Politechnika Wrocławska

Sprawozdanie 4

Ćwiczenie 4.Systemy nawigacji satelitarnej(GPS)

Krzysztof Zalewa

3.12.2024

Spis treści

1 Wstęp teoretyczny

1.1 Systemy nawigacji satelitarnej

1.1.1 Budowa i zasada działania systemu GPS

GPS składa się z konstelacji satelit umieszczonych na orbitach obecnie jest to aktywnych satelit i 3 zapasowe. GPS to system stadiometryczny co oznacza że pozycja wyznaczana jest na podstawie czasu propagacji sygnału z satelit. Satelita wysyła sygnał z pewną częstoliwością, gdy sygnał ten dociera do odbiornika czas jest mierzony i w połączeniu z czasem propagacji do innych widocznych satelit wyliczana jest lokalizacja(Rys1.)

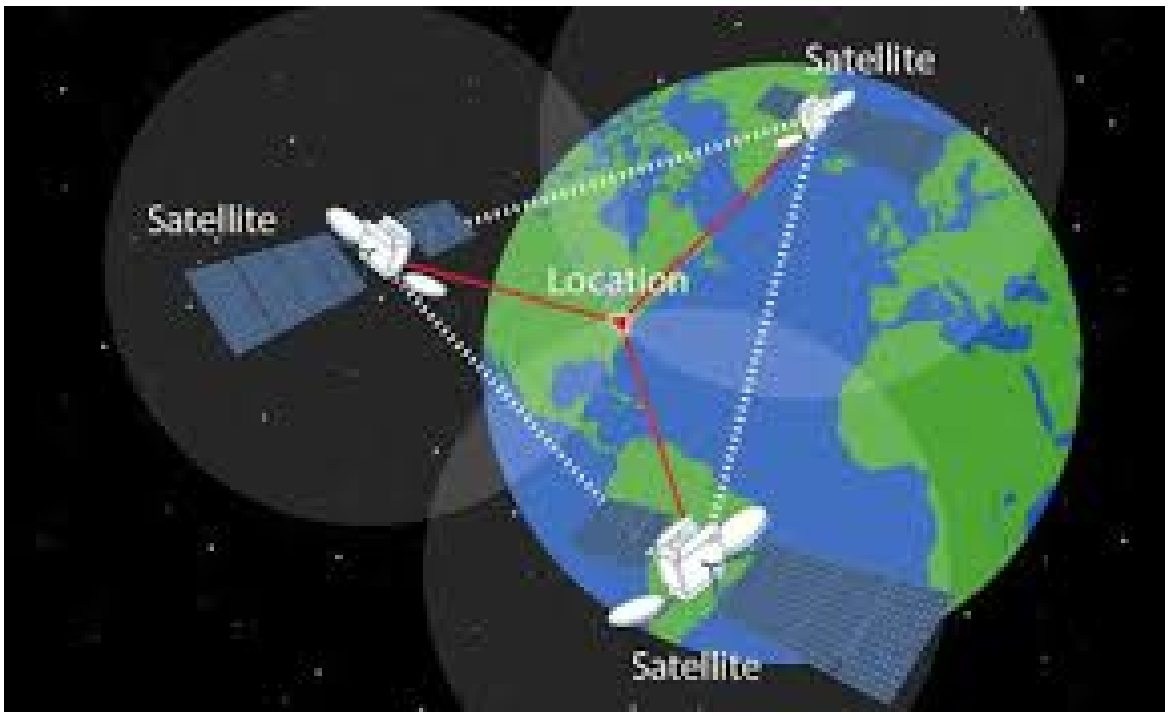

Figure 1: Schemat działania systemów stadiometrycznych

1.1.2 GLONASS [RF]

To Rosyjski system nawigacji satelitarnej. Tak jak pozostałe systemy składa się z kosmicznego, kontroli i użytkownika.Segment kosmiczny to 24 satelity umieszczone na 3 orbitach (po 8 na każdej). Satelity mają kąt inklinacji równy 64,8°co pozwala na całkowite pokrycie kuli ziemskiej. Segmnet kontroli to główna stacja kontrolna i kilka pomniejszych umieszconych w Europie,Rosji i Brazylii. Przez długi czas odbiorniki GLONASS używane były w sprzęcie wojskowym, obecnie jednak są one dostępne dla użytkowników cywilnych.

1.1.3 Galileo [EU]

To Europejski system nawigacji satelitarnej. Tak jak pozostałe systemy składa się z trzech segmentów kosmicznego, kontroli i użytkownika. Segment kosmiczny to obecnie 24 satelit (z 30 planowanych) na trzech orbitach. Satelity mają kąt inklinacji równy 56°. Co daje dobry sygnał głównie do serokości nawet 75°N (Najwyżej położony punkt w europie).Segment kontroli znajduje się głównie w państwach europejskich. Segment użytkownika to od samego początku (rok 2016) segment gospodarczy.

