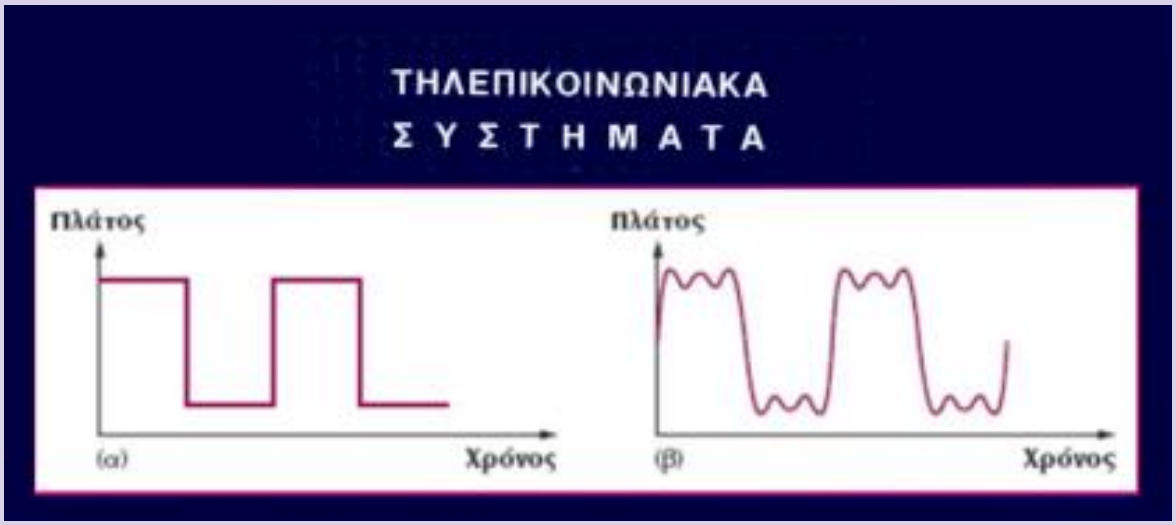
Προαιρετική εργασία στο μάθημα
"Τηλεπικοινωνιακά Συστήματα"
5ο εξάμηνο.



Χαροκόπειο Πανεπιστήμιο Αθηνών (HUA)

Τμήμα Πληροφορικής και Τηλεματικής

Ονοματεπώνυμο: Άρτεμις Στεφανίδου

Αριθμός φοιτητή: it21996

ΠΕΡΙΕΧΟΜΕΝΑ

Μεθοδολογία

Η πρώτη σκέψη πριν την γραφή του κώδικα σε python ήταν:

Για να αποτυπώσω γραφικά τον 8-ppm θα πρέπει να περάσω πρώτα από τον gray και μετά στον ppm η αντιστοιχία που κατάλαβα πως ήταν είναι η εξής :

Gray	8-PPM
000	10000000
001	01000000
011	00100000
010	00010000
110	00001000
111	00000100
101	00000010
100	00000001

