# Τηλεπικοινωνιακά Συστήματα

## Διάλεξη 5η

Θωμάς Καμαλάκης

Χαροκόπειο Πανεπιστήμιο Αθηνών

Οκτώβριος 2020

# Περιεχόμενα

# Διαμόρφωση κατά πλάτος

- Ήδη στα προηγούμενα είδαμε κάποια πράγματα για την διαμόρφωση κατά πλάτος.
- Στην ουσία αλλάζουμε το πλάτος που έχουν διαδοχικοί παλμοί $p(t - kT_{\mathrm{S}})$ ανάλογα με τα σύμβολα $a_i$ που θέλουμε να μεταδώσουμε.
- Ωστόσο, πως φτιάχνουμε τα σύμβολα $a_i$ από τα bits που έχουμε;
- Έχει νόημα να χρησιμοποιήσουμε και άλλα είδη παλμών $p(t)$ εκτός από τετραγωνικούς;
- Τι γίνεται όταν θέλουμε να μεταδώσουμε στις υψηλές συχνότητες;

# Πως επιλέγουμε τα σύμβολα;

- Είδαμε στα προηγούμενα ότι θέλουμε:

$$\mathbb{E}\{a_p\} = 0$$

- αν θεωρήσουμε ότι τα σύμβολα είναι ισοπίθανα και ότι οι πιθανές τους τιμές είναι τα $A_m$ για $1 \leq m \leq M$ όπου $M$ το πλήθος των συμβόλων, τότε:

$$\Pr\{a_p = A_m\} = \frac{1}{M}$$

$$\mathbb{E}\{a_p\} = \sum_{m=1}^{M} A_m \Pr\{a_p = A_m\} = \frac{1}{M} \sum_{m=1}^{M} A_m$$

- Επομένως για να έχουμε $\mathbb{E}\{a_p\} = 0$ θα πρέπει και

$$\sum_{m=1}^{M} A_m = 0$$

# Πως επιλέγουμε τα σύμβολα;

- Μία πιθανή επιλογή είναι να έχουμε:

$$A_m = \beta(2m - M - 1)$$

- Αυτό σημαίνει ότι:

$$A_{m+1} - A_m = 2\beta$$

- δηλαδή οτι τα σύμβολα ισαπέχουν. Επίσης:

$$A_{M-m+1} = \beta(2M - 2m + 2 - M - 1) = \beta(M - 2m + 1) = -A_m$$

- Επομένως τα σύμβολα είναι συμμετρικά τοποθετημένα γύρω από το 0.

# Αστερισμοί συμβόλων

- Είναι χρήσιμο να παριστάνουμε γραφικά τα σύμβολα $A_m$.
- Τα $A_m$ στην περίπτωση του PAM είναι πραγματικοί αριθμοί.
- Θα δούμε παρακάτω ότι μπορεί να έχουμε σύμβολα $S_m$ που είναι *μιγαδικά* δηλαδή να έχουν πραγματικό και φανταστικό μέρος,

$$S_m = I_m + jQ_m$$

- Ο *αστερισμός* των συμβόλων είναι ένα διάγραμμα που τα αναπαριστά στο μιγαδικό επίπεδο.
- Στην περίπτωση του PAM τα σύμβολα θα είναι πάνω στον φανταστικό άξονα.

