09.05.2020

Sprawozdanie z LAB 07

Zadaniem zajęć było wykonanie cyfrowej demodulacji amplitudy, częstotliwości i fazy dla funkcji bazowej.

Oto kod źródłowy (modulatory i demodulatory):

```
#define _USE_MATH_DEFINES
//#define _PIXEL_SIZE_X "7680"
//#define _PIXEL_SIZE_X "4320"
#define _PIXEL_SIZE_X "3840"
#define _PIXEL_SIZE_X "2160"
#include <sstream>
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <cstddef>
#include <bitset>
#include <iostream>
#include "dft_algorithm.h"
#include "my_plot.h"

#define ARG1 " with linespoints pointtype 6 pointsize 1 lc rgb 'red' "
#define ARG2 " with linespoints lc rgb 'red'  "
#define ARG3 " with linespoints pointtype 6 pointsize 0.3 lc rgb 'black' "

enum Endian
{
	littleEndian = 1,
	bigEndian = 2
};

// STRUMIEN BINARNY = 011000010110001001100011
std::string S2BS(char in[], Endian sw = littleEndian)
{

	int N = strlen(in) - 1;
	std::ostringstream str;
	std::string wynik;

	if (sw == littleEndian)
	{
		for (int i = 0; i <= N; i++)
		{
			std::bitset<8> x(in[i]);
```

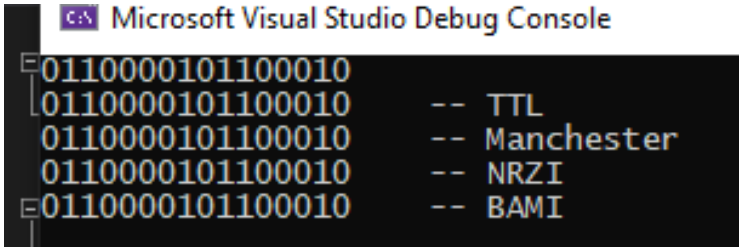
```
                        str << x;
                }
        }
        else
        {
                for (int i = N; i >= 0; i++)
                {
                        std::bitset<8> x(in[i]);
                        str << x;
                }
        }

        str << std::endl;
        wynik = str.str();

        return wynik;
};

namespace demodualtor
{
        std::string TTL(std::vector<std::pair<double, double>> const &input, double Tb = 1.0)
        {
                double Ts = input.at(1).first - input.at(0).first;
                std::string wynik;
                std::vector<std::pair<double, double>>::const_iterator it = input.begin();

                while((std::distance(it, input.end()) >= floor(Tb / Ts)))
                {
                        if (it->second < 0.5) wynik.append("0");
                        else wynik.append("1");

                        std::advance(it, floor(Tb / Ts));
                }

                return wynik;
        }

        std::string Manchester(std::vector<std::pair<double, double>> const& input, double Tb = 1.0)
        {
                double Ts = input.at(1).first - input.at(0).first;
                std::string wynik;
                std::vector<std::pair<double, double>>::const_iterator it = input.begin(),it2 = input.begin();
                std::advance(it, floor((Tb / Ts) * 0.4));
                std::advance(it2, floor((Tb / Ts) * 0.6));

                while ((std::distance(it2, input.end()) >= floor(Tb / Ts)))
                {
                        if (it->second < 0.0 && it2->second > 0.0) wynik.append("0");
```

```
                else if (it->second > 0.0 && it2->second < 0.0) wynik.append("1");

                std::advance(it, floor(Tb / Ts));
                std::advance(it2, floor(Tb / Ts));
        }
        if (it->second < 0.0 && it2->second > 0.0) wynik.append("0");
        else if (it->second > 0.0 && it2->second < 0.0) wynik.append("1");
        return wynik;
}

std::string NRZI(std::vector<std::pair<double, double>> const& input, double Tb = 1.0)
{
        double Ts = input.at(1).first - input.at(0).first;
        std::string wynik;
        std::vector<std::pair<double, double>>::const_iterator it = input.begin(), it2 = input.begin();
        std::advance(it2, 10);
        if ((it->second < 0.0 && it2->second > 0.0) || (it->second > 0.0 && it2->second < 0.0)) wynik.append("1");
        else wynik.append("0");

        std::advance(it, floor((Tb / Ts)*0.8));
        std::advance(it2, floor(Tb / Ts));

        while ((std::distance(it2, input.end()) >= floor(Tb / Ts)))
        {
                if ((it->second < 0.0 && it2->second > 0.0) || (it->second > 0.0 && it2->second < 0.0)) wynik.append("1");
                else wynik.append("0");

                std::advance(it, floor(Tb / Ts));
                std::advance(it2, floor(Tb / Ts));
        }
        if ((it->second < 0.0 && it2->second > 0.0) || (it->second > 0.0 && it2->second < 0.0)) wynik.append("1");
        else wynik.append("0");
        return wynik;

}

std::string BAMI(std::vector<std::pair<double, double>> const& input, double Tb = 1.0)
{
        double Ts = input.at(1).first - input.at(0).first;
        std::string wynik;
        std::vector<std::pair<double, double>>::const_iterator it = input.begin(), it2 = input.begin();
        std::advance(it2, 10);
        if ((it->second < 0.5 && it2->second > 0.5) || (it->second > -0.5 && it2->second < -0.5)) wynik.append("1");
        else wynik.append("0");

        std::advance(it, floor((Tb / Ts) * 0.8));
        std::advance(it2, floor(Tb / Ts));
```