Εικόνα 1.1

Binary	Decimal	Gray
0	0	0
1	1	1
3	3	2
2	2	3
6	6	4
7	7	5
5	5	6
4	4	7

Εικόνα 1.2

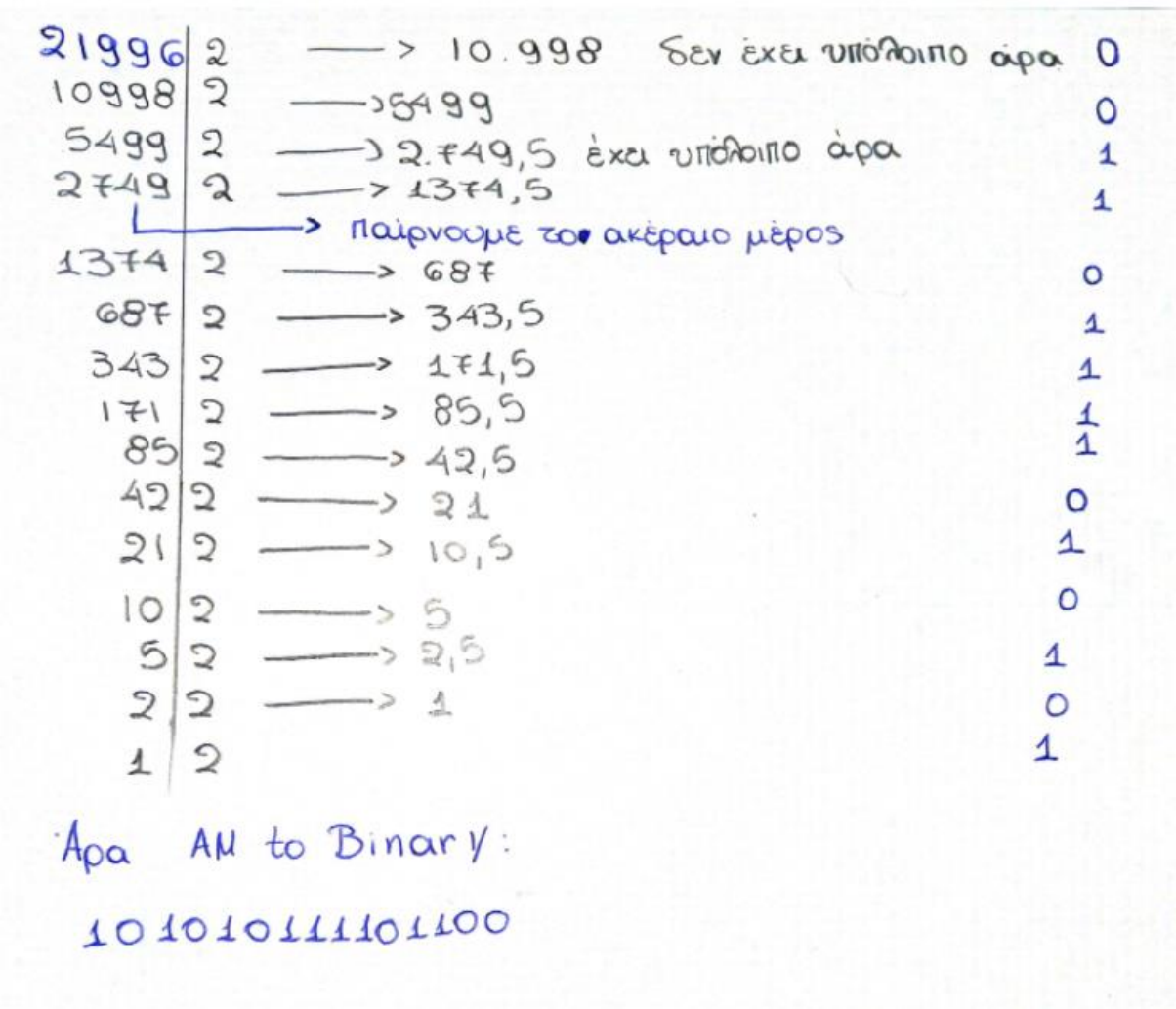
Για αρχή ας εξηγήσουμε την κάθε συνάρτηση - μέθοδο που υλοποιήσαμε :

```
#plot the signal
def plot_signal(t, x, close_all = False,show_samples = False,show_grid = False,
                plot_type = 'o',xlabel = 't', ylabel = 'x(t)'):


    if close_all:
        plt.close('all')

    plt.plot(t,x)
    plt.xlabel('t [μsec]')
    plt.ylabel('x(t)')


    if show_samples:
        samples = np.asarray(yAxons)
        samplesAxon = np.ma.masked_where(samples < 1,samples)
        plt.plot(xAxons,samplesAxon,plot_type)

    if show_grid:
        plt.grid()
```