1.1.4 BeiDou [Chiny]

To Chiński system nawigacji satelitarnej. Obecna wersja (BeiDou-3) ma zasięg na całą kulę ziemską. Podbnie jak pozostałe systemy składa się z trzech segmentów kosmicznego,kontroli i użytkownika. Segment kosmiczny to 24 satelity umieszczone na 3 orbitach. Segment kontroli znajduję się w chinach. A segent użytkownika to głównie urządzenia militarne i komercjalne.

1.2 Struktura zdań NMEA

Protokół opublikowany przez National Marine Electronics Association służy do komunikacji między morskimi urządzeniami.Przykładowe zdanie NMEA:
$GPGGA,170834,N,41224.55000,08150.8500,W,1,05,1.5,280.2,M,-34.0,M,,,*75

nazwa	przykładowa wartość	opis
Początek sekwencji	$	początek sekwencji
identyfikator nadawcy	GP	GPS
rodzaj sekwencji	GGA	Global Positioning System Fix Data
czas informacji	170834	17:08:34 UTC
szerokość geograficzna	4124.5500, N	41° 24,5500' N (41° 24' 33" N)
długosc geograficzna	08150.8500, W	81° 50,8500' W (81° 50' 51" W)
jakość pozycji	1	jakość ustalonej pozycji (0 - nieważna, 1 - GPS, 2- DGPS...)
liczba satelitów	05	5 widocznych satelitów
Horizontal Dilution of Precision (HDOP)	1.5	dokładność pozycji (horyzontalna) = 1,5
wysokość	280.2, M	280,2 ponad średni poziom morza
wysokość geoidy ponad elipsoidę WGS84	-34.0, M	-34,0 metry
czas od aktualizacji danych DGPS	pusta	brak
identyfikator stacji nadającej poprawki różnicowe DGPS	pusta	brak
suma kontrolna	*75	używana do kontroli poprawności sentencji

Table 1: Odczyt przykładowego zdania NMEA

2 Zadanie laboratoryjne

2.1 Treść zadania

W ramach zadania laboratoryjnego należało napisać program który będzie obsługiwał transmisję z urządzenia GPS. Program ten miał być w stanie rozkodować otrzymane zdania NMEA i zlokalizować na mapie punkt w którym znajduje się urządzenie.

2.2 Opis działania programu

Po włączeniu program iteruje po wszystkich aktywnych portach COM. Do każdego portu wysyłane jest 20 pytań by sprawdzić czy podpięte urządzenie to GPS. Jeżeli w odpowiedzi program dostanie conajmniej jedno poprawne zdanie NMEA przechodzi on do głównej części zadanie. Co 200ms

program pobiera zdanie NMEA i sprawdza jego poprawność. Jeżeli zdanie jest poprawne to koordynaty wypisywane są w konsoli a na mapie pojawia się pinezka w odpowiednim miejscu.

