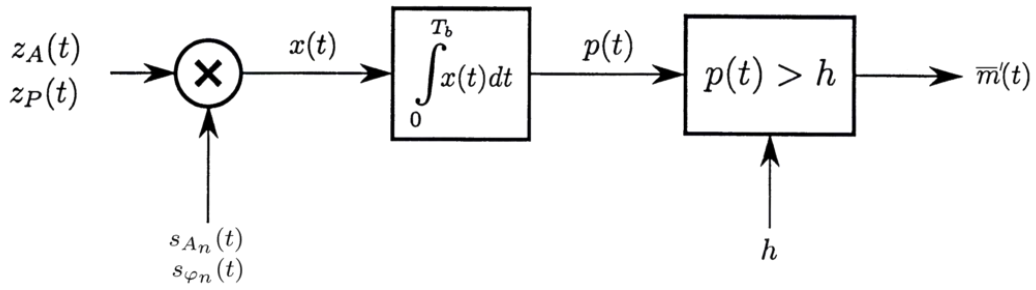
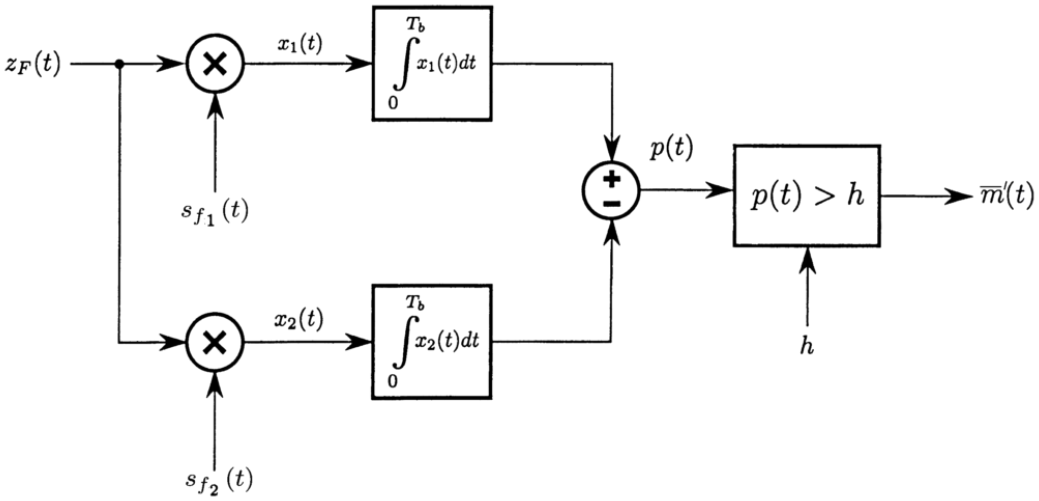
Demodulacja dyskretna

Dla poznanych na poprzednich zajęciach modulacji dyskretnych demodulacja odbywa się następująco:

- dla zA(t) i zP(t) (ASK i PSK) :



- dla zF(t) (FSK):



gdzie m(t) należy do {0, 1} – sygnał zdemodulowany, p(t) – sygnał całkowy, h– próg komparatora.

Wykonaj w formie programistycznej implementacji poniżej przedstawione zadania.

1) Wykorzystaj sygnały zA(t), zP(t) i zF(t) i z poprzednich zajęć laboratoryjnych.

2) Zaimplementuj powyżej przedstawione demodulatory.

3) Na podstawie obserwacji sygnału m(t) dobierz eksperymentalnie wartość progu h. W wyniku porównania z wartością progową na wyjściu komparatora wygenerować odpowiedź postaci:

$$m(t) = \begin{cases} 0 & p(t) < h \\ 1 & p(t) \geq h \end{cases}$$

4) Wygeneruj wykresy przedstawiajace wynik demodulacji dla rozpatrywanych sygnałów modulowanych (sygnały wejściowe: x(t)/x1(t)/x2(t), p(t) oraz sygnał wynikowy m(t)).