Εικόνα 2.1

Για την plot_signal βασίστηκα σε όσα υπήρχαν και στην βιβλιοθήκη που χρησιμοποιούμε και στο μάθημα την comlib. Έτσι, οι μεταβλητές έχουν την ίδια σημασία που έχουν και στην βιβλιοθήκη που προαναφέρθηκε δηλαδή το t είναι οι τιμές στον άξονας χ'χ , το χ είναι οι τιμές στον άξονα y'y (0 ή 1 τα πλάτη δηλαδή) το show_samples είναι το αν θέλουμε να φαίνονται 10 σημεία κυματομορφής ανά παλμό (εκεί που το πλάτος είναι 1) που διαρκεί $T_S/8$ όπως αναφέρεται και στην εκφώνηση.Επίσης, κανονικά αν δεν βάλουμε τίποτα σαν όρισμα στη συνάρτηση για το label στον άξονα χ'χ τότε θα πάρει την περιγραφή "t".Εδώ όμως διευκρινίζεται πως ό,τι μέτρηση φαίνεται στον άξονα αυτό είναι πολλαπλασιασμένη με 1e-6 (αυτό γίνεται για λόγους καλύτερης απεικόνισης και διευκόλυνσης στον κώδικα).

Για την μετατροπή του ΑΜ μου σε δυαδικό χρησιμοποίησα την λογική :




```python
# AM from decimal to binary
def AM_toBinary(AM):

    queue_code = []

    x = AM

    while x> 0 :

        if x % 2 == 0:
            queue_code.append(0)
        else :
            queue_code.append(1)
        x = x / 2.0
        x = int(x)

    #add 0 to be the len multiple of 3
    while len(queue_code) % 3 != 0:
        queue_code.append(0)

    queue_code.reverse()
    return queue_code
```

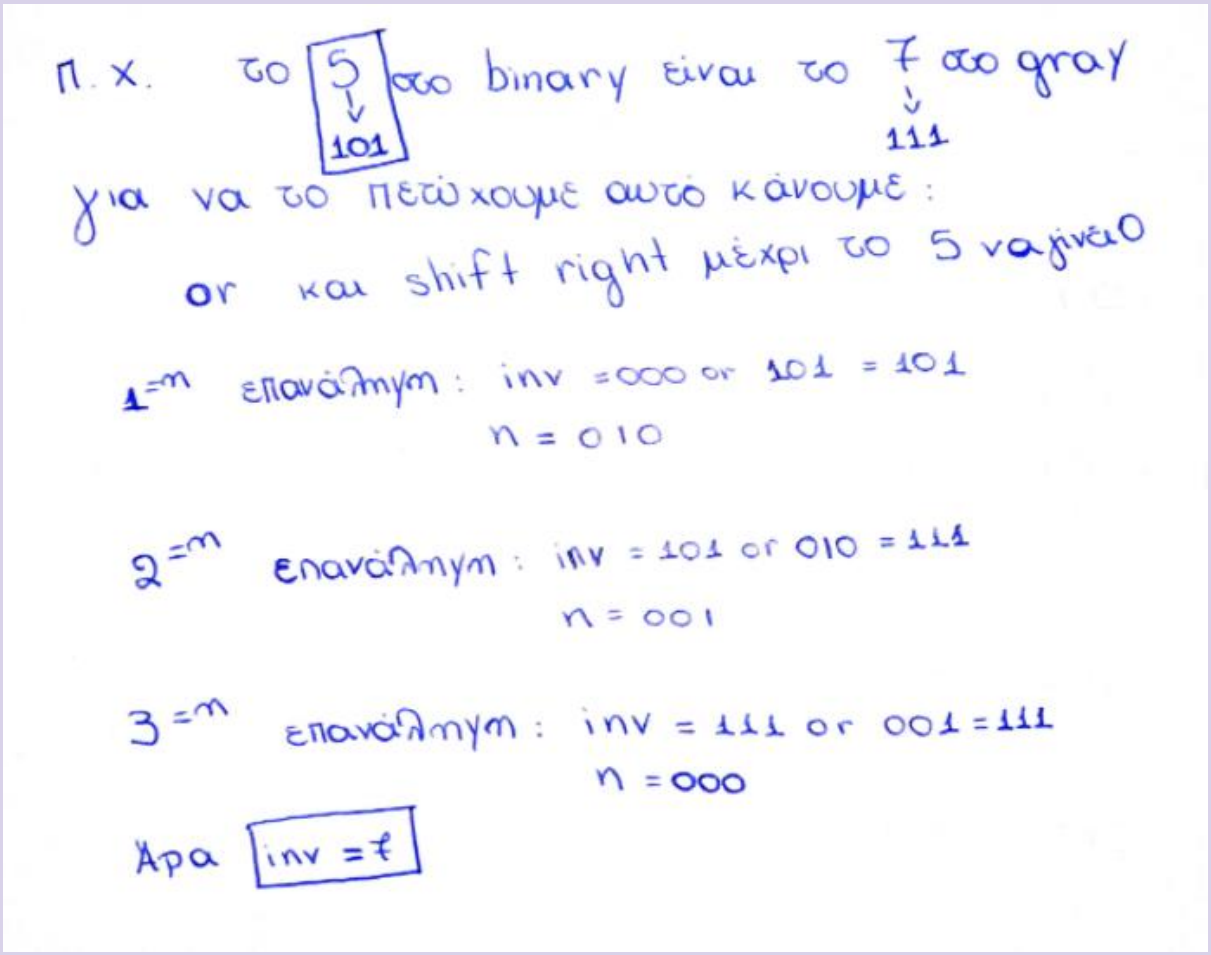
Στη συνέχεια υλοποίησα την παραδοχή που αναφέρεται στην εκφώνηση που έλεγε ότι πρέπει να μεταδώσουμε αριθμό bit που είναι ακέραιο πολλαπλάσιο του 3 γιατί έχουμε 8 σύμβολα να αναπαραστήσουμε άρα αν τα χωρίσουμε σε τριάδες τα bit θα έχουμε 2^3 = 8.Γι' αυτό το λόγο συμπλήρωσα τόσα μηδενικά στην αρχή της λίστας όσα και έλειπαν.