2.3 Kod programu

.

```
import tkinter
import serial
import serial.tools.list_ports
import tkintermapview


def gga_nmea_to_coordinates(nmea_string):
    # Split the NMEA string by commas
    parts = nmea_string.split(',')

    # Check if the string is a GGA sentence
    if parts[0] != '$GPGGA':
        raise ValueError("Invalid GGA sentence")

    # Extract latitude and longitude
    lat_str = parts[2]
    lat_dir = parts[3]
    lon_str = parts[4]
    lon_dir = parts[5]

    # Convert latitude
    lat_degrees = int(lat_str[:2])  # First two characters are degrees
    lat_minutes = float(lat_str[2:])  # Remaining characters are minutes
    latitude = lat_degrees + (lat_minutes / 60)
    if lat_dir == 'S':
        latitude = -latitude  # South is negative

    # Convert longitude
    lon_degrees = int(lon_str[:3])  # First three characters are degrees
    lon_minutes = float(lon_str[3:])  # Remaining characters are minutes
    longitude = lon_degrees + (lon_minutes / 60)
    if lon_dir == 'W':
        longitude = -longitude  # West is negative

    return latitude, longitude, float(parts[9]), float(parts[11])


def parse_gga_time(sentence):
    """Parsuje zdanie NMEA typu $GPGGA i zwraca czas UTC w formacie HH:MM:SS."""
    if sentence.startswith("$GPGGA"):
        parts = sentence.split(',')
        if len(parts) >= 2 and parts[1]:  # Drugie pole to czas
            utc_time = parts[1]
            # Podział na godziny, minuty i sekundy
            hours = int(utc_time[:2]) + 1
            minutes = utc_time[2:4]
            seconds = utc_time[4:6]
            return f"{hours}:{minutes}:{seconds} UTC+1"
    return None
```

```
50
51
52  def initial_verification(raw_sentence):
53      try:
54          if raw_sentence[0] != '$' or raw_sentence[-1] != '\n':
55              raise Exception("wrong string")
56          xor = 0
57          for val in raw_sentence[1:raw_sentence.find('*') - 1].split(','):
58              for char in val:
59                  xor = xor ^ ord(char)
60          xor2 = 0
61          mult = 16
62          for char in raw_sentence[raw_sentence.find('*') + 1:
            ↪  raw_sentence.find('\r')]:
63              xor2 += int(char) * mult
64              mult /= 16
65          return xor == xor2
66
67      except Exception as e:
68          return False
69
70
71  def parse_rmc_date(sentence):
72      """Parsuje zdanie NMEA typu $GPRMC i zwraca datę w formacie DD-MM-YYYY."""
73      if sentence.startswith("$GPRMC"):
74          parts = sentence.split(',')
75          if len(parts) >= 10 and parts[9]:  # Dziesiąte pole to data
76              rmc_date = parts[9]
77              # Podział na dzień, miesiąc i rok
78              day = rmc_date[:2]
79              month = rmc_date[2:4]
80              year = rmc_date[4:6]
81              # Zakładamy, że lata zaczynają się od 2000 roku
82              full_year = f"20{year}" if int(year) < 50 else f"19{year}"
83              return f"{day}-{month}-{full_year}"
84      return None
85
86
87  def map_init(com_port):
88      root_tk = tkinter.Tk()
89      root_tk.geometry(f"{800}x{600}")
90      root_tk.title("map_view_example.py")
91
92      map_widget = tkintermapview.TkinterMapView(root_tk, width=800, height=600,
        ↪  corner_radius=0)
93      map_widget.place(relx=0.5, rely=0.5, anchor=tkinter.CENTER)
94      map_widget.set_position(51.109230, 17.060283)  # Wrocław, Poland
95      map_widget.set_zoom(10)
96
97      # Add labels for latitude, longitude, and altitude
98      lat_label = tkinter.Label(root_tk, text="Szerokość: ")
99      lat_label.place(relx=0.1, rely=0.9, anchor=tkinter.W)
100     lon_label = tkinter.Label(root_tk, text="Długość: ")
101     lon_label.place(relx=0.4, rely=0.9, anchor=tkinter.W)
102     alt_label = tkinter.Label(root_tk, text="Wysokość: ")
103     alt_label.place(relx=0.7, rely=0.9, anchor=tkinter.W)
```

```
104
105     root_tk.after(200, gps_main_loop, map_widget, com_port, lat_label, lon_label,
106                   alt_label)  # Schedule gps_main_loop to run after 200 ms
107     root_tk.mainloop()
108     return map_widget
109
110
111 def gps_main_loop(map_widget, com_port, lat_label, lon_label, alt_label):
112     marker = map_widget.set_position(51.109230, 17.060283, "Lokalizacja",
        ↪  marker=True)
113
114     def update_position():
115         raw_data = com_port.readline().decode('ascii', errors='replace')
116         if not initial_verification(raw_data):
117             map_widget.after(200, update_position)
118             return
119
120         raw_data = raw_data.strip()
121         print(f"\nOtrzymane zdanie NMEA: {raw_data}")
122         rmc_date = parse_rmc_date(raw_data)
123         if rmc_date:
124             print(f"Data RMC: {rmc_date}")
125
126         time_utc = parse_gga_time(raw_data)
127         print(f"Czas UTC: {time_utc}")
128
129         latitude, longitude, height, geoid_height =
            ↪  gga_nmea_to_coordinates(raw_data)
130         print(
131             f"Szerokość geograficzna: {latitude} N, długość geograficzna:
                ↪  {longitude} E\n Wysokość nad poziomem morza : {height} \t Wysokość
                ↪  Geoidy : {geoid_height}")
132         marker.set_position(latitude, longitude)
133         map_widget.set_position(latitude, longitude)
134         # Update labels with the new position and altitude
135         lat_label.config(text=f"Szerokość: {latitude}")
136         lon_label.config(text=f"Długość: {longitude}")
137         alt_label.config(text=f"Wysokość: {height}")
138
139         map_widget.after(200, update_position)  # Schedule the next update
140
141     update_position()  # Start the first update
142
143
144 def main():
145     ports = serial.tools.list_ports.comports()
146     for port, desc, hwid in sorted(ports):
147         print(f"{port}: {desc}")
148         try:
149             com_port = serial.Serial(port, baudrate=9600, timeout=1)
150             good_port = False
151             for i in range(0, 20):
152                 raw_data = com_port.readline().decode('ascii', errors='replace')
153                 if raw_data:
154                     if not raw_data[0] == '$':
155                         continue
```

```
                    else:
                        good_port = True
                        break
            if good_port:
                print(f"Nasłuchiwanie na porcie {port}...")
                map_init(com_port)
        except serial.SerialException as e:
            print(f"Błąd: Nie można otworzyć portu szeregowego - {e}")
            return


if __name__ == "__main__":
    main()
```

Fragment kodu 1: Kodu programu

3 Wnioski

Program udało się ukończyć na zajęciach i działał poprawnie. Na mapie pinezka pojawiała się w okolicach budynków C1,C2 i C3(W którym znajdowało się urządzenie).

4 Źródła

- https://gisplay.pl/nawigacja-satelitarna/glonass.html
- https://www.esa.int/Applications/Satellite_navigation/Galileo/What_is_Galileo
- https://defence-industry-space.ec.europa.eu/eu-space/galileo-satellite-navigation_en
- https://www.seinxon.com/en-eu/blogs/blog-posts/how-does-a-gps-tracker-work
- https://www.geotab.com/blog/what-is-gps/