Kody dla demodulacji ASK

Kod do obliczenia x(t):

```
public ChartDetails multiplyASK(String bits, List<Double> list) {
    List<Double> xlist = makeXList(bits);
    List<Double> scores = new ArrayList<>();
    boolean loop = true;

    for (int i = 0; i < xlist.size() && loop; i++) {
        double tmp = list.get(i) * a2 * Math.sin(fi + 2 * Math.PI * f * xlist.get(i));
        scores.add(tmp);
    }
    return new ChartDetails( title: "Demodulacja ASK - x(t)", scores, xAxisTitle: "t[s]", yAxisTitle: "x[t]");
}
```

Kod do obliczenia p(t):

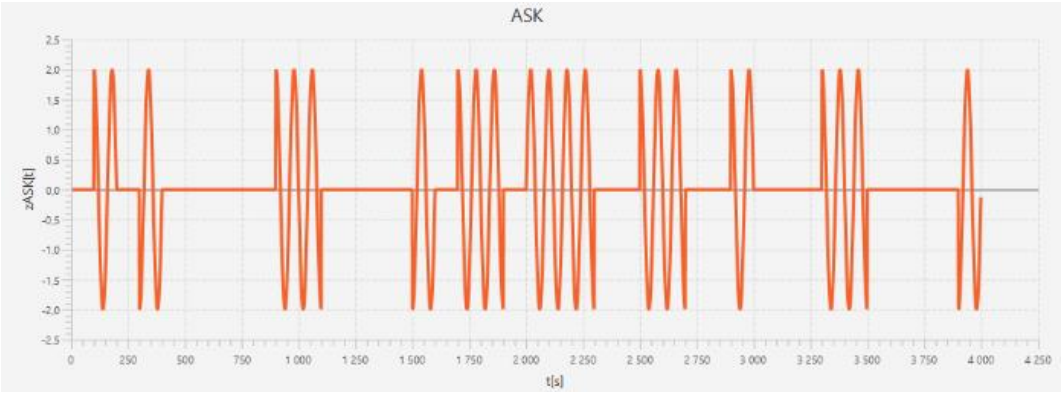
```
public ChartDetails integralASK(String bits, List<Double> list) {
    List<Double> scores = new ArrayList<>();
    List<Double> xlist = makeXList(bits);
    int bitsLength = list.size() / bits.length();
    int bitNumber = 0;
    double sum = 0;
    for (int i = 0; i < list.size(); i++) {
        if (i < xlist.size() - 1) {
            sum += list.get(i) * (xlist.get(i + 1) - xlist.get(i));
        }

        if (bitNumber != i / bitsLength) {
            sum = 0;
            bitNumber++;
        }
        scores.add(sum * 1000);

    }
    return new ChartDetails( title: "Demodulacja ASK - p(t)", scores, xAxisTitle: "t[s]", yAxisTitle: "p[t]");
}
```

Kod do obliczenia m(t):

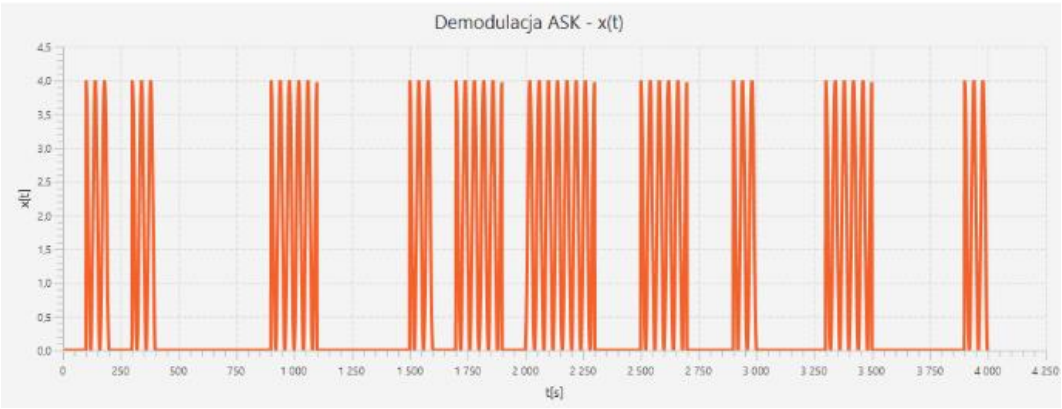
```
public ChartDetails gateASK(List<Double> list) {
    List<Double> scores = new ArrayList<>();

    for (Double e : list) {
        if (e < h) {
            scores.add(0.0);
        } else {
            scores.add(1.0);
        }
    }

    return new ChartDetails( title: "Demodulacja ASK - m(t) ", scores, xAxisTitle: "t[s]", yAxisTitle: "m[t]");
}
```