# Αστερισμοί συμβόλων: plotpamsymbols.py

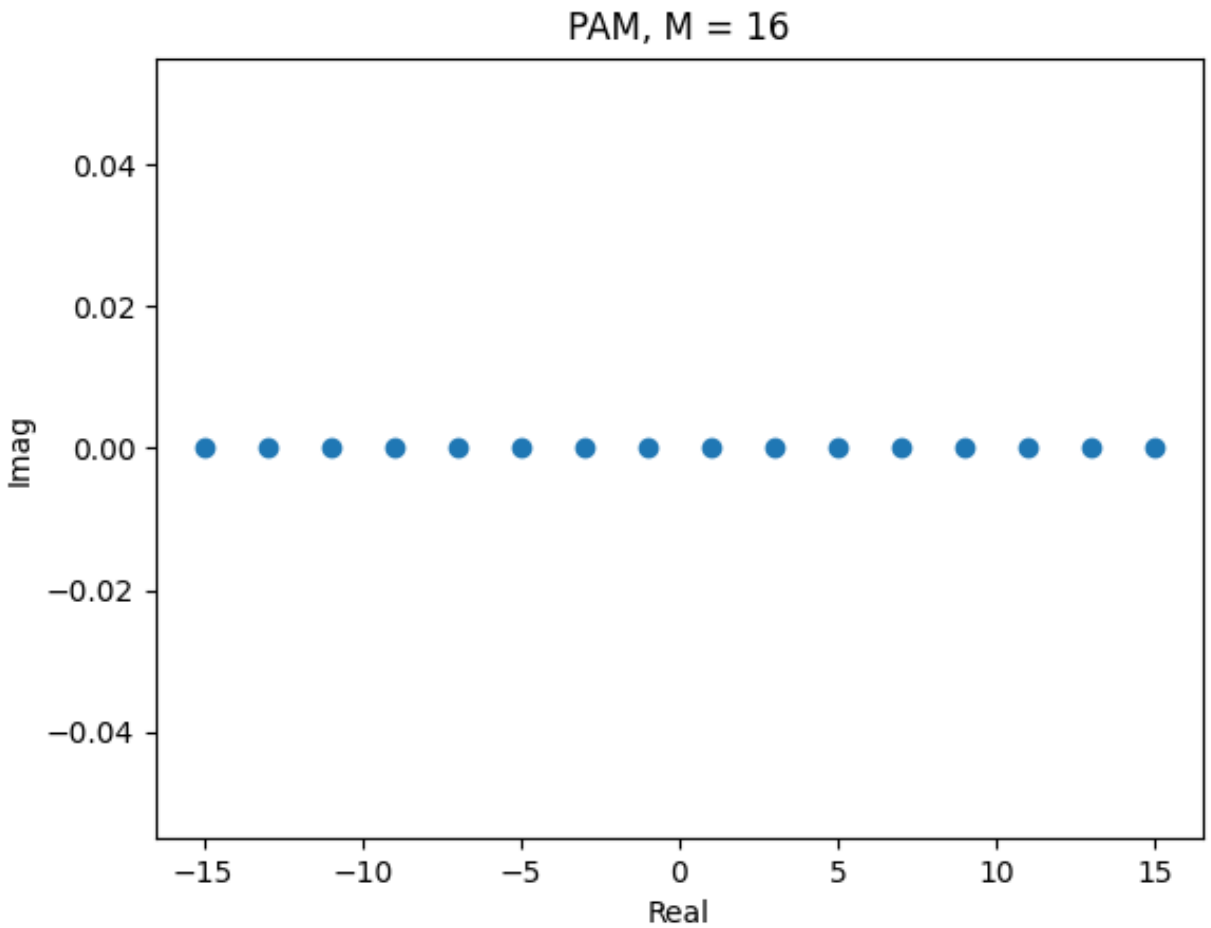
```
1 import commlib as cl
2
3 M = 16
4 Am = cl.pam_constellation( M )
5 cl.plot_constellation( Am, title = 'PAM, M = ' +
      str(M))
```

## Μερικές αλλαγές στην comm_lib.py:

```
1     return y
2
3 # PAM symbol constellation
4 def pam_constellation(M, beta = 1):
5     m = np.arange(1, M + 1).astype(int)
6     return (2 * m - M - 1) * beta
```

```
7
8 # Plot symbol constellation
9 def plot_constellation(c, figure_no = None, title
      = None):
10
11     if figure_no is None:
12         plt.figure()
13     else:
14         plt.figure(figure_no)
15
16     cr = np.real(c)
17     ci = np.imag(c)
18     plt.plot(cr, ci, 'o')
19     plt.xlabel('Real')
20     plt.ylabel('Imag')
```

# Αστερισμοί συμβόλων: plotpamsymbols.py



# Ισχύς συμβόλων του PAM

- Για να υπολογίσουμε την μέση ισχύ των συμβόλων $\sigma_a^2$ έχουμε:

$$\sigma_a^2 = \mathbb{E}\left\{a_p^2\right\} = \sum_{m=1}^{M} A_m^2 \Pr\left\{a_p = A_m\right\} = \frac{\beta^2}{M}\sum_{m=1}^{M}(2m - M - 1)^2 =$$

$$\frac{4\beta^2}{M}\sum_{m=1}^{M} m^2 - 4\beta^2\sum_{m=1}^{M} m + \beta^2 = \beta^2\frac{M^2-1}{3}$$

- Η μέγιστη ισχύς $P_{\max}$ είναι η ισχύς που αντιστοιχεί στο σύμβολο $A_M = -A_1$, δηλαδή:

$$P_{\max} = \beta^2(M-1)^2$$

- Για μεγάλες τιμές του $M$ θα έχουμε:

$$\frac{P_{\max}}{\sigma_a^2} = 3\frac{M^2 - 2M - 1}{M^2 - 1} \cong 3$$

# Ισχύς της κυματομορφής PAM

- Η κυματομορφή του PAM έχει φασματική πυκνότητα:

$$S_X(f) = \frac{\sigma_a^2}{T_S}|P(f)|^2$$

- Επομένως η μέση ισχύς του PAM είναι:

$$P_{\text{av}} = \int_{-\infty}^{+\infty} S_X(f)\mathrm{d}f = \frac{\sigma_a^2}{T_S}\int_{-\infty}^{+\infty}|P(f)|^2\mathrm{d}f = \frac{\sigma_a^2}{T_S}\int_{-\infty}^{+\infty}|p(t)|^2\mathrm{d}t$$

- Δεδομένου ότι το $\mathcal{E}_p = \int_{-\infty}^{+\infty}|p(t)|^2\mathrm{d}t$ είναι η ενέργεια του παλμού, έχουμε:

$$P_{\text{av}} = \frac{\sigma_a^2}{T_S}\mathcal{E}_p = \frac{\beta^2}{T_S}\mathcal{E}_p\frac{M^2-1}{3}$$

