

Urządzenie z akcelerometrem i GPS

Kamil Ambroży 145259
Przemysław Woźniak 145423
Jakub Wróbel 145288

20 luty 2022

1 Cele i zakres projektu

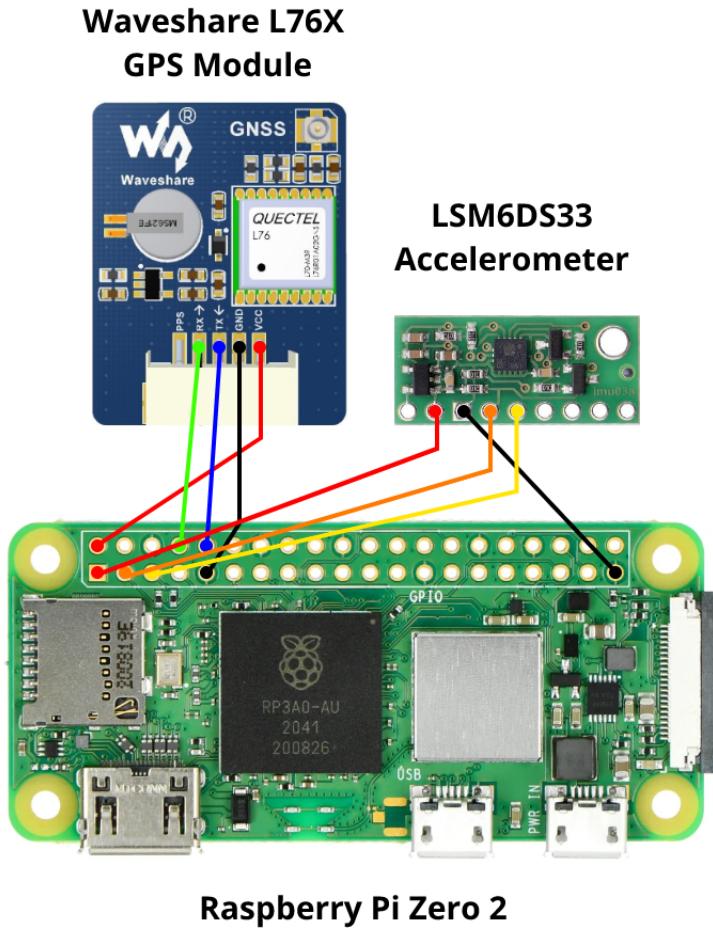
Celem projektu była budowa urządzenia, które wyświetlało dane z akcelerometru oraz GPSu na stronie internetowej.

Jako główne urządzenie wykorzystaliśmy Raspberry Pi Zero 2 z wbudowanym modułem Wi-Fi. Jako akcelerometr użyliśmy modułu LSM6DS33 firmy Pololu. Za moduł GPS posłużyliśmy się modułem L76X firmy Waveshare.

Urządzenie w celu poprawnego działania wymaga dostarczenia mu Internetu. Po połączeniu się z siecią – taką samą jak Raspberry Pi – i wejściu na odpowiedni adres ukazuje się strona z uaktualnianymi na żywo wynikami pomiarów akcelerometru oraz mapą, na której są naniesione dane z modułu GPS.

2 Schematy

2.1 Schemat połączeniowy



3 Projekt a założenia

Początkowym założeniem było posiadanie serwera strony internetowej po stronie klienta. Dane z bazy danych znajdującej się na Raspberry miały być przesyłane do urządzenia klienta. Ostatecznie jednak to po stronie urządzenia znajduje się serwer stron oraz cała baza danych. Kolejnym pomysłem było rozgłaszenie własnej sieci WiFi przez urządzenie, jednak z powodu konieczności wyświetlenia Map Google przez klienta, musiał on być połączony

z internetem.

Początkowo całe urządzenie miało być zamknięte w specjalnej obudowie wydrukowanej na drukarce 3D, jednak wymiary zostały źle dobrane i musielibyśmy zrezygnować z tego pomysłu. Zamiast tego postanowiliśmy umieścić urządzenie w odpowiednio dobranym, kartonowym opakowaniu.