Εικόνα 2.2

```
#converse decimal to gray
def binaryToGray(n):
    #converse binary to
    #decimal to do the or and shift
    n = int(n,2)
    inv = 0;

    # Taking or end shift right until
    # n becomes zero
    while(n):
        inv = inv ^ n;
        n = n >> 1;
    return inv;
```

Εδώ ήθελα να μετατρέψω τον δυαδικό αριθμό που πήρα από την αναπαράσταση σε τριάδες από τον binary AM μου σε gray για να γίνει σωστά η αντιστοιχία σε 8-ppm στη συνέχεια.Ο τρόπος για να το πετύχω αυτό αναλύεται παρακάτω σε παράδειγμα (εικόνα 3.1)

Εικόνα 2.3



Εικόνα 3.1

Binary	Gray	Decimal
0	0	0
1	1	1
3	3	2
2	2	3
6	6	4
7	7	5
5	5	6
4	4	7

2.4

```
#shift the 1 to right so many times as the gray
#the correspondence between gray and 8-ppm
def shiftRight(timesOfShift,shift_number):

    for i in range(0,timesOfShift):
        shift_number = "0" + shift_number

    if shift_number == "10000000":
        return shift_number
    else:
        #how many 0 will cut from the end of the string to be 8bits again
        #to plot for the 8-ppm
        shift_number = shift_number[:-timesOfShift]
        return shift_number
```

Εικόνα 2.5

Εδώ φαίνεται η μέθοδος με την οποία αντιστοιχίζω τον gray με τον 8-ppm όπως αναφέρθηκε παραπάνω. Δηλαδή,προσθέτω τόσα 0 όσα και ο αριθμός που πήρα σε gray και μετά τα αφαιρώ από το τέλος του string.Έτσι έχω μετακινήσει το 1 προς τα δεξιά τόσες θέσεις όσες και ο αριθμός που δόθηκε στη μέθοδο

```
#separate in pairs of three the binary AM to encoded for PPM
def pairsOfThree(queue_code):

    listOfThree = []
    pairs= ""
    while len(queue_code) != 0:

        for i in range(0, 3):

            pairs = pairs + str(queue_code[0])
            queue_code.pop(0)

        listOfThree.append(pairs)
        pairs = ""
    return listOfThree
```

Εικόνα 2.6

Εδώ χωρίζω ανά τρία τα bit και την κάθε μια τριάδα την βάζω σε διαφορετική θέση στην λίστα για να κανω την κωδικοποίηση σε 8-ppm όπως εξηγήσαμε παραπάνω (2^3=8).