# Μερικές παρατηρήσεις

- Παρατηρούμε ότι όσο αυξάνουμε τον αριθμό των συμβόλων $M$ αυξάνεται και η μέση ισχύς ($\propto M^2$).
- Όσο αυξάνει η απόσταση των συμβόλων $2\beta = A_{m+1} - A_m$ αυξάνεται η μέση ισχύς ($\propto \beta^2$)
- Το $\mathcal{E}_p/T_\mathrm{S}$ μπορεί να ερμηνευθεί ως η μέση ισχύς του παλμού $p(t)$ μέσα στη διάρκεια ενός συμβόλου.
- αρκεί βέβαια ο παλμός να είναι μηδενικός εκτός της διάρκειας αυτής, δηλαδή $p(t) = 0$ όταν $|t| < \frac{T_\mathrm{S}}{2}$.

$$\frac{1}{T_\mathrm{S}} \int_{-\infty}^{+\infty} |p(t)|^2 \mathrm{d}t = \frac{1}{T_\mathrm{S}} \int_{-T_\mathrm{S}/2}^{T_\mathrm{S}/2} |p(t)|^2 \mathrm{d}t$$

# Πως μετατρέπουμε τα bits σε σύμβολα;

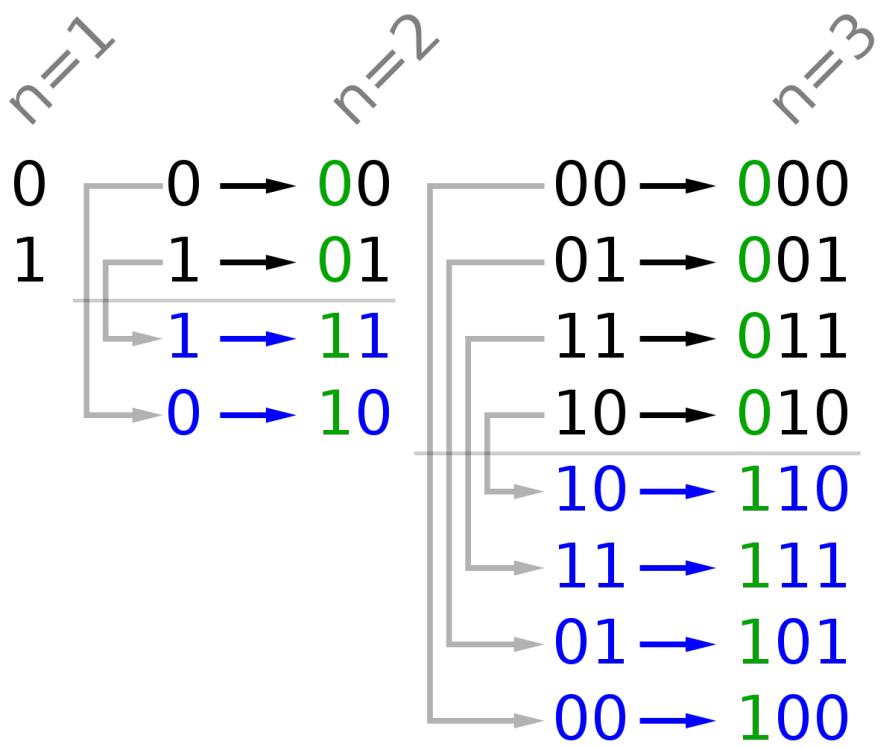
- Μία ιδέα είναι να χωρίσουμε τα bit που έχουμε να μεταδώσουμε σε ομάδες.
- Αν κάθε ομάδα έχει $m$ bits τότε μπορούμε να φτιάξουμε $M = 2^m$ διαφορετικούς συνδυασμούς από bits.
- Π.χ. αν $m = 3$ θα έχουμε $\{000, 001, 010, 011, 100, 101, 110, 111\}$ δηλαδή $M = 2^3 = 8$ συνδυασμούς.
- σε κάθε έναν από αυτούς του συνδυασμούς αναθέτουμε ένα ξεχωριστό σύμβολο (πλάτος) PAM
- Ο πιο εύκολος (άλλα όχι ο καλύτερος) τρόπος θα είναι να πάρουμε τα σύμβολα $A_m$ με την σειρά.
- Έτσι το $A_1$ αντιστοιχεί στο $000$, το $A_2$ στο $001$, κτλ.

# Κωδικοποίηση Gray

- Υπάρχει και ένας καλύτερος τρόπος να αναθέσουμε τους συνδυασμούς σε σύμβολα.
- Θέλουμε να εξασφαλίσουμε ότι στα γειτονικά σύμβολα, οι συνδυασμοί δεν διαφέρουν πολύ.
- Έτσι όταν γίνεται ένα σφάλμα και μετακινούμαστε σε γειτονικό σύμβολο (π.χ. εξαιτίας του θορύβου)
- Μπορούμε να ορίσουμε έναν αναδρομικό τρόπο υλοποίησης του κώδικα.
- Στην ουσία περιγράφουμε πως παράγεται ο κώδικας με κωδικές λέξεις μήκους $k$ bits από τον κώδικα με μήκος $k-1$ bits.
- Έστω ότι ο κώδικας με μήκος $k-1$ bits έχει τις κωδικές λέξεις $\{c_1, c_2, \cdots, c_L\}$ με $L = 2^{k-1}$.

