD

6. Una vez se configura el entorno virtual, se debe ejecutar el comando para iniciar el servidor flask:

python manage.py runserver

7. Una vez incie el servidor se accede en el navegador a la URL **http:localhost:8001**

8. Se muestra un sitio web donde se carga un mapa con la ubicación actual de cada una de las unidades que se van monitorear.

9. Hay tres (3) opciones de menú:

Inicio: Muestra el mapa general con las ubicaciones de cada una de las unidades.

Vehículos: Muestra el listado de los vehículos a monitorear, allí se muestra un botón "**Ubicación**" en cada registro para ubicar en detalle la unidad seleccionada.

Datos Prueba: Se encarga de importar datos de prueba para ver funcionamiento del mapa.

SEGUIMIENTO MICROPYTHON

1. Descargar el código fuente del repositorio.

2. Crear el entorno virtual en el directorio creado con la clonación del repositorio.

3. Activar el entorno virtual.

4. Montar el código fuente en la ESP32

5. Conectar la ESP32 a una fuente de alimentación de 6v.

6. Esperar que se incialice y listo.

Fig 15. Unidad conectada

REFERENCIAS

[1] ESP32 with Ublox NEO-6M - MicroPython Forum. (2019, 16 junio). MicroPython Forum. https://forum.micropython.org/viewtopic.php?t=7766&p=44311

[2] I.+.D.E. (s. f.-a). Plataforma Robótica FEETECH 1 nivel-robot-chasis. I+D Electrónica. Recuperado 31 de julio de 2021, de https://www.didacticaselectronicas.com/index.php/robotica/robot/chassis/plataforma-rob%C3%B3tica-aluminio-feetech-2-niveles-robot-carros-carritos-chassis-chasis-plataformas-rob%C3%B3ticas-rob%C3%B3ticos-feet-chasis-2wd-b-electr%C3%B3nica-i-d-detail

[3] I. (s. f.-b). GitHub - inmcm/micropyGPS: A Full Featured GPS NMEA-0183 sentence parser for use with Micropython and the PyBoard embedded platform. GitHub. Recuperado 31 de julio de 2021, de https://github.com/inmcm/micropyGPS

[4] M. (s. f.-c). GitHub - micropython/micropython: MicroPython - a lean and efficient Python implementation for microcontrollers and constrained systems. GitHub. Recuperado 31 de julio de 2021, de https://github.com/micropython/micropython

[5] Micropython - PWM DC Motor. (2018, 5 diciembre). [Vídeo]. YouTube. https://www.youtube.com/watch?v=RE9mvVx-gKw

[6] Santos, S. (2021, 30 junio). ESP32 Built-in OLED Board (Wemos Lolin32): Pinout, Libraries and OLED Control. Random Nerd Tutorials. https://randomnerdtutorials.com/esp32-built-in-oled-ssd1306/