4 Kod

```
1 def read_gps_data():
2     ser = serial.Serial("/dev/ttyS0", 9600)    # Open port with
3         baud rate
4     last_lat = 0
5     last_lon = 0
6     while True:
7         received_data = ser.read()    # read serial port
8         data_left = ser.inWaiting()   # check for remaining byte
9         received_data += ser.read(data_left)
10        my_list = received_data.decode("utf-8").split('$')
11        try:
12            lat = np.mean(
13                [math.modf(float(item.split(',') [1]) / 100)[1] + math.
14                 modf(float(item.split(',') [1]) / 100)[0] * (100 / 60)
15                 for item in my_list if item.startswith('GNGLL')])
16            last_lat = lat
17        except:
18            lat = last_lat
19        try:
20            lon = np.mean(
21                [math.modf(float(item.split(',') [3]) / 100)[1] + math.
22                 modf(float(item.split(',') [3]) / 100)[0] * (100 / 60)
23                 for item in my_list if item.startswith('GNGLL')])
24            last_lon = lon
25        except:
26            lon = last_lon
27        if (not np.isnan(lat)) and (not np.isnan(lon)):
28            try:
29                socketio.emit('gps_data', {'cords': [lat, lon]})
30            except:
31                print('Could not emit')
32            print(lat, lon)
33            socketio.sleep(1)
```

Listing 1: GPS

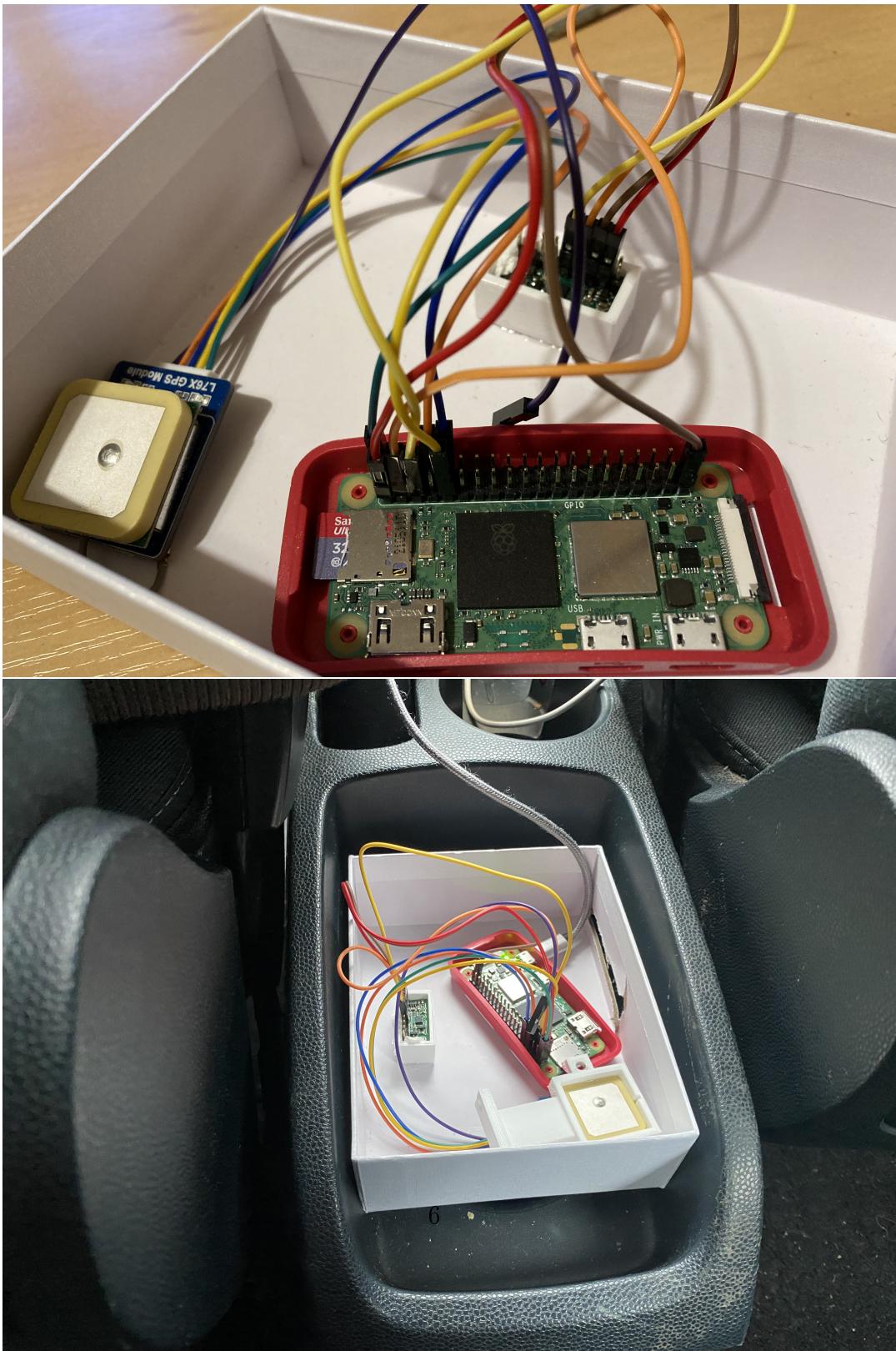
Kod służy do odczytu danych z GPSa przy pomocy portu szeregowego. Dane przesyłane są w formacie NMEA, przez co powyższy kod służy również do przetwarzania współrzędnych do formatu wspieranego przez Mapy Google oraz przesłaniu ich na stronę.

```
1 def read_accelerometer_data():
2     i2c = board.I2C()
3     sensor = LSM6DS33(i2c)
4     """Example of how to send server generated events to
5      clients."""
6     while True:
7         acc = [sensor.acceleration[0]/9.81, sensor.acceleration
8             [1]/9.81, sensor.acceleration[2]/9.81]
9         socketio.emit('acc_data',
10             {'acc': acc})
11         socketio.sleep(0.05)
```