(γνωρίζω ότι γίνεται και με πιο γρήγορο τρόπο αλλά προτίμησα να το κάνω σε βήματα για να γινει πιο εύκολα κατανοητός ο κώδικας και ο τρόπος σκέψης μου).

```
#separate one by one the bits that i can plot them
def oneByOne(ppm_list):

    one_by_one_list = []
    #put all the ppm in a string
    temp = ''
    for i in range(0,len(ppm_list)):
        temp = temp + ppm_list[i]

    for i in range(1,len(temp)+1):
        one_by_one_list.append(int(temp[i-1:i]))
    return one_by_one_list
```

Χωρίζω σε ένα ένα τα bit από τη λίστα με τα strings που είχα για να μπορέσω να τα κάνω εύκολα plot μιας και είναι τα πλάτη που θα χρησιμοποιηθούν σε επόμενη μέθοδο.

Εικόνα 2.7

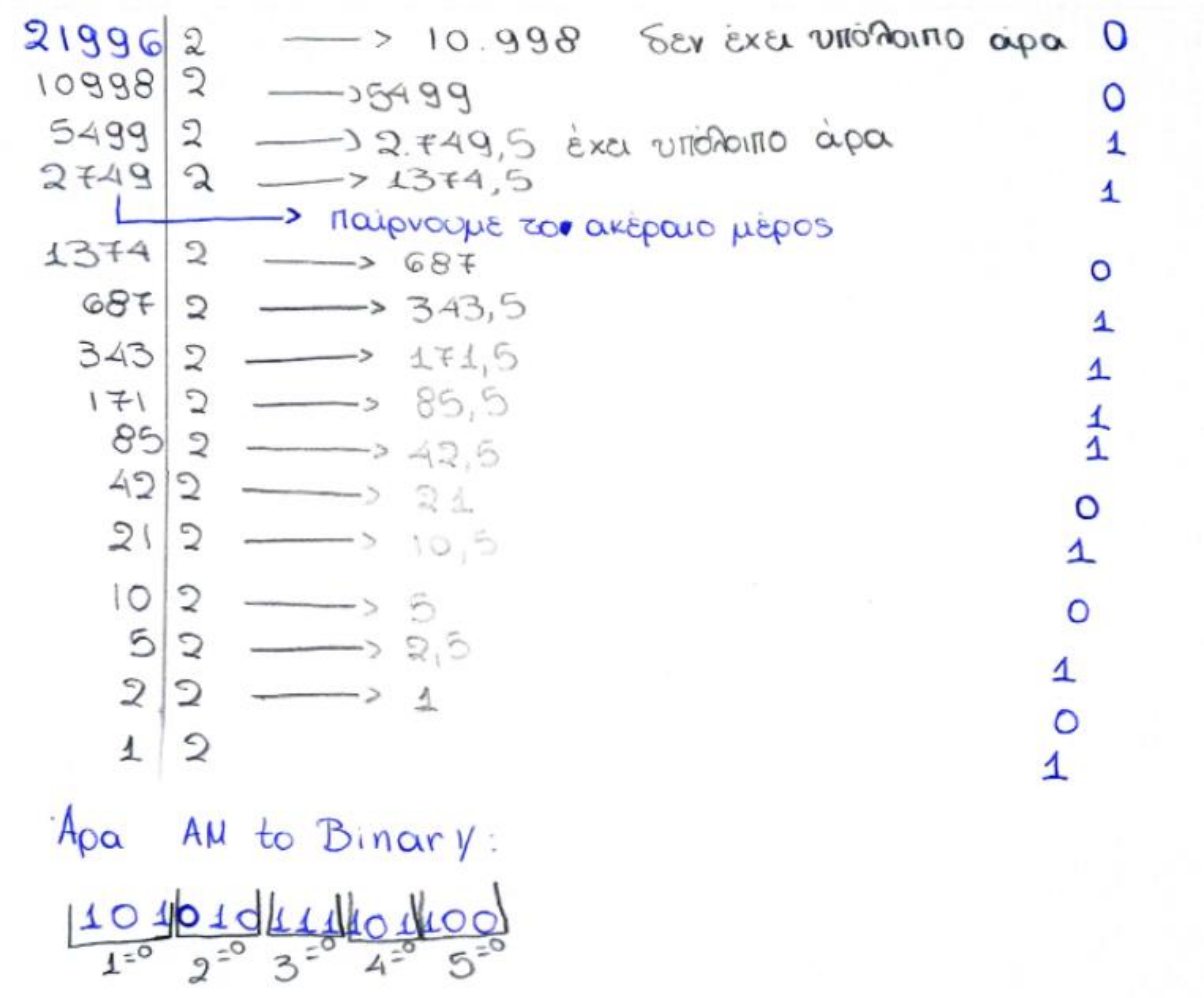
```python
# create a waveform for the PPM
def ppm_waveform(ak #ta platoi
                 , TS #period of each palm
                 , samples = 10 #the 10 samples that asked in the exercise
                 ):

    M = 8  # 8-PPM

    ak=list(np.repeat(ak,samples))

    #Πολλαπλασιάζω με 10**6 για να το μετρήσω τον χρόνο σε μs για να είναι
    #πιο ευανάγνωστο το διάγραμμα
    xAxons = list(np.arange(0, len(ak)*(Ts*10**6/(M*10)), Ts*10**6/(M*10)))
    print(len(ak)*(Ts*10**5/M))
    return xAxons,ak
```

