Kodowanie transmisyjne – kodowanie Menchester

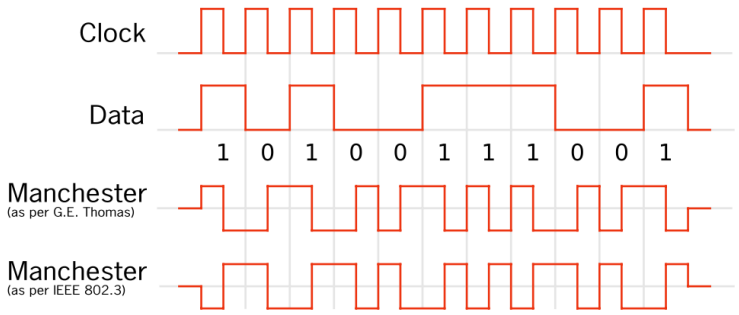
Kodowanie TTL unipolarne RZ (Return to Zero)

Praktycznie używane w transmisjach przewodowych przy małych odległościach. Kodowanie mało odporne na zakłócenia. Wymaga dobrej synchronizacji.
Sposób kodowania:
Wraz z początkiem cyklu zegara każdej wartości bitu przy przyporządkowywana jest odpowiednia wartość napięcia trwająca przez dany cykl.

- „1” binarna jest zamieniana na stan wysoki
- „0” binarne jest zamieniane na stan niski

Dekodowanie odbywa się wprost z odczytanej wartości.

Kodowanie Manchester



Wykonaj w formie programistycznej implementacji poniżej przedstawione zadania.

1) Napisz funkcję generującą sygnał zegarowy, będący sygnałem prostokątnym o zadanej częstotliwości.

```
public ChartDetails clk(double fregency, double start, double stop, int stepsValue) {
    List<Double> yValues = new ArrayList<>();
    double size = start - stop;
    if (size == 0) {
        size = 1;
    }
    double step = size / stepsValue;
    double bitStep = (1 / fregency) / 2;  //długość jednego bitu

    for (int i = 0; i < stepsValue; i++) {
        xValues.add(i * step);
        if (((int) (i * step / bitStep)) % 2 == 0) {
            yValues.add(1.0);
        } else {
            yValues.add(0.0);
        }
    }
    return new ChartDetails( title: "CLK", yValues, xAxisTitle: "", yAxisTitle: "");
}
}
```

2) Jako generatora TTL użyj kodu generującego sygnał informacyjny m(t)z tematu laboratoryjnego „5. Modulacja dyskretna”. Wykorzystaj do wygenerowania sygnału m(t) dwa bajty.

```
public ChartDetails stringToBinaryStream(String text, Boolean isBigEndian) {
    byte[] bytes = text.getBytes();
    List<Double> list = new ArrayList<>();
    StringBuilder bits = new StringBuilder();
    System.err.println(text + " contains " + bytes.length + " bytes");
    for (byte b : bytes) {
        int val = b;
        for (int i = 0; i < 8; i++) {
            bits.append((val & 128) == 0 ? 0 : 1);
            val <<= 1;
        }
    }
    for (int bit = 0; bit < bits.length(); bit++) {
        for (int sample = 0; sample < tb; sample++)
            list.add(Double.parseDouble(String.valueOf(bits.charAt(bit))));
    }
    if (isBigEndian)
        Collections.reverse(list);
    System.err.println(bits.toString());
    return new ChartDetails( title: "TTL", list, xAxisTitle: "t[s]", yAxisTitle: "TTL");
}
```

3) Zgodnie z regułami przedstawionymi w skrócie z teorii napisz funkcje/programy generujące przebiegi sygnału Manchaster.