# Κωδικοποίηση Gray

- Για να φτιάξουμε τον κώδικα με μήκος $k$ bits βάζουμε ένα «0» μπροστά από τις λέξεις του κώδικα με μήκος $k-1$
- επίσης αντιστρέφουμε την σειρά των λέξεων του κώδικα με μήκος $k-1$ και $\{c_L, c_{L-1}, \cdots, c_1\}$ βάζουμε μπροστά το «1»



# Υλοποίηση σε Python

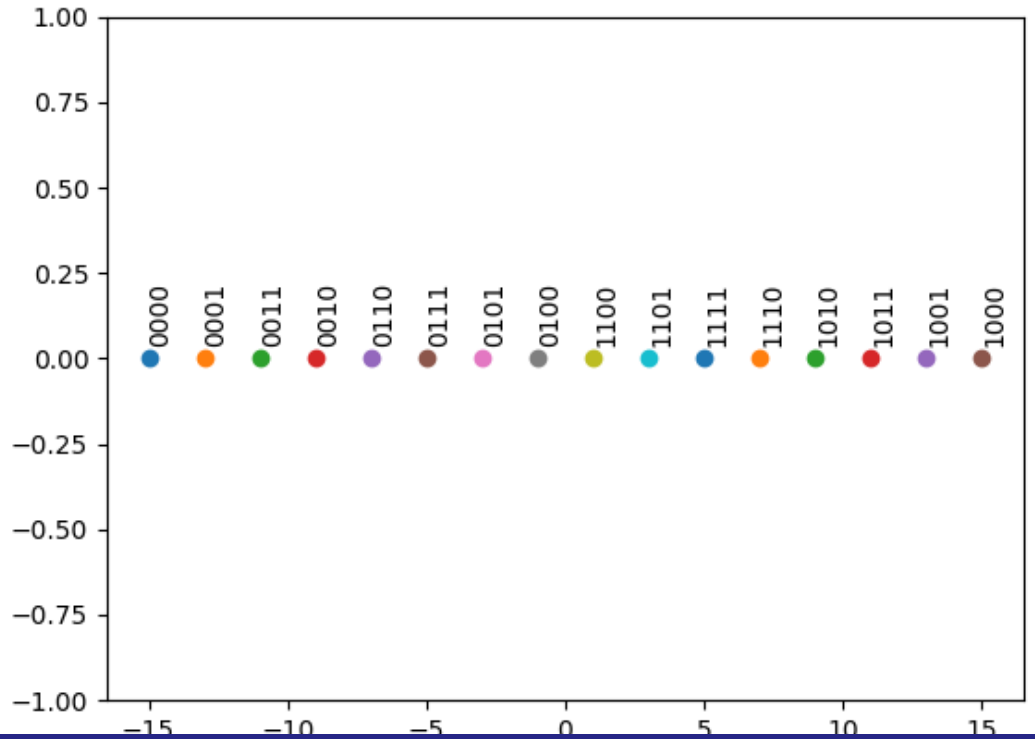
## Μερικές προσθήκες στην comm_lib

```
1
2  # gray coding
3  def gray_code(m):
4
5      if m==1:
6          g = ['0','1']
7
8      elif m>1:
9          gs = gray_code(m-1)
10         gsr = gs[::-1]
11         gs0 = ['0' + x for x in gs]
12         gs1 = ['1' + x for x in gsr]
13         g= gs0 + gs1
14     return g
15
16 def pam_gray_map(M, beta = 1):
```

```
17
18     gc = gray_code( np.log2(M) )
19     Am = pam_constellation(M, beta = beta)
20     pam_map = []
21
22     for i, cw in enumerate(gc):
23
24         pam_map.append([ i, gc[i], Am[i] ])
25
26     return pam_map
27
28 def plot_map( smap, figure_no = None, disp_x =
       0.0, disp_y = 0.0 ):
29
30     for i, bits, symbol in smap:
```

# Υλοποίηση σε Python: pam_gray_code.py

```
1 import commlib as cl
2 import matplotlib.pyplot as plt
3
4 pam_map = cl.pam_gray_map(16)
5 plt.close('all')
6 cl.plot_map(pam_map, disp_y = 0.05)
7 plt.ylim([-1, 1])
```



# Υλοποίηση σε Python: Κωδικοποίηση

- Ίσως είναι πιο αποδοτικό να φτιάξουμε ένα dictionary.
- Είναι μία έτοιμη δομή δεδομένων της Python.
- Στην ουσία αντιστοιχούμε τιμές σε strings (κλειδιά).
- Π.χ. 'thomas'→10, 'bob'→'my friend', 'alice'→[1,2]

```
In [36]: a = {}

In [37]: a['thomas']=10

In [38]: a['bob'] = 'my friend'

In [39]: a['alice'] = [1,2]

In [40]: a
Out[40]: {'thomas': 10, 'bob': 'my friend', 'alice': [1, 2]}
```

# Υλοποίηση σε Python: Κωδικοποίηση (comm_lib.py)

- Η παρακάτω συνάρτηση φτιάχνει ένα dictionary για τα σύμβολα του PAM.
- Κάνει map τα bits σε σύμβολα

```
1
2  # build a dictionary like map for faster encoding
3  def pam_gray_forward_map(M, beta = 1):
4
5      pam_map = pam_gray_map(M, beta = beta)
6      symbols = [x[2] for x in pam_map]
7      bits = [x[1] for x in pam_map]
8      forward_map = {}
9
10     for i, symbol in enumerate(symbols):
11         key = bits[i]
12         forward_map[ key ] = symbol
```

