Practica 2

Desarrollo del mapa electrónico de la carretera

SISTEMAS Y SERVICIOS DE NAVEGACIÓN CON GPS

Almudena Vila Álvaro

Luis Mariano García Morales

Álvaro Lameiro García

Alberto Luis Navarro Melgar

Rubén Mena Cabanillas

Adrián Ruiz Serrano

Para la realización de la práctica se van a utilizar un proyecto en Python con distintas funciones que serán explicadas en este documento. Además, se incluirán fotos del Campus Sur de la UPM, de INSIA, un receptor GPS y nuestro código que con las coordenadas trazará la trayectoria.

Acerca del receptor GPS, se quiere comentar que se trata del modelo GPS Trimble R4, el cual genera tramas de posicionamiento GGA siguiendo el estándar NMEA a una frecuencia de 10Hz.

En primer lugar, como se mencionó en la práctica 1 presentan dos clases que caracterizan las propiedades de dos elipsoides geodésicos distintos: el elipsoide de referencia europeo de 1950 (EU1950) y el elipsoide de referencia mundial de 1984 (WGS84).

datum.py

```
class EU1950:
    a = 6378388
    f = 1/297
    e2 = 0.00672267
    c = 6399936.608
class WGS84:
    a = 6378137
    f = 1/298.25722
    e2 = 0.00669437999013
    c = 6399593.626

```

En segundo lugar, se ha modificado la función **leer_datos_gps** mediante el uso de threads.

operaciones.py

```
def leer_datos_gps(queue, stop_event):
    ports = list(serial.tools.list_ports.comports())
    for p in ports:
        port = p.name

    #Configuracion del puerto serie
    puerto = serial.Serial('COM3', baudrate = 4800, bytesize = serial.EIGHTBITS,
                           parity = serial.PARITY_NONE, stopbits = serial.STOPBITS_ONE,
                           xonxoff = False, rtscts = False, dsrdtr = False)

    while not stop_event.is_set():
        data = puerto.readline().decode().strip()
        if data:
            campos = data.split(',')
            if campos[0]=='$GPGGA' and campos[2] != "":
            #Conversion de la latitud y longitud que ofrece la trama a grados decimales
                latitud = float (campos[2][:2]) + float(campos[2][2:]) /60
                longitud = float(campos[4][:3]) + float(campos[4][3:]) / 60

                if campos[3] == 'S':
                    latitud = -latitud
                if campos[5] == 'W':
                    longitud = -longitud

                queue.put(conversorUTM(latitud,longitud,WGS84))
```

En tercer lugar, la clase **Map** tiene la función **utm_a_pixel**, que como su nombre indica, pasa la coordenada utm a pixel.

Map.py

```
def utm_a_pixel(self, utm_x: float, utm_y: float):
    print(f"X: {utm_x:.2f}".ljust(24)+f"\nY: {utm_y:.2f}")

    #A cambiar según imagen
    #utm_x_min, utm_y_max = 446220.0, 4471000.0
    #utm_x_max, utm_y_min = 446400.0, 4470780.0
    utm_x_min, utm_y_max = 455406.00, 4474489.00
    utm_x_max, utm_y_min = 455647.80, 4474362.50

    x_size = utm_x_max - utm_x_min
    y_size = utm_y_max - utm_y_min

    x_scale = x_size/self.image.width
    y_scale = y_size/self.image.height

    pixel_x = (utm_x - utm_x_min) / (x_scale + self.x)
    pixel_y = (utm_y - utm_y_min) / (y_scale + self.y)

    bs = len (f"{pixel_x:.2f}")
    print(f"pixel_x:{pixel_x:.2f}".ljust(30-bs)+f"\npixel_y:{pixel_y:.2f}")
    print("------------------------------------------------------------")

    return pixel_x-self.markar_size/2, pixel_y-self.markar_size/2
```

En esta misma clase, la función **new_marker** indica exactamente el punto de la coordenada marcada. En dicha función se utiliza la clase **Marker** que añade un widget con la imagen del marcador.

```
def new_marker(self, utm_x: float, utm_y: float):
    pixel_x,pixel_y=self.utm_a_pixel(utm_x,utm_y)
    marker = Marker(pixel_x,pixel_y,size=(self.markar_size,self.markar_size))
    self.add_widget(marker)

    return marker
```

Seguida de la clase **Map**, la clase **MapaGPS** crea una interfaz de usuario con un mapa y muestra por pantalla las coordenadas de la trayectoria implementando todas las clases mencionadas anteriormente.

MapaGPS.py

```
class MapaGPS(BoxLayout):
    def __init__(self, **kwargs):
        super(MapaGPS, self).__init__(**kwargs)
        self.orientation = "horizontal"
        self.size_hint = (None, None)

        self.map = Map()
        self.add_widget(self.map)

        self.right_box = BoxLayout(orientation = "vertical", width = 300)

        self.buttons_box = BoxLayout(size_hint = (None, None), orientation = "horizontal", width = 500, height = 50)

        output_widget = OutputWidget(size_hint = (1,1))
        sys.stdout = StdoutRedirect(output_widget)

        self.size = self.map.width + self.right_box.width, self.map.height
        self.right_box.add_widget(output_widget)
        self.right_box.add_widget(self.buttons_box)

        self.add_widget(self.right_box)
```

Por último, todas las clases con sus respectivas funciones son recogidas en la clase **MyMainApp**.