Wykresy dla ASK i demodulacji ASK:

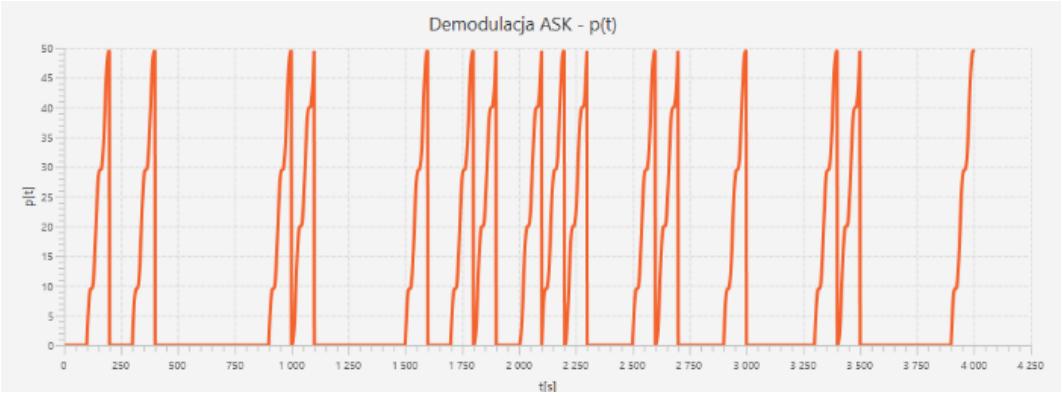
Wykres ASK



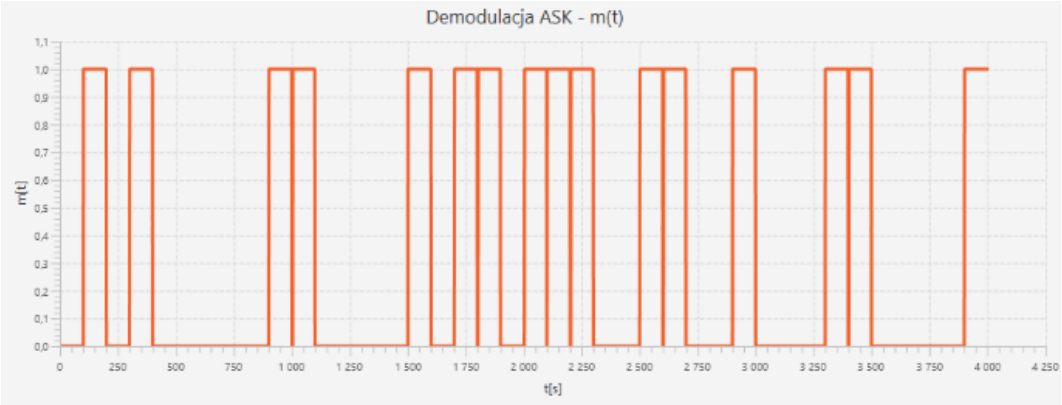
Wykres x(t):



Wykres p(t):



Wykres m(t):



Kody do demodulacji PSK:

Kod do obliczenia x(t):

```
public ChartDetails multiplyPSK(String bits, List<Double> list) {
    List<Double> xlist = makeXList(bits);
    List<Double> scores = new ArrayList<>();
    boolean loop = true;

    for (int i = 0; i < xlist.size() && loop; i++) {
        double tmp = list.get(i) * a * Math.sin(fi2 + 2 * Math.PI * f * xlist.get(i));
        scores.add(tmp);
    }
    return new ChartDetails( title: "Demodulacja PSK - x(t)", scores, xAxisTitle: "t[s]", yAxisTitle: "x[t]");
}
```

Kod do obliczenia p(t):