# Υλοποίηση σε Python: Κωδικοποίηση (comm_lib.py)

```
    return forward_map

def array_to_str(a):
    astr = [str(x) for x in a]
    return ''.join(astr)

# bits to symbols
def bits_to_symbols(bits, fmap, return_bits = False):

    M = len( fmap )
    m = np.log2( M ).astype( int )
    Nbits = bits.size

    symbols = []
    bitgroups = []
    i = 0
    j = 0

    while i < Nbits:
        key = array_to_str(bits[ i : i+m ])
        bitgroups.append( key )
        symbols.append( fmap[ key ] )
        i += m
        j += 1
    if not return_bits:
        return np.array(symbols)
    else:
        return np.array(symbols), bitgroups

# random_bits : generation of random bits with equal probability
```

# Υλοποίηση σε Python: Κωδικοποίηση (comm_lib.py)

```
    return np.random.randint(0, high = 2, size = Nbits, dtype = int)

# plot_pam : plot pam waveform and show associated bits
def plot_pam(t, x, M, bits, TS, tstart = 0, dy = 0, plot_type = 'o'):

    plot_signal(t, x, xlabel = 't', ylabel = 'x(t)',
                plot_type = plot_type)

    Nbits = 0
    i = 0
    m = np.log2(M).astype(int)
    tcurrent = tstart

    while ( i < Nbits ) or ( tcurrent <= np.max(t) ):
       key = array_to_str(bits[ i : i+m ])
       xcurrent = np.interp( tcurrent, t, x )
       plt.text(tcurrent, xcurrent + dy, key)
```

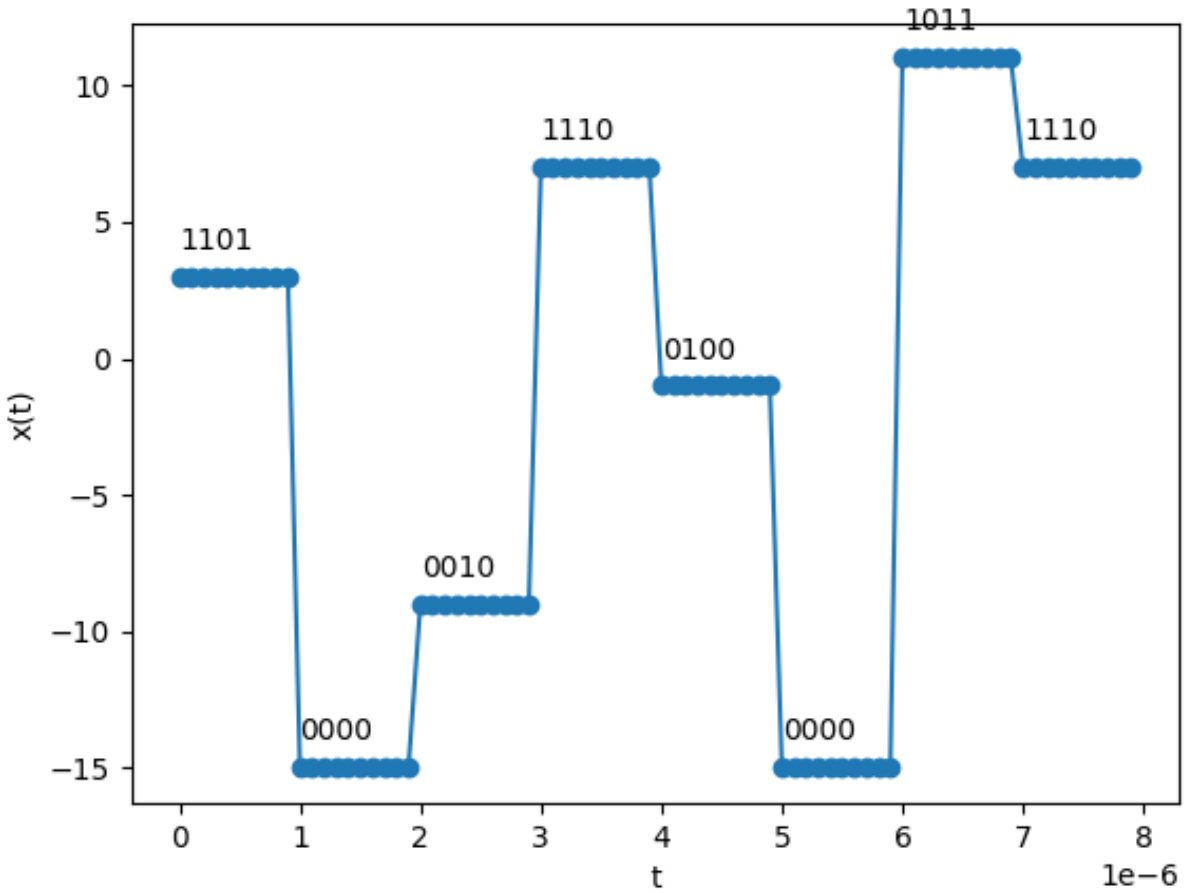
# Υλοποίηση σε Python: Κωδικοποίηση (pam_encoding.py)

```
1  import commlib as cl
2  import matplotlib.pyplot as plt
3
4  M = 16
5  Nbits = 32
6  TS = 1e-6
7
8  fmap = cl.pam_gray_forward_map(M)
9  bits = cl.random_bits(Nbits)
10 symbols, bitgroups = cl.bits_to_symbols(bits, fmap, return_bits = True)
11
12 print('Symbol mappings:')
13 print(fmap)
14 print('Input bits:')
15 print(bits)
16 print('Bit groups:')
17 print(bitgroups)
18 print('Encoded symbols:')
19 print(symbols)
20
21
22 t, x = cl.pam_waveform2(symbols, TS)
23 plt.close('all')
24 cl.plot_pam(t, x, M, bits, TS, dy = 1, plot_type = '-o')
```