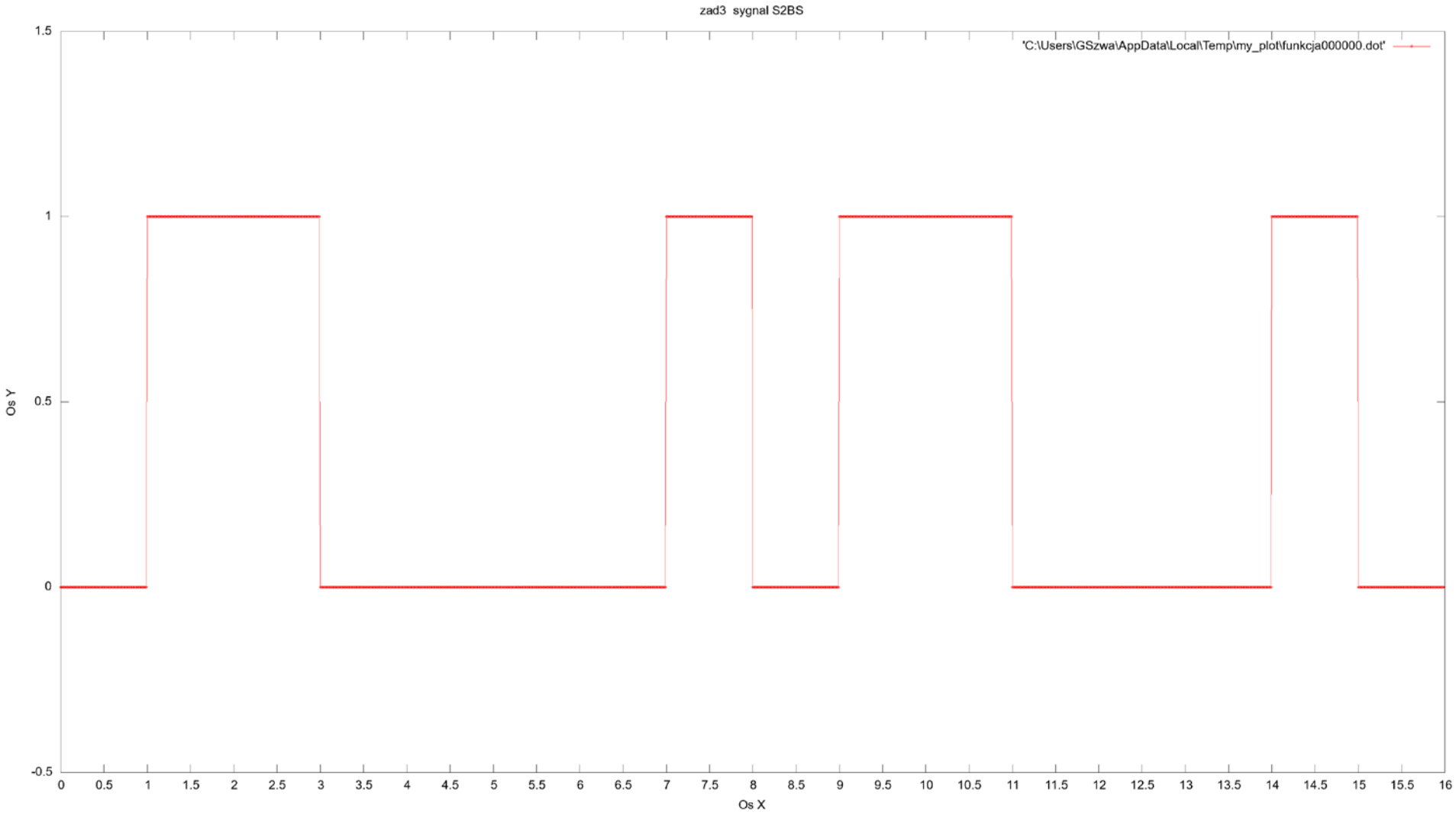
```
		while ((std::distance(it2, input.end()) >= floor(Tb / Ts)))
		{
			if ((it->second < 0.5 && it2->second > 0.5) || (it->second > -0.5 && it2->second < -0.5)) wynik.append("1");
			else wynik.append("0");

			std::advance(it, floor(Tb / Ts));
			std::advance(it2, floor(Tb / Ts));
		}
		if ((it->second < 0.5 && it2->second > 0.5) || (it->second > -0.5 && it2->second < -0.5)) wynik.append("1");
		else wynik.append("0");
		return wynik;
	}
}

namespace modulators
{
	double zegar(double const& t, int const& data)
	{
		return data;
	}
	double const Tb = 1.0;

	double TTL(double const& t, int const& data)
	{
		return data;
	}

	double Manchester(double const& t, int const& data)
	{
		if (data == 0)
		{
			if (t < (Tb / 2.0))
			{
				return -1;
			}
			else
			{
				return 1;
			}
		}
		else
		{
			if (t >= (Tb / 2.0))
			{
				return -1;
			}
			else
			{
```