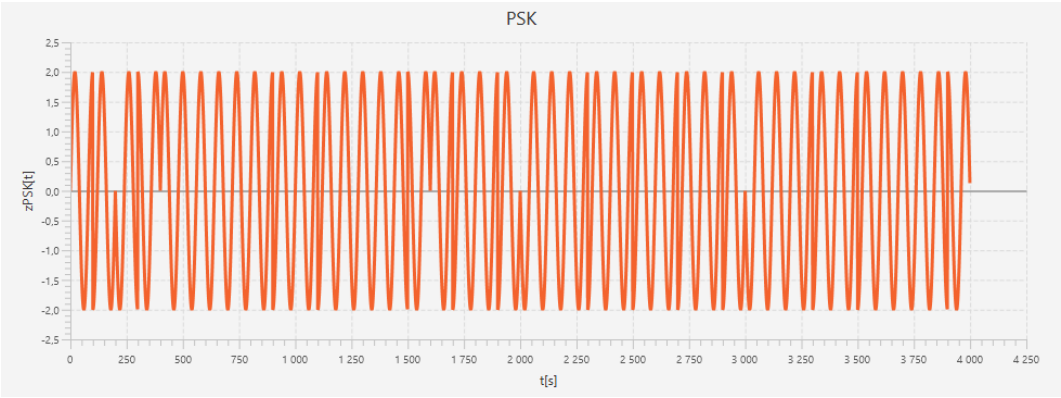
```
public ChartDetails integralPSK(String bits, List<Double> list) {
    List<Double> scores = new ArrayList<>();
    List<Double> xlist = makeXList(bits);
    int bitsLength = list.size() / bits.length();
    int bitNumber = 0;
    double sum = 0;
    for (int i = 0; i < list.size(); i++) {
        if (i < xlist.size() - 1) {
            sum += list.get(i) * (xlist.get(i + 1) - xlist.get(i));
        }

        if (bitNumber != i / bitsLength) {
            sum = 0;
            bitNumber++;
        }
        scores.add(sum * 1000);

    }
    return new ChartDetails( title: "Demodulacja PSK - p(t)", scores, xAxisTitle: "t[s]", yAxisTitle: "p[t]");
}
```

Kod do obliczenia m(t):

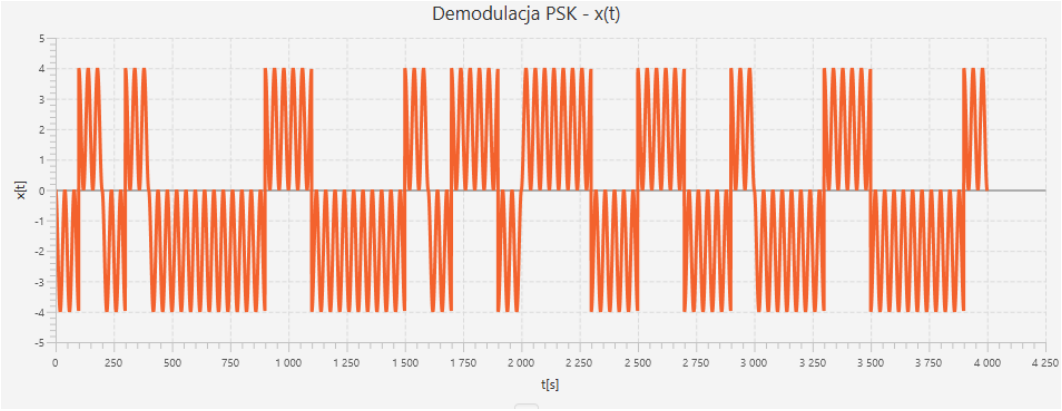
```
public ChartDetails gatePSK(List<Double> list) {
    List<Double> scores = new ArrayList<>();

    for (Double e : list) {
        if (e < h) {
            scores.add(0.0);
        } else {
            scores.add(1.0);
        }
    }

    return new ChartDetails( title: "Demodulacja PSK - m(t)", scores, xAxisTitle: "t[s]", yAxisTitle: "m[t]");
}
```

Wykresy dla PSK i demodulacji PSK:

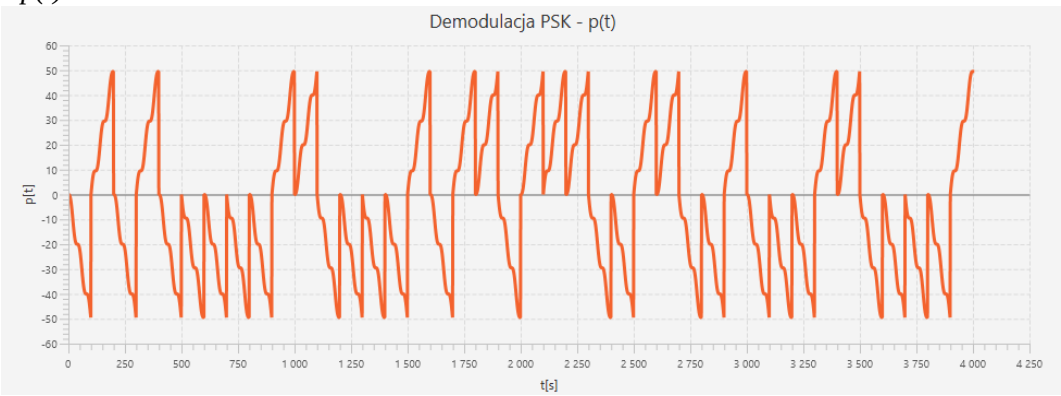
Wykres PSK



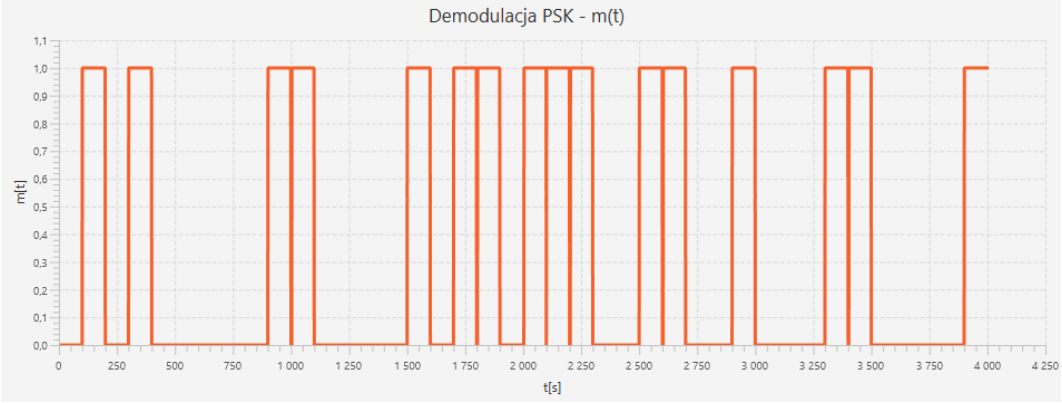
Wykres x(t):



Wykres p(t):



Wykres m(t):



Kody do demodulacji FSK:

Kod do obliczenia x1(t):

```
public ChartDetails multiplyFSK1(String bits, List<Double> list) {
    List<Double> xlist = makeXList(bits);
    List<Double> scores = new ArrayList<>();
    boolean loop = true;

    for (int i = 0; i < xlist.size() && loop; i++) {
        double tmp = list.get(i) * a * Math.sin(fi + 2 * Math.PI * f1 * xlist.get(i));
        scores.add(tmp);
    }
    return new ChartDetails( title: "Demodulacja FSK - x1(t)", scores, xAxisTitle: "t[s]", yAxisTitle: "x1[t]");
}
```

Kod do obliczenia x2(t):

```
public ChartDetails multiplyFSK2(String bits, List<Double> list) {
    List<Double> xlist = makeXList(bits);
    List<Double> scores = new ArrayList<>();
    boolean loop = true;

    for (int i = 0; i < xlist.size() && loop; i++) {
        double tmp = list.get(i) * a * Math.sin(fi + 2 * Math.PI * f2 * xlist.get(i));
        scores.add(tmp);
    }
    return new ChartDetails( title: "Demodulacja FSK - x2(t)", scores, xAxisTitle: "t[s]", yAxisTitle: "x2[t]");
}
```

Kod do obliczenia p1(t):

```
public ChartDetails integralFSK1(String bits, List<Double> list) {
    List<Double> scores = new ArrayList<>();
    List<Double> xlist = makeXList(bits);
    int bitsLength = list.size() / bits.length();
    int bitNumber = 0;
    double sum = 0;
    for (int i = 0; i < list.size(); i++) {
        if (i < xlist.size() - 1) {
            sum += list.get(i) * (xlist.get(i + 1) - xlist.get(i));
        }

        if (bitNumber != i / bitsLength) {
            sum = 0;
            bitNumber++;
        }
        scores.add(sum * 1000);

    }
    return new ChartDetails( title: "Demodulacja FSK - p1(t)", scores, xAxisTitle: "t[s]", yAxisTitle: "p1[t]");
}
```