# Υλοποίηση σε Python

```
Symbol mappings:
{'0000': -15, '0001': -13, '0011': -11, '0010': -9, '0110': -7,
'0111': -5, '0101': -3, '0100': -1, '1100': 1, '1101': 3, '1111':
5, '1110': 7, '1010': 9, '1011': 11, '1001': 13, '1000': 15}
Input bits:
[1 1 0 1 0 0 0 0 0 0 1 0 1 1 1 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 1 0]
Bit groups:
['1101', '0000', '0010', '1110', '0100', '0000', '1011', '1110']
Encoded symbols:
[  3 -15  -9   7  -1 -15  11   7]
```

# Υλοποίηση σε Python



# Γιατί να κάνουμε κλάσεις;

- Δεν ξέρω για εσάς αλλά εγώ δυσκολεύομαι να ψάχνω σε μία συνεχή παράθεση από functions που δεν είναι οργανωμένη.
- Οι κλάσεις μας επιτρέπουν να οργανωθούμε καλύτερα.
- Θα χρησιμοποιήσουμε λίγο object oriented programming.
- Θα μας βοηθήσει αυτό να επαναχρησιμοποιήσουμε κώδικα που γράφουμε (π.χ. μέσω της κληρονομικότητας).
- Θυμηθείτε το DRY - Do not Repeat Yourself.

# Μερικές βασικές κλάσεις

- Ας κάνουμε μερικές βασικές κλάσεις (objects):
- *signal*: πρόκειται για ένα γενικό *αντικείμενο* σήματος. Περιέχει τον άξονα του χρόνου, τα δείγματα του σήματος στο πεδίο του χρόνου και στο πεδίο των συχνοτήτων.
- Επίσης περιέχει *μεθόδους* για την διαχείριση των σημάτων: υπολογισμό του φάσματος, ενέργειας, φασματικής πυκνότητας καθώς και visualization.
- χρησιμοποιώντας την signal ως πατέρα μπορούμε να ορίσουμε και άλλες κλάσεις σημάτων όπως ο τετραγωνικός πλαμός (square_pulse) και το σήμα του φέροντος (carrier).
- Η signal μπορεί να περιγράψει ένα οποιοδήποτε αναλογικό σήμα.

# Η κλάση signal

Μερικά attributes:

- `t` : Ο άξονας του χρόνου.
- `Dt` : Η δειγματοληψία στον άξονα του χρόνου.
- `N` : Το πλήθος των δειγμάτων.
- `samples` : Το δείγματα στο πεδίο του χρόνου.
- `f` : Ο άξονας των συχνοτήτων.
- `Df` : Η συχνοτική απόσταση μεταξύ διαδοχικών σημείων στον άξονα των συχνοτήτων.
- `spec` : Το φάσμα του σήματος (όπως αυτό υπολογίζεται από τον FFT).
- Επίσης και μερικά attributes που έχουν να κάνουν με το visualization, π.χ. `xlabelt`, `xlabelf` κτλ.

# Η κλάση signal: Μέθοδοι

- `__init__` : Αρχικοποιεί την κλάση.
- `set_time_axis` : φτιάχνει τον άξονα του χρόνου.
- `set_time_axis` : φτιάχνει τον άξονα των συχνοτήτων.
- `calc_spectrum` : Υπολογίζει το φάσμα (με τον FFT).
- `calc_invspectrum` : Υπολογίζει το σήμα από το φάσμα (με τον IFFT).
- `energy` : Υπολογίζει την ενέργεια του σήματος.
- `average_power` : Υπολογίζει την μέση ισχύ του σήματος.
- `power_density` : Υπολογίζει την φασματική πυκνότητα ισχύος.
- `set_default_plot_properties` : Θέτει μερικές προκαθορισμένες τιμές για τα attributes που καθορίζουν το visualization.
- `plot` : κάνει γραφική παράσταση του σήματος.
- `windowed` : επιστρέφει μία παραθυρωμένη έκδοση του σήματος.

# comm_libv2

Η δομή του αρχείου comm_libv2 που θα χρησιμοποιούμε από εδώ και πέρα έχει ως εξής:

- Ορισμός ενός dictionary `DEFAULT_PLOT_SETTINGS` που ορίζει κάποιες τυπικές τιμές για τις παραμέτρους του visualization.
- Κάποιες functions που δεν ανήκουν σε κλάσεις οι οποίες θα χρησιμοποιηθούν σε μεθόδους των κλάσεων που φτιάχνουμε.
- Ορισμός των κλάσεων και των μεθόδων τους.