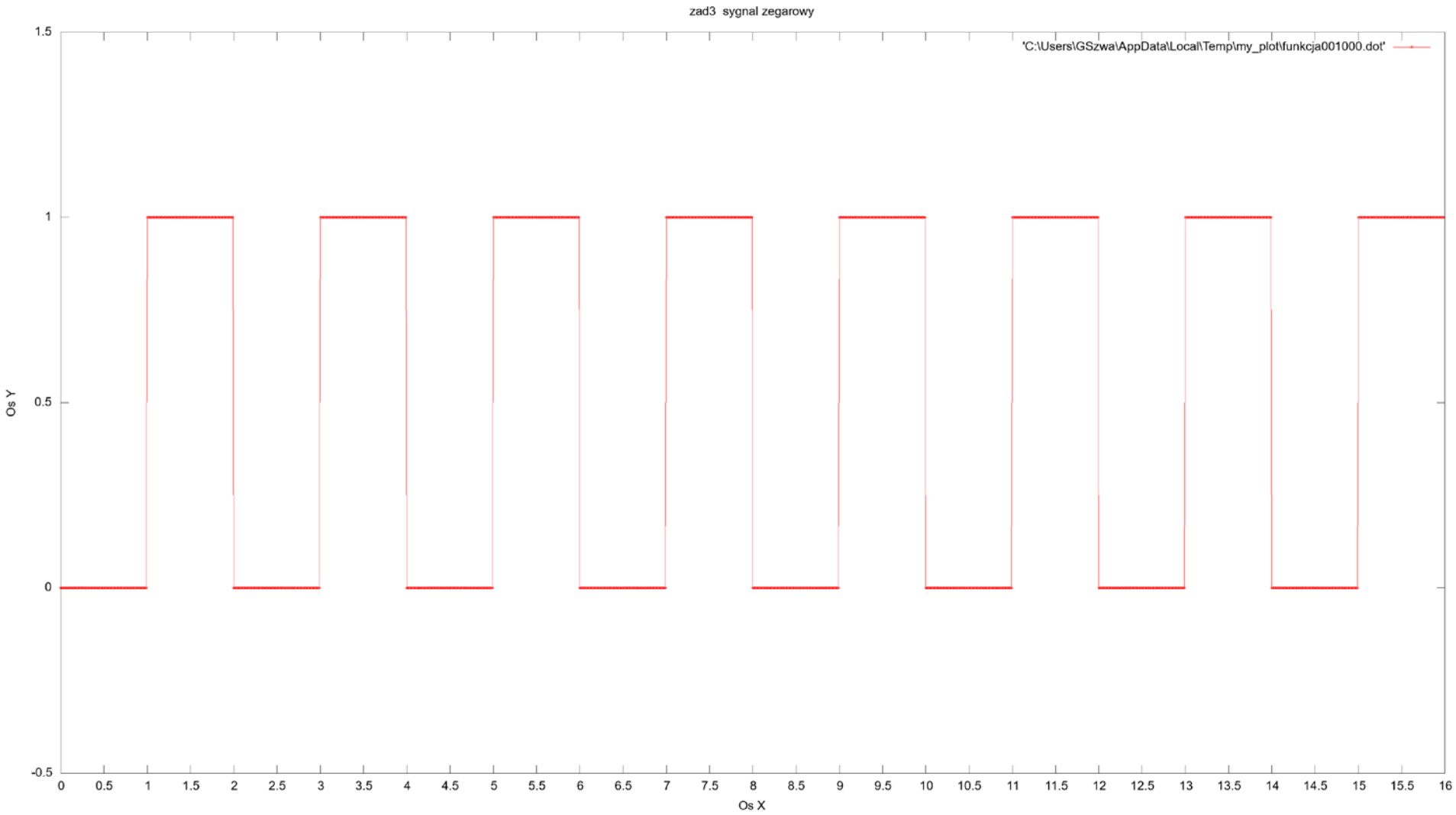
```
                        return 1;
                }
        }
}

int NRZI_back = 1;
double NRZI(double const& t, int const& data)
{
        if (data == 1 && t < 0.005)
        {
                if (NRZI_back == 1) NRZI_back = -1;
                else NRZI_back = 1;
        }
        return NRZI_back;

}

int BAMI_back = 0;
double BAMI(double const& t, int const& data)
{
        if (data == 0.0) return 0;

        if (data == 1 && t < 0.005)
        {
                if (BAMI_back == 1) BAMI_back = -1;
                else BAMI_back = 1;
        }
        return BAMI_back;

}
}
```