Kod do obliczenia p2(t):

```
public ChartDetails integralFSK2(String bits, List<Double> list) {
    List<Double> scores = new ArrayList<>();
    List<Double> xlist = makeXList(bits);
    int bitsLength = list.size() / bits.length();
    int bitNumber = 0;
    double sum = 0;
    for (int i = 0; i < list.size(); i++) {
        if (i < xlist.size() - 1) {
            sum += list.get(i) * (xlist.get(i + 1) - xlist.get(i));
        }

        if (bitNumber != i / bitsLength) {
            sum = 0;
            bitNumber++;
        }
        scores.add(sum * 1000);

    }
    return new ChartDetails( title: "Demodulacja FSK - p2(2)", scores, xAxisTitle: "t[s]", yAxisTitle: "p2[t]");
}
```

Kod do obliczenia p(t):

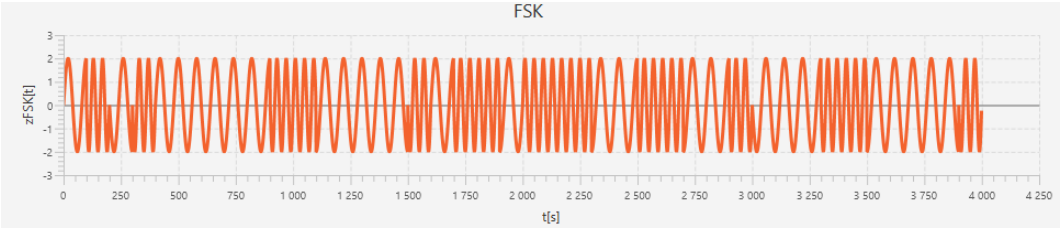
```
public ChartDetails result(List<Double> list1, List<Double> list2) {
    List<Double> scores = new ArrayList<>();
    for (int i = 0; i < list1.size() && i < list2.size(); i++) {
        scores.add(list1.get(i) - list2.get(i));
    }
    return new ChartDetails( title: "Demodulacja FSK - p(t)", scores, xAxisTitle: "t[s]", yAxisTitle: "p[t]");
}
```

Kod do obliczenia m(t):

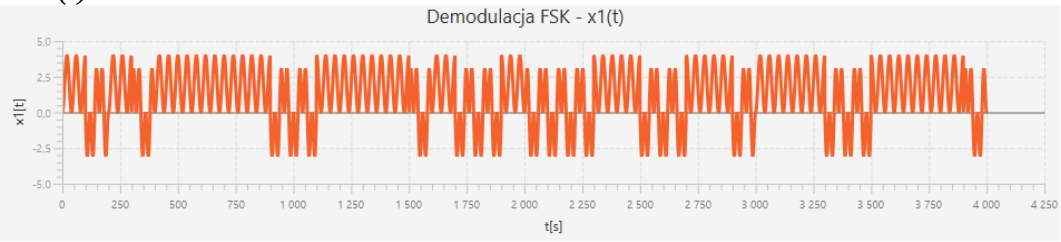
```
public ChartDetails gateFSK(List<Double> list) {
    List<Double> scores = new ArrayList<>();

    for (Double e : list) {
        if (e < h) {
            scores.add(0.0);
        } else {
            scores.add(1.0);
        }
    }

    return new ChartDetails( title: "Demodulacja FSK - m(t)", scores, xAxisTitle: "t[s]", yAxisTitle: "m[t]");
}
```

Wykresy dla PSK i demodulacji PSK:

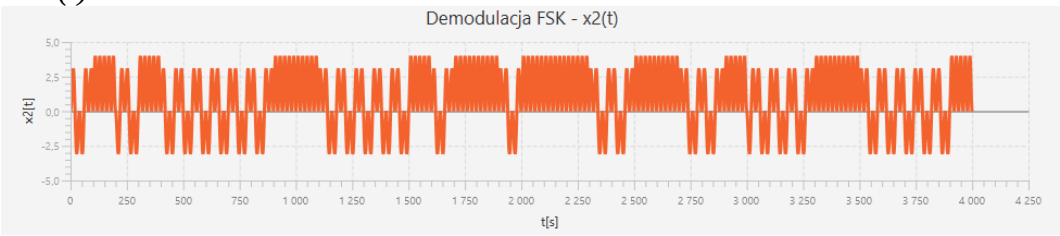
Wykres PSK



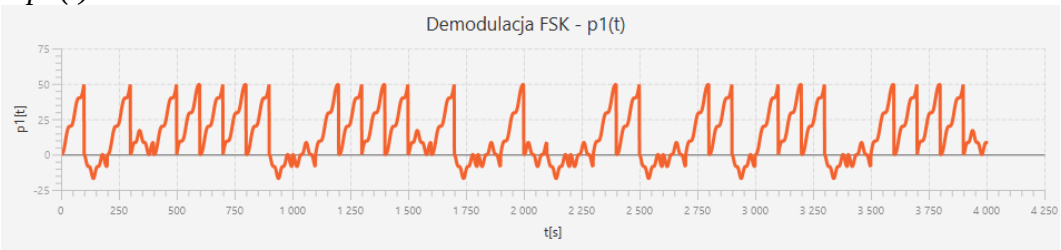
Wykres x1(t)



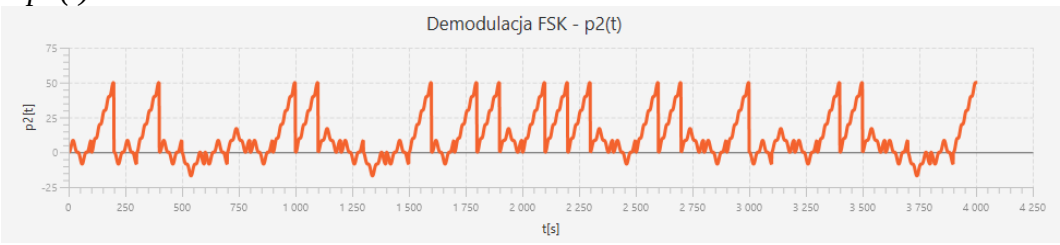
Wykres x2(t)



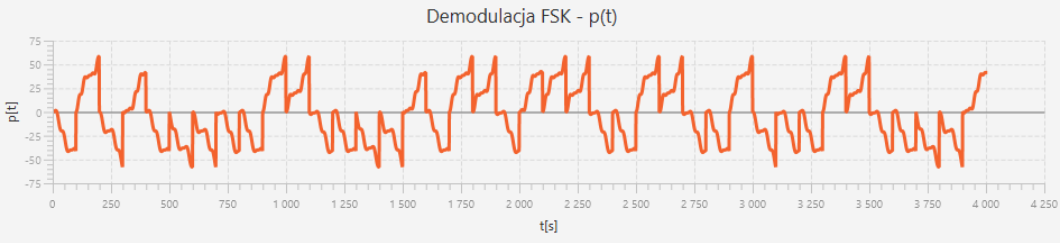
Wykres p1(t)



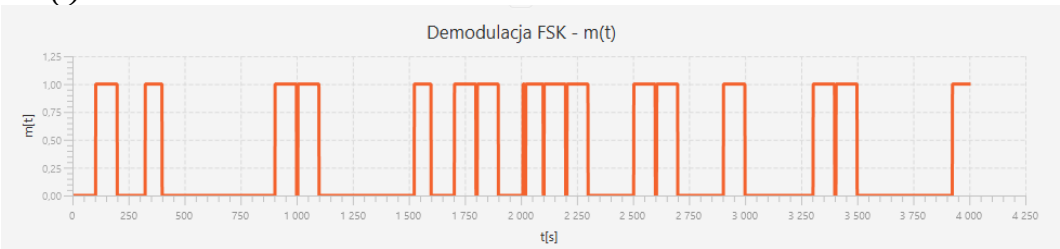
Wykres p2(t)



Wykres p(t)



Wykres m(t)


Demodulacja FSK - m(t)
m[t]
1,25
1,00
0,75
0,50
0,25
0,00
0
250
500
750
1 000
1 250
1 500
1 750
2 000
2 250
2 500
2 750
3 000
3 250
3 500
3 750
4 000
4 250
t[s]