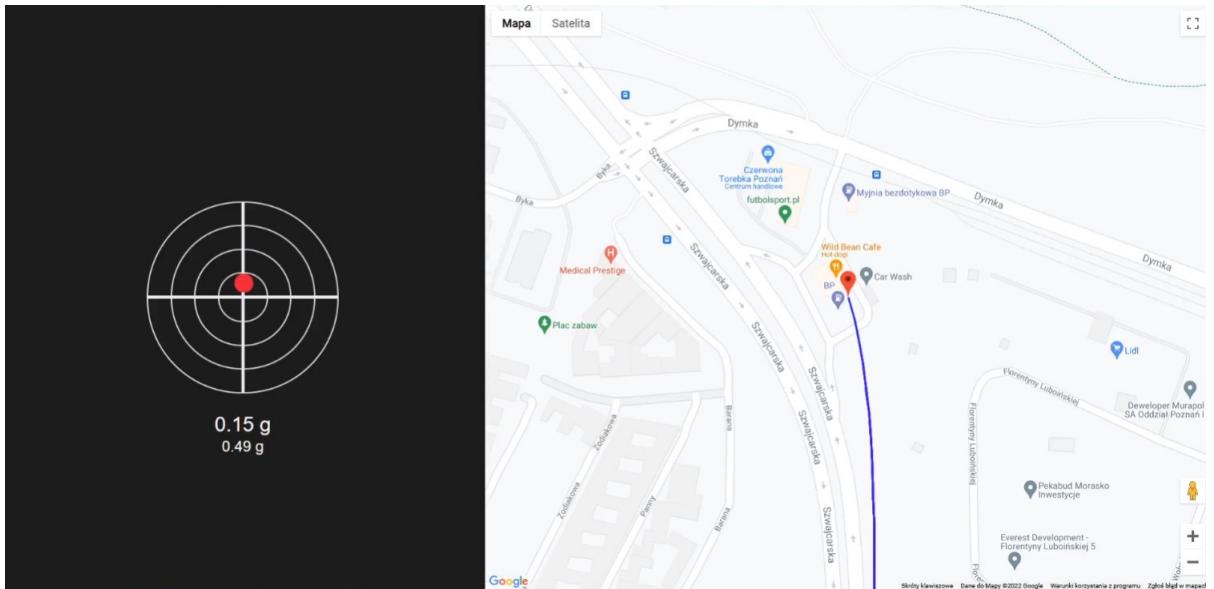
Listing 2: Accelerometr

Kod służący do odczytu danych z akcelerometru, transformacji i wysłaniu ich na stronę.

5 Zdjęcia



6 Zdjęcia aplikacji



7 Podsumowanie i wnioski

Projekt okazał się trudniejszy w realizacji niż początkowo zakładaliśmy. Na początku wystąpiły problemy z modułem GPS. Część ta czasami nie włączała się całkowicie albo przesyłała nic nieznaczące ciągi znaków. Problemy te udało się rozwiązać.

Po pierwszych testach okazało się, że zakupiony akcelerometr jest czulszy na ruchy niż zakładaliśmy, dlatego tym ważniejsze było jego prawidłowe unieruchomienie w obudowie. Urządzenie musi leżeć nieruchomo i prostopadle do podłogi pojazdu.

Komponenty urządzenia – Raspberry Pi, GPS oraz akcelerometr – są zamknięte w kartonowej obudowie. Urządzenie wymaga stałego źródła zasilania. Aby je dostarczyć, można skorzystać z gniazda samochodowego lub złącza USB (na przykład w laptopie czy power banku).

Aplikacja – z powodu tego, że ma być wyświetlana w trakcie jazdy – jest przezroczysta, a wyświetlane elementy są duże.

Projekt udało się zrealizować. Aplikacja wyświetla prawidłowe wartości przeciżenia, a rysowana na mapie trasa odpowiada tej przejechanej przez pojazd. Użytkownik ma możliwość wyświetlenia aplikacji, gdy znajduje się w

tej samej sieci co projektowane urządzenie.

Projekt można by rozszerzyć o funkcję wyznaczania trasy, która następnie byłaby wyświetlana na mapie, a kierowca mógłby wzdłuż niej podążać. Innym pomysłem wartym rozważenia jest dodanie w aplikacji przycisku umożliwiającego wykonanie kalibracji akcelerometru.