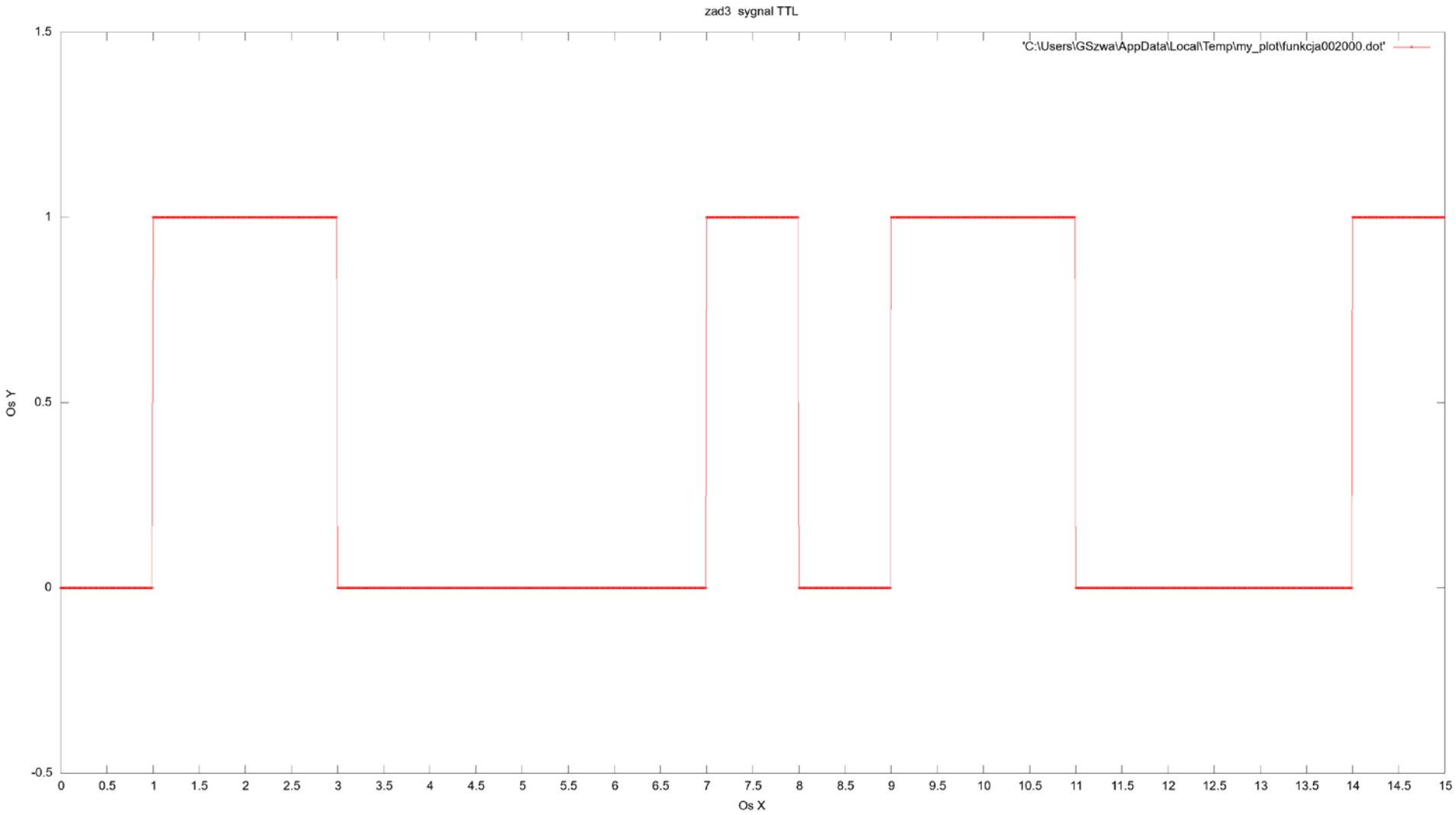
Wynik demodulacji:

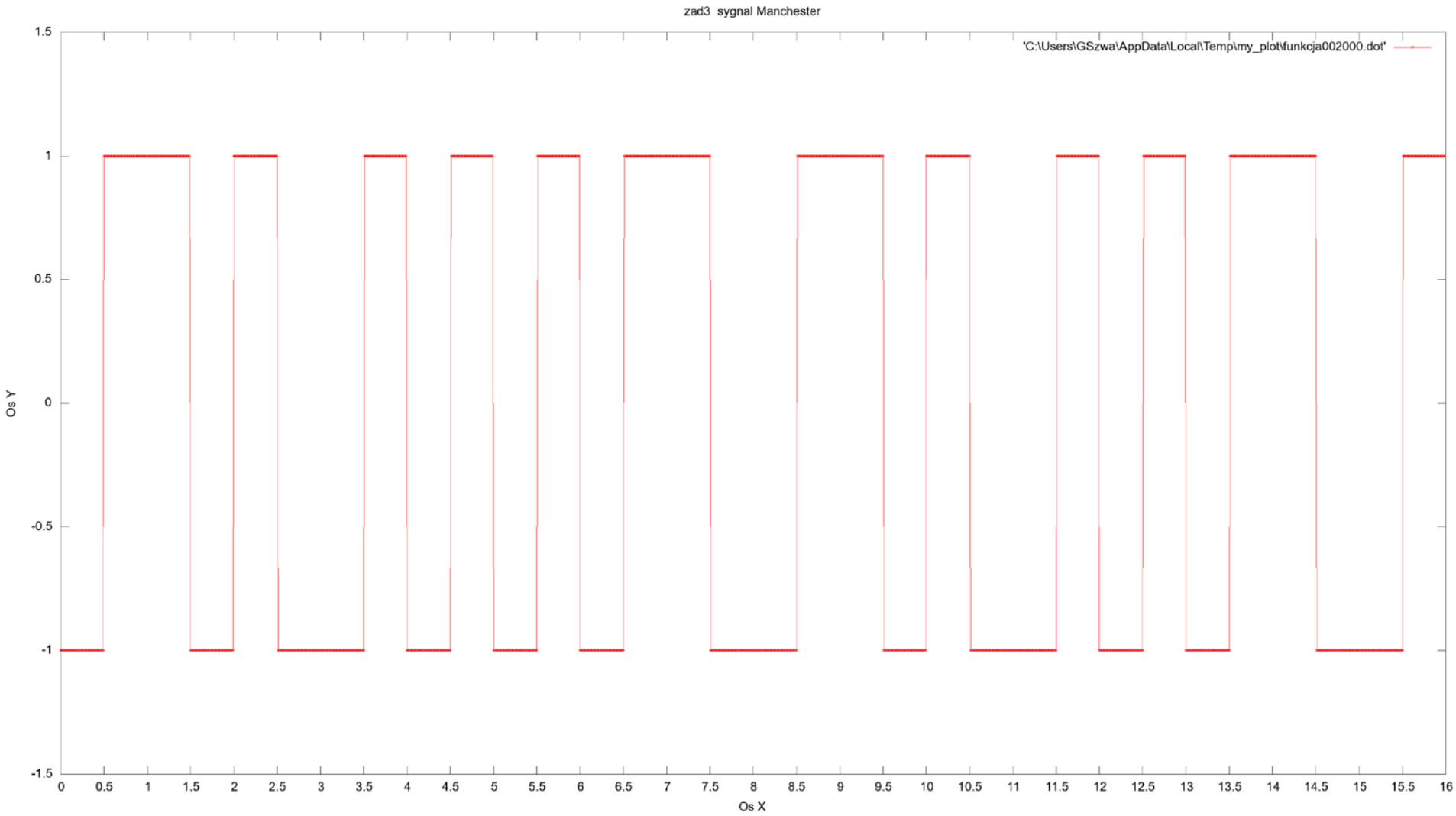


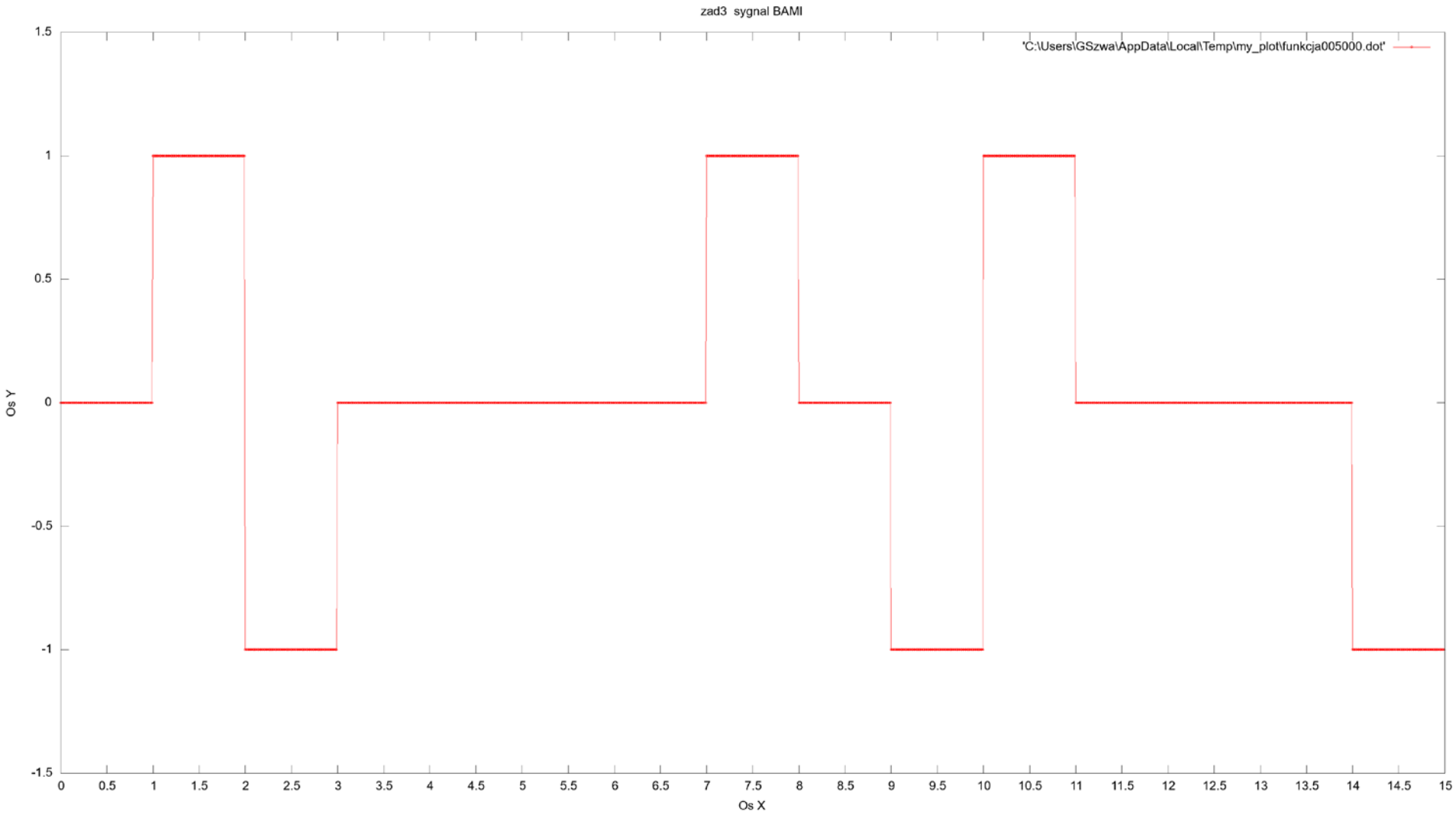
Podczas zajęć sporządziłem 6 wykresów (dodatkowo sygnał informacyjny).