Εικόνα 2.8

Η συνάρτηση αυτή είναι περίπου όπως και η pam_waveform που είχαμε δει στο μάθημα. Εδώ δημιουργώ τον άξονα του χρόνου αλλά και τα χ(t) που θα χρειαστώ για να αναπαραστήσω τον 8-ppm σε ένα διάγραμμα. Τα σημεία στον άξονα του χ'χ δεν είναι τυχαία αλλά είναι τόσα όσα και τα πλάτη που έχω και έχουν απόσταση μεταξύ τους αυτή που ζητήται από την εκφώνηση. Επίσης, επειδή ζητήθηκε να παρθούν 10 σημεία κυματομορφής ανά παλμό έπρεπε να πολλαπλασιάσω το κάθε πλάτος που είχα επί τον αριθμό των samples (repeat() → όσα περισσότερα samples δοθούν τόσο πιο τετραγωνικός θα είναι ο παλμός).

Πως Επαλήθευσα Τον Κώδικα

Το ΑΜ μου σε binary με βάση την υλοποίηση στο χαρτί βγαίνει :



και στον κώδικα είναι :

```
In [62]: runfile('/home/artemis/Desktop/queueGray.py',
wdir='/home/artemis/Desktop')

In [63]: listOfThree
Out[63]: ['101', '010', '111', '101', '100']
```

Η αντιστοιχία με τον gray για τις παραπάνω τριάδες είναι :

Gray	Αρίθμηση
000	0
001	1
011	2
010	3
110	4
111	5
101	6
100	7

και το αποτέλεσμα που βγάζει το πρόγραμμα είναι :

```
In [64]: gray_list
Out[64]: [6, 3, 5, 6, 7]
```

Επίσης η αντιστοιχία του gray με τον 8-ppm είναι :

Gray	Αρίθμηση	8-PPM
000	0	10000000
001	1	01000000
011	2	00100000
010	3	00010000
110	4	00001000
111	5	00000100
101	6	00000010
100	7	00000001

και το πρόγραμμα δείχνει:

```
In [64]: gray_list
Out[64]: [6, 3, 5, 6, 7]

In [65]: ppm_list
Out[65]: ['00000010', '00010000', '00000100', '00000010', '00000001']
```

Στη συνέχεια τα πλάτη που έχω είναι 40 και επί 10 που είναι ο default αριθμός για τα samples που υπάρχει στην συνάρτηση ppm_waveform και που ζητήθηκε και από την εκφώνηση βγαίνει ότι πρέπει να έχω 400 σημεία που θα αναπαραστήσω στον άξονα 0χ και αυτό στο πρόγραμμα επαληθεύεται :

```
In [66]: len(one_by_one_list)
Out[66]: 40

In [67]: len(one_by_one_list) * 10
Out[67]: 400

In [68]: len(xAxons)
Out[68]: 400
```