# Η κλάση constellation

Attributes:

- `bits` : ένα numpy array που περιέχει τα bits που αντιστοιχούν σε κάθε σύμβολο.
- `symbols` : Τα σύμβολα του αστερισμού.
- `bits_str` : Τα bits του attribute `bits` σε μορφή string.
- `bits_map` : Ένα dictionary που αντιστοιχεί bits σε σύμβολα.
- `title` : Ο τίτλος του σχήματος διαμόρφωσης στο οποίο αντιστοιχεί ο αστερισμός.
- `m` : Ο αριθμός των bits ανά σύμβολο.

# Η κλάση constellation

Methods:

- `plot` : Αναπαράσταση των συμβόλων του αστερισμού
- `plot_map` : Αναπαράσταση των συμβόλων του αστερισμού μαζί με τα αντίστοιχα bits.
- `set_symbols` : Ανάθεση των συμβόλων του αστερισμού.
- `set_gray_bits` : Αντιστοίχιση συμβόλων και bits σύμφωνα με τον κώδικα Gray.
- `title` : Ο τίτλος του σχήματος διαμόρφωσης στο οποίο αντιστοιχεί ο αστερισμός.
- `bits_to_symbols` : Κωδικοποίηση μίας ακολουθίας από bits σε σύμβολα βάσει του αστερισμού.

# Η κλάση pam_constellation

- Έχει ως γονέα την κλάση `constellation`.
- Δημιουργεί τον αστερισμό του PAM τάξης `M` που έχουν απόσταση ίση με το διπλάσιο του `beta`.
- Η κωδικοποίηση γίνεται βάσει του κώδικα Gray.

# Η κλάση digital_signal

Attributes:

- `TS` : Διάρκεια του κάθε συμβόλου.
- `Tmin` : Η αρχή του άξονα του χρόνου.
- `Tmax` : Το τέλος του άξονα.
- `samples_per_symbol` : Αριθμός δειγμάτων σήματος ανά σύμβολο.
- `tguard` : Αρχικό και τελικό χρόνικο διάστημα όπου το σήμα είναι ίσο με μηδέν εκτός της διάρκειας των συμβόλων.
- `constellation` : Αστερισμός που χρησιμοποιούμε για την διαμόρφωση του σήματος.
- `input_bits` : Τα bits εισόδου (προς μετάδοση).
- `Dt` : περίοδος δειγματοληψίας στο πεδίο του χρόνου

# Η κλάση digital_signal

Methods:

- `__init__` : Δημιουργία του σήματος.
- `set_constellation` : Ορισμός αστερισμού διαμόρφωσης.
- `set_input_bits` : Ορισμός bits εισόδου.
- `modulate_from_symbols` : Δημιουργία κυματομορφής από μία ακολουθία συμβόλων.
- `modulate_from_bits` : Δημιουργία κυματομορφής από μία ακολουθία bits.

# Η κλάση system

Attributes:

- `input_signal` : Το σήμα στην είσοδο.
- `output_signal` : Το σήμα στην έξοδο.
- `transfer_function` : Ένα Python callable που περιγράφει την συνάρτηση μεταφοράς του συστήματος.

# Η κλάση system

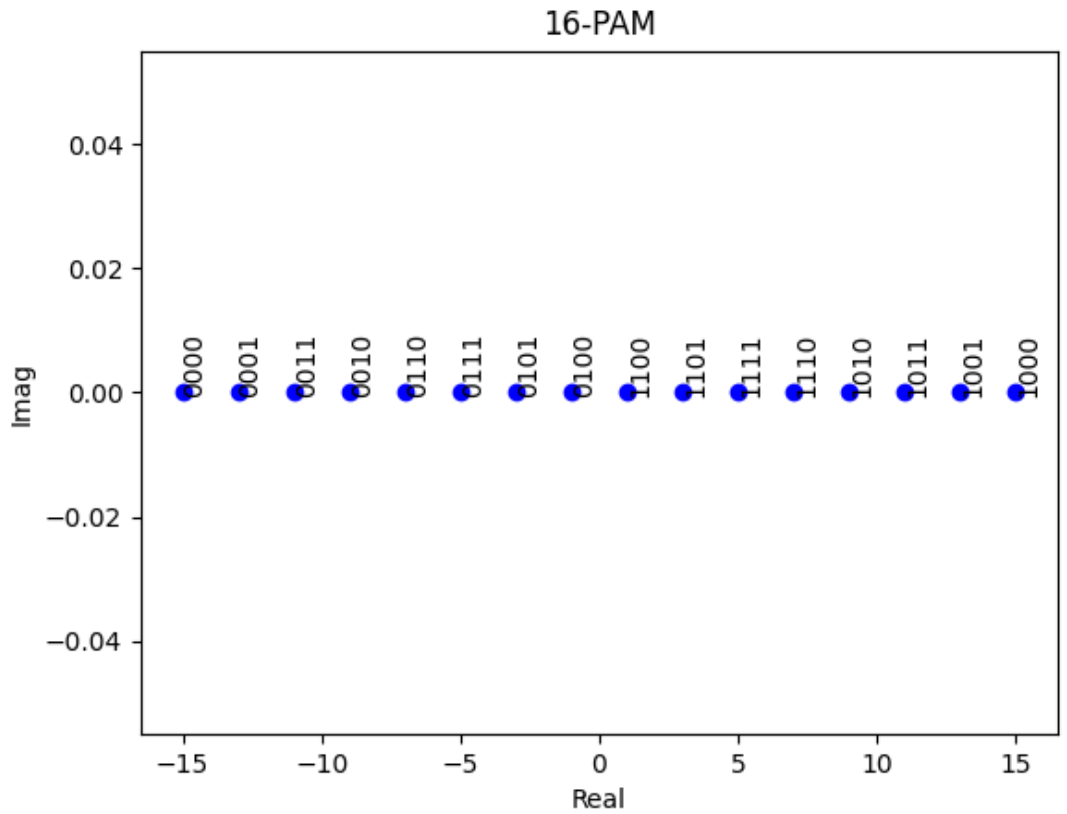
Methods:

- `__init__` : Δημιουργία του συστήματος
- `set_input` : Θέτουμε το σήμα εισόδου.
- `set_output` : Θέτουμε το σήμα εξόδου.
- `calc_transfer_function` : Υπολογίζουμε την συνάρτηση μεταφοράς από το Python callable `transfer_function` πάνω στον άξονα των συχνοτήτων του `input_signal`.
- `apply` : Υπολογίζουμε την έξοδο του συστήματος.

# Ένα παράδειγμα

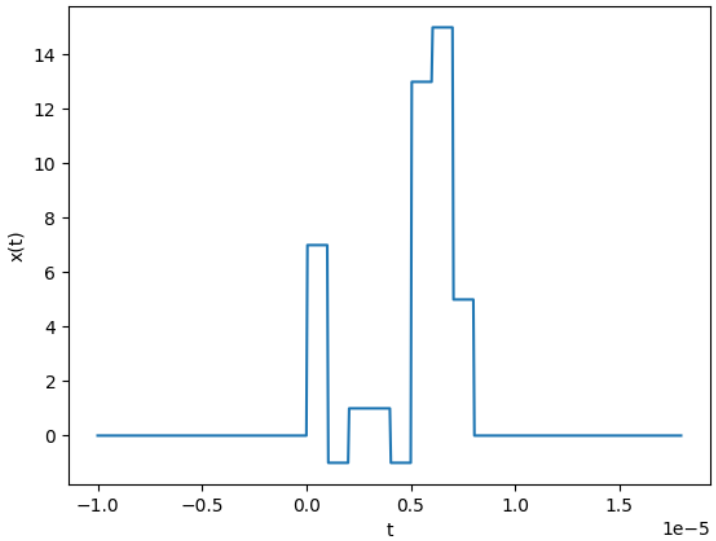
```
import matplotlib.pyplot as plt
import commlibv2 as cl
import numpy as np

# Parameters
TS = 1e—6
samples_per_symbol = 20
tinitial = 0
tguard = 10 * TS
f0 = 1 / TS

# system transfer function
H = lambda f : np.exp( —f**2.0 / f0 ** 2.0 / 2)

# Signal constellation
c = cl.pam_constellation(16, title = '16—PAM')

# Plot PAM constellation
plt.close('all')
c.plot()
c.plot_map()

# set bits to be transmitted
bits = cl.random_bits(32)

# build input waveform and plot
x = cl.digital_signal(TS = TS, samples_per_symbol
        = samples_per_symbol,
                      tinitial = tinitial, tguard
        = tguard, constellation = c)

x.modulate_from_bits( bits, constellation = c )
x.plot()

# create channel system and apply
s = cl.system(input_signal = x, transfer_function
        = H)
s.apply()

# get output signal and plot
y = s.get_output()
y.plot()
```

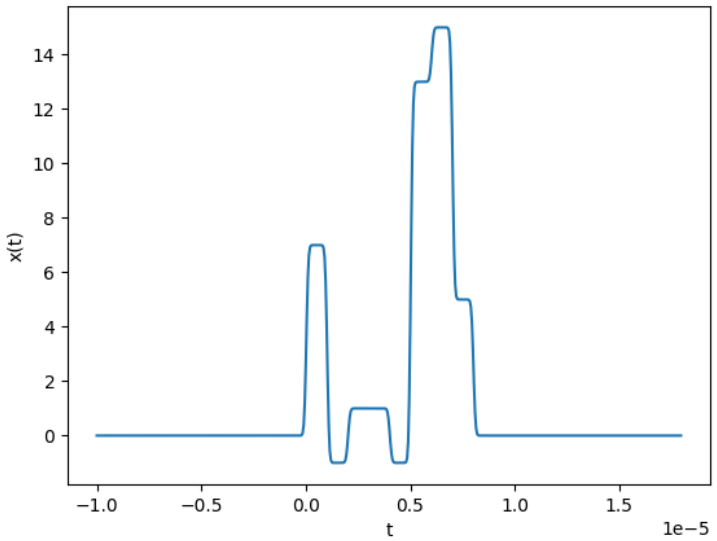
# Ένα παράδειγμα : Αστερισμός



# Ένα παράδειγμα : Είσοδος και έξοδος

$$f_0 = \frac{2}{T_S}$$



# Ένα παράδειγμα : Είσοδος και έξοδος

$$f_0 = \frac{1}{T_S}$$