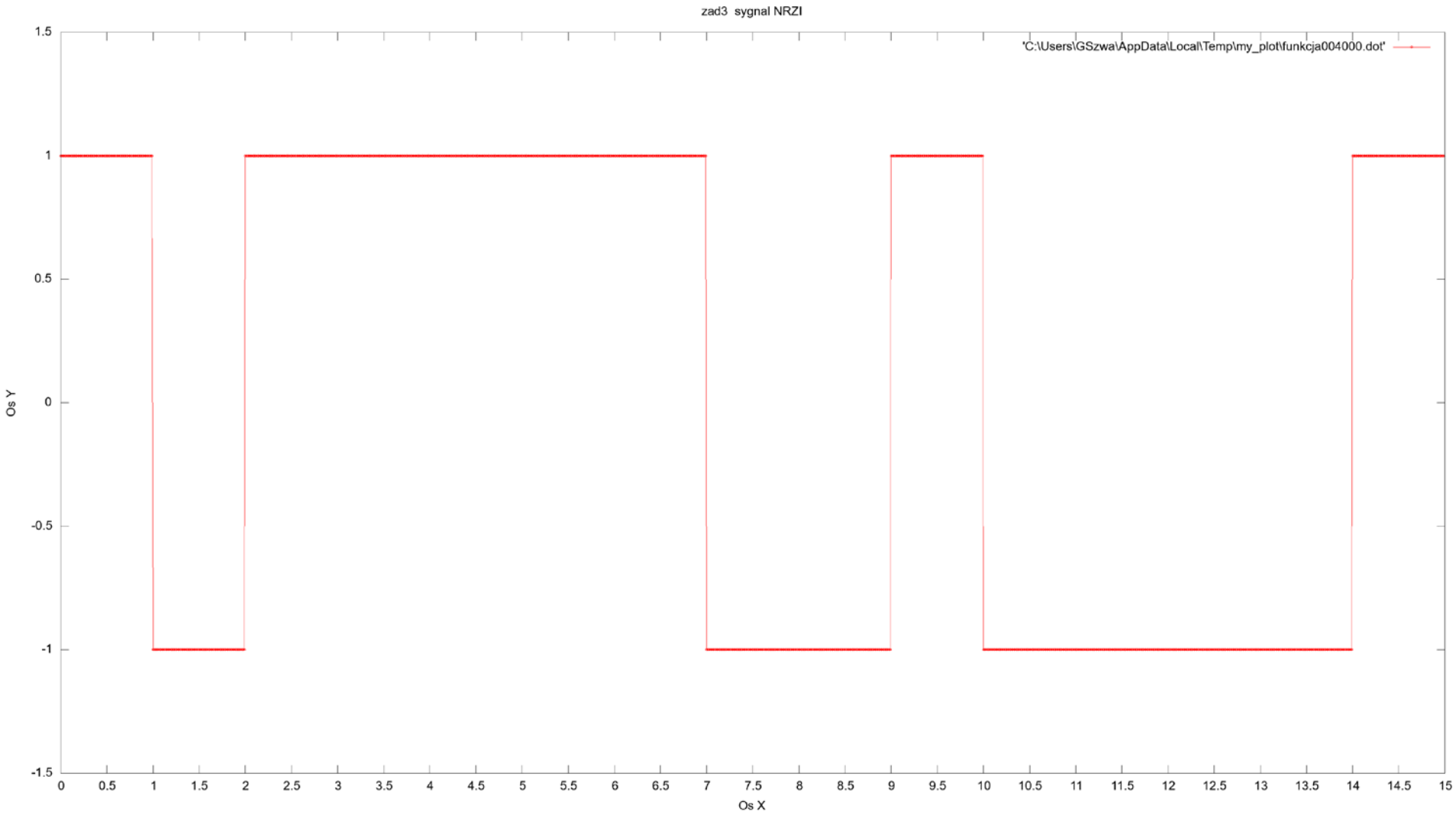



zad3 sygnal zegarowy
'C:\Users\GSzwa\AppData\Local\Temp\my_plot\funkcja001000.dot'
Os Y
Os X
1.5
1
0.5
0
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6
6.5
7
7.5
8
8.5
9
9.5
10
10.5
11
11.5
12
12.5
13
13.5
14
14.5
15
15.5
16


zad3 sygnal TTL
'C:\Users\GSzwa\AppData\Local\Temp\my_plot\funkcja002000.dot'
Os Y
Os X
1.5
1
0.5
0
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6
6.5
7
7.5
8
8.5
9
9.5
10
10.5
11
11.5
12
12.5
13
13.5
14
14.5
15


zad3 sygnal Manchester
'C:\Users\GSzwa\AppData\Local\Temp\my_plot\funkcja002000.dot'
Os Y
Os X
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6
6.5
7
7.5
8
8.5
9
9.5
10
10.5
11
11.5
12
12.5
13
13.5
14
14.5
15
15.5
16


zad3 sygnal BAMI
'C:\Users\GSzwa\AppData\Local\Temp\my_plot\funkcja005000.dot'
Os Y
Os X
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6
6.5
7
7.5
8
8.5
9
9.5
10
10.5
11
11.5
12
12.5
13
13.5
14
14.5
15


zad3 sygnal NRZI
'C:\Users\GSzwa\AppData\Local\Temp\my_plot\funkcja004000.dot'
Os Y
Os X
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6
6.5
7
7.5
8
8.5
9
9.5
10
10.5
11
11.5
12
12.5
13
13.5
14
14.5
15

Podsumowanie

Dzięki tym laboratoriom nauczyłem się przystosowywania binarnej informacji do przesyłu za pomocą analogowego ośrodka, co pozwala na zrozumienie w jaki sposób jest wysyłany sygnał za pomocą analogowego ośrodka. Wiedza to jest wykorzystywana w transmisji danych stanowiących rdzeń komunikacji pomiędzy urządzeniami cyfrowymi.

Wykonał Szwarc Grzegorz