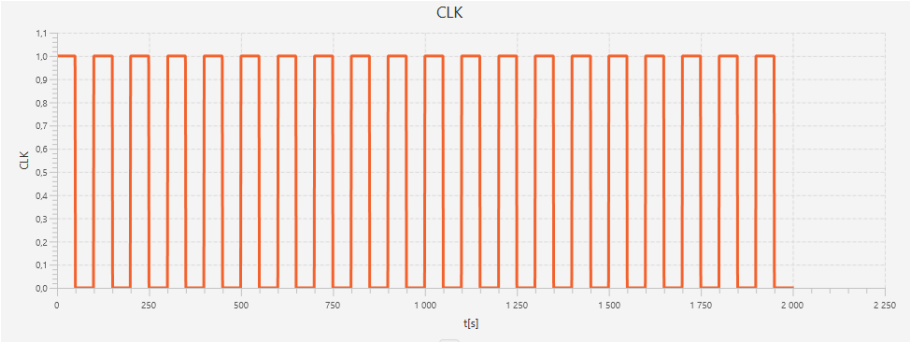
```
public ChartDetails menchester(List<Double> clock, String bits) {
    List<Double> scores = new ArrayList<>();
    boolean up = false;
    boolean down = false;
    char prevBit = '0';
    double prevClock = 1.0;
    double value = 0;

    for (int i = 0, j = 0; i < clock.size(); i++) {
        if (j < bits.length()) {
            if (prevClock == 0 && clock.get(i) == 1) {
                up = true;
                j++;
            }
            if (prevClock == 1 && clock.get(i) == 0) {
                down = true;
            }

            if (down) {
                if (bits.charAt(j) == '0') {
                    value = 1.0;
                } else {
                    value = -1.0;
                }
            }

            if (up && j < bits.length()) {
                if (j != 0 && bits.charAt(j) == prevBit) { //if value = 1 -> -1 else 1
                    value = (value == 1) ? -1 : 1;
                }
                prevBit = bits.charAt(j);
            }

            up = false;
            down = false;
            prevClock = clock.get(i);
            scores.add(value);
        }
    }
    return new ChartDetails( title: "Menchester", scores, xAxisTitle: "", yAxisTitle: "");
}
```

4) Napisz dekodery dla kodu Manchaster.
Przetestuj poprawność ich działania.

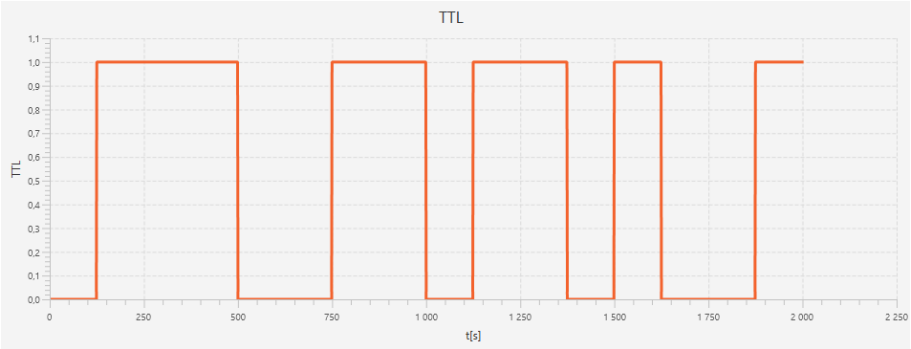
```java
public StringBuilder decMenchester(List<Double> menchester, int size) {
    StringBuilder bits = new StringBuilder();
    int bitsLength = menchester.size() / (size * 2);
    int start = bitsLength / 2;
    double prevBit = 0;
    double tmp = 0;

    for (int i = start, j = 0; i < menchester.size(); i += bitsLength, j++) {
        if (j % 2 != 0) {
            if (prevBit == 1 && menchester.get(i) == -1) {
                bits.append(1);
            } else {
                bits.append(0);
            }
        }
        prevBit = menchester.get(i);
    }
    return bits;
}
```

Wykresy:

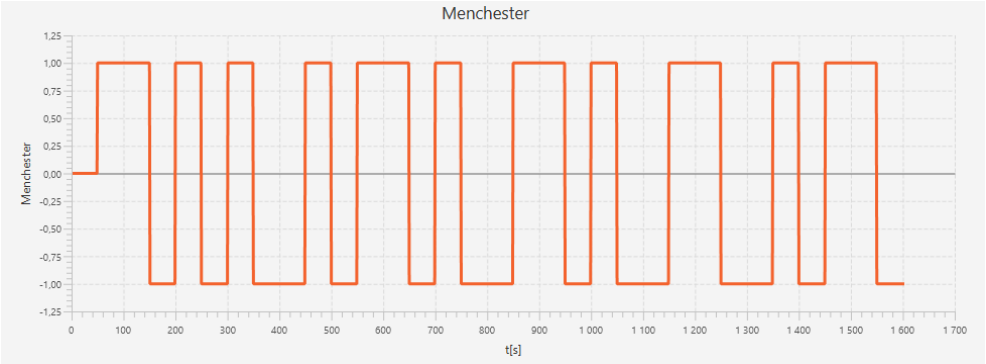
Wykres CLK:



Wykres TTL:



Wykres Menchester:



Wynik dekodowania:

```
si contains 2 bytes
0111001101101001
Wynik dekodowania:
0111001101101001
```