Η απόσταση μεταξύ των τιμών στον άξονα χ'χ θα πρέπει να είναι Ts / 8 * samples. Το Ts όμως είναι 3*1e-6 (**ρυθμός μετάδοσης ίσος με 1Mb/s → Άρα σε Tς μεταδίδουμε 3bit , σε 1s → 10^6**) όμως το διάγραμμα μας δείχνει τις τιμές σε μs άρα η απόσταση που θα πρέπει να έχουν τα σημεία μεταξύ τους είναι 3 / 80 = 0,0375

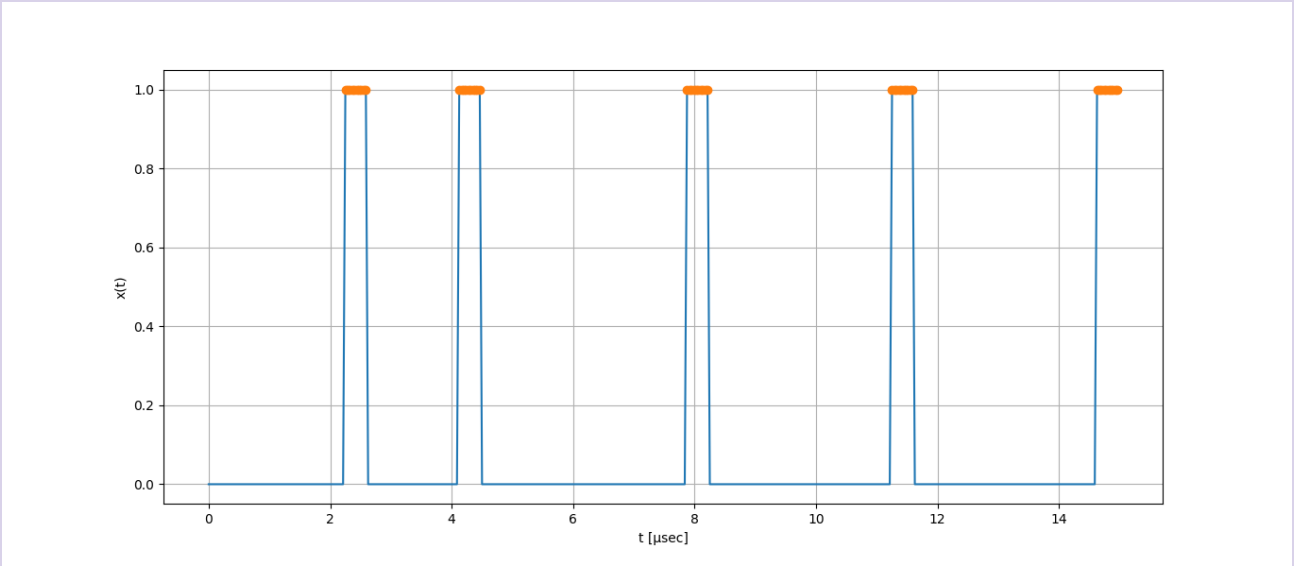
```
In [70]: xAxons[1] - xAxons[0]
Out[70]: 0.0375
```

και κάθε bit θα πρέπει να απέχει Ts / 8 →3 / 8 = 0,375

```
In [73]: xAxons[10] - xAxons[0]
Out[73]: 0.375
```

Τέλος, το διάγραμμα θα πρέπει να έχει τόσους παλμούς όσες και οι τριάδες στο πρόγραμμα αφού κάθε κομμάτι του 8-ppm έχει μόνο ένα 1

```
In [71]: listOfThree
Out[71]: ['101', '010', '111', '101', '100']

In [72]: ppm_list
Out[72]: ['00000010', '00010000', '00000100', '00000010', '00000001']
```



Τα 10 σημεία κυματομορφής ανά παλμό που ζητήθηκαν: