# Dokumentacja

# Dekodowanie tekstu z bitów (RDS)

## 1. Informacje wstępne.

Głównym zadaniem projektu było dekodowanie i odczytywanie tekstu z bitów systemu RDS – (wybrana część serwisów). RDS jest standardem protokołu komunikacyjnego wykorzystywanego w radiofonii VHF/FM, nadającej w zakresie 87,5-108 MHz, służącym do przesyłania niedużych ilości danych cyfrowych.

Kod źródłowy aplikacji znajduje się na repozytorium GitHub pod adresem: https://github.com/Evexis/IoT-Project.

## 2. Zastosowane technologie.

Do stworzenia programu wykorzystano język Python. Jako pakiet narzędzi umożliwiający zaawansowane obliczenia matematyczne został użyty moduł NumPy.

## 3. Dekodowane serwisy.

Informacjami dekodowanymi i wyświetlanymi dla użytkownika przez program są:

*Programme Service* Name *(PSN)* Nazwa programu: Przekazuje użytkownikowi informację o nazwie programu.

*Traffic Programme (TP) flag:* Flaga służąca do identyfikacji programów, które od czasu do czasu przenoszą ogłoszenia przeznaczone dla kierowców.

*Traffic Announcement (TA) Signal:* Jest to sygnał pozwalający wskazać chwile w których nadawane są wiadomości głosowe przeznaczone dla kierowców. Sygnał ten jest używany w odbiornikach radiowych do automatycznego przełączania.

*Music/Speech (MS) switch:* Dostarcza informacje czy aktualnie nadawana jest mowa czy muzyka.

*RadioText (RT):* Umożliwia przesłanie tekstu składającego się z 32 lub 64 znaków, pokazywanego na wyświetlaczu odbiornika.

*Data (Day, Month, Year, Hour, Minutes, LocalTimeOffset)* - Przekazuje użytkownikowi informację o dacie.

```
Group 2A detected:
Radio Text1 : Tel. 22 645 9804  Fax.22 843 3732 e-mail:dwojka@polskieradio.pl
Radio Text2 :  *  Program II  *   al. Niepodleglosci 77/85   00-977  Warszawa


Group 0A detected:
PSN: Dwojka
TATP: This programme carries traffic announcements but none are being broadcast at present and may also carry EON information About other traffic announcements
MS: Music
```

```
Group 0A detected:
PSN: Dwojka
TATP: This programme carries traffic announcements but none are being broadcast at present and may also carry EON information About other traffic announ
MS: Music


Date: 18/4/2016  21:57 +0.0
```

# 4. Działanie programu.

Główna pętla programu odpowiedzialna jest za synchronizację grup i bloków w odbiorniku. Kluczowym elementem algorytmu jest ciągłe obliczenie syndromu dla kolejnych porcji bitów, i porównywanie ich ze znanymi wzorcami. Każda analizowana grupa 104 bitów jest złożona z 4 bloków 26 bitowych, ustawionych w odpowiedniej kolejności tzn. A, B, C lub C', D.

```
for i in range(0, bits_length + 2):
    data[i] = int(float(data[i]))
# index - start of block A
while index < bits_length:
    # obliczamy syndromy kazdego z bloku
    syndrome_result = syndrome(index, data, check)
    if numpy.array_equal(syndrome_result, syndromA):

        syndrome_result = syndrome(index + 26, data, check)
        if numpy.array_equal(syndrome_result, syndromB):
            syndrome_result = syndrome(index + 52, data, check)

            if numpy.array_equal(syndrome_result, syndromCa):
                syndrome_result = syndrome(index + 78, data, check)

                if numpy.array_equal(syndrome_result, syndromD):
                    # inkrementacja licznika petli o dlugsc grupy -1
                    analise(index, data, PSN, text1, text2, TA, TP, MS, Day, Month, Year,Hour,Minutes,LocalTimeOffset)
                    index = index + 103

                    detected_block_counter = detected_block_counter + 1
                    # cała grupa znaleziona
            elif numpy.array_equal(syndrome_result, syndromCb):

                syndrome_result = syndrome(index + 78, data, check)
                if numpy.array_equal(syndrome_result, syndromD):

                    analise(index, data, PSN, text1, text2, TA, TP, MS, Day, Month, Year,Hour,Minutes,LocalTimeOffset)

                    index = index + 103
                    # index increament 104-1
                    detected_block_counter = detected_block_counter + 1

    index = index + 1
```

Zadaniem funkcji znajdującej się w tym pliku jest identyfikacja oraz analiza, każdej wykrytej grupy. Działanie algorytmu polega na identyfikacji typu grupy oraz odpowiednim zinterpretowaniu zawartości bloków. W pierwszym

kroku algorytmu pobierany jest kod typu grupy, następnie znajdowany jest blok programowy realizujący przetwarzanie dla danego typu grupy.

```
def analise(index, data, PSN, text1, text2, TA, TP, MS, Day, Month, Year,Hour,Minutes,LocalTimeOffset):
    # print("analize")
    # pobieramy bity blokow danych
    blockA = data[index: index + 25]
    blockB = data[index + 26: index + 51]
    blockC = data[index + 52: index + 77]
    blockD = data[index + 78: index + 103]
    groupType = blockB[0:5]
    # print(groupType)
    # znajdujemy blok programowy rzetwarzajacy grupe
    if numpy.array_equal(groupType[0:4], [0, 0, 0, 0]):
        # print("GrupaA");
        TP.data = blockB[5]
        TA.data = blockB[11]
        MS.data = blockB[12]  # Music Speech switch
        array = [0, 0, 0]
        for i in blockB[6:11]:
            array.append(i)



        seg_addr = blockB[14:16]
        # adres segmentu

        ascii_1 = blockD[0:8]
        ascii_2 = blockD[8:16]

        char_1 = vbin2char(ascii_1)  # bin -> char
        char_2 = vbin2char(ascii_2)  # bin -> char

        if numpy.array_equal(seg_addr, [0, 0]):
            s = list(PSN.data)
            s[0] = char_1
            s[1] = char_2
            PSN.data = "".join(s)
        elif numpy.array_equal(seg_addr, [0, 1]):
            s = list(PSN.data)
            s[2] = char_1
            s[3] = char_2
            PSN.data = "".join(s)
    elif numpy.array_equal(seg_addr, [0, 1]):
        s = list(PSN.data)
        s[2] = char_1
        s[3] = char_2
        PSN.data = "".join(s)
        # PSN[2:4]=[char_1,char_2]
    elif numpy.array_equal(seg_addr, [1, 0]):

        s = list(PSN.data)
        s[4] = char_1
        s[5] = char_2
        PSN.data = "".join(s)

    elif numpy.array_equal(seg_addr, [1, 1]):
        s = list(PSN.data)

        s[6] = char_1
        s[7] = char_2
        PSN.data = "".join(s)
```

## 4. Trudności napotkane podczas tworzenia programu.

Teksty są zmieniane po 2 znaki, co wymagało stworzenia najpierw tekstuu składającego się z dowolnych znaków (np: `PSN = Container('xxxxxxxx'),` tak aby  dało się je podmieniac w trakcie otrzymywania kolejnych znaków.

## 10. Bibliografia

[1] Oficjalna strona środowiska Python https://https://www.python.org//
[2] System kontroli wersji Git https://github.com/