Εκφώνηση

Να υλοποιήσετε πρόγραμμα λογισμικού, το οποίο να δίνει την ψηφιακή ακολουθία που αντιστοιχεί σε ένα σύνολο μετρήσεων, το οποίο λαμβάνεται από ένα νοητό σύστημα δειγματοληψίας.

Το πρόγραμμα θα πρέπει να δέχεται ως εισόδους:

- Το εύρος ζώνης του αναλογικού σήματος.
- Το πεδίο τιμών του αναλογικού σήματος (π.χ., σήμα το οποίο μπορεί να πάρει τιμές από 0 έως και 7 Volt, ή κάτι άλλο).
- Τον αριθμό των σταθμών κβαντοποίησης (π.χ., μπορεί να είναι 16 ή 32).
- Μια ληφθείσα ακολουθία μετρήσεων (π.χ., 0.78, 1.22, 2.36, 3.80, 4.04, 6.59, 6.99, …).

Έξοδος: Το πρόγραμμα θα πρέπει να:

- Εμφανίζει το Bit Rate το οποίο προκύπτει.
- Παράγει την ψηφιακή ακολουθία που αντιστοιχεί στο σύνολο των μετρήσεων που ελήφθησαν.

Λειτουργία (υποδείξεις):

- Οι στάθμες κβαντοποίησης να κατανέμονται (όσο το δυνατόν) ομοιόμορφα στο πεδίο τιμών, αρχίζοντας από την πρώτη και καταλήγοντας στην τελευταία τιμή του πεδίου τιμών του σήματος.
- Να γίνεται χειρισμός λανθασμένης εισαγωγής δεδομένων (π.χ., ακολουθία μετρήσεων εκτός του πεδίου τιμών).

Γλώσσα προγραμματισμού:

- Να χρησιμοποιηθεί μια γλώσσα προγραμματισμού εκ των Java ή C ή Python.

Παραδοτέα:

- Η εργασία θα παραδοθεί σε ηλεκτρονική μορφή και θα περιλαμβάνει:
    - Εκφώνηση της εργασίας
    - Σκεπτικό επίλυσης
    - Πηγαίο κώδικα
    - Ενδεικτικά αποτελέσματα
    - Εκτελέσιμο αρχείο εάν χρειάζεται

Σκεπτικό επίλυσης

Το πρόγραμμα βασίζεται σε τρία βασικά στάδια:

1. **Εισαγωγή και έλεγχος δεδομένων:**
    - Λήψη τιμών από το χρήστη.
    - Έλεγχος ορθότητας των δεδομένων (π.χ., το εύρος τιμών, δύναμη του 2 για στάθμες κβαντοποίησης).
2. **Κβαντοποίηση και κωδικοποίηση:**
    - Υπολογισμός βήματος κβαντοποίησης.
    - Αντιστοίχιση των μετρήσεων στις πλησιέστερες στάθμες.
    - Μετατροπή των κβαντοποιημένων τιμών σε δυαδική μορφή.
3. **Υπολογισμός και έξοδος των αποτελεσμάτων:**
    - Υπολογισμός Bit Rate με βάση τη συχνότητα Nyquist και τα bits ανά δείγμα.
    - Εμφάνιση των αποτελεσμάτων στον χρήστη.

Πηγαίος Κώδικας (Python)

```
import math


def is_power_of_two(n):
    """Ελέγχει αν ένας αριθμός είναι δύναμη του 2."""
    return n > 0 and (n & (n - 1)) == 0


def quantize_signal(measurements, min_val, max_val, quantization_levels):
    """Κβαντοποιεί τις μετρήσεις σύμφωνα με τις στάθμες κβαντοποίησης."""
    step_size = (max_val - min_val) / (quantization_levels - 1)  # Υπολογισμός βήματος
    quantized_values = [round((val - min_val) / step_size) for val in measurements]  # Αντιστοίχιση στις πλησιέστερες στάθμες
    return quantized_values


def encode_to_binary(quantized_values, quantization_levels):
    """Μετατρέπει τις κβαντοποιημένες τιμές σε δυαδική μορφή."""
    bits_per_sample = int(math.log2(quantization_levels))
    binary_sequence = [format(qv, f'0{bits_per_sample}b') for qv in quantized_values]
```

```
    return ''.join(binary_sequence)


# Λήψη εισόδων από τον χρήστη
f = int(input("Εισάγετε το εύρος ζώνης του αναλογικού σήματος (Hz): "))


volt_range = list(map(int, input("Εισάγετε το πεδίο τιμών (min max): ").split()))
while len(volt_range) != 2 or volt_range[0] >= volt_range[1]:
    print("Σφάλμα: Εισάγετε δύο αριθμούς, όπου ο πρώτος είναι μικρότερος από τον δεύτερο.")
    volt_range = list(map(int, input("Εισάγετε το πεδίο τιμών (min max): ").split()))


quantums = int(input("Εισάγετε τον αριθμό των σταθμών κβαντοποίησης (δύναμη του 2): "))
while not is_power_of_two(quantums):
    print("Σφάλμα: Ο αριθμός πρέπει να είναι δύναμη του 2 (2, 4, 8, 16, 32, ...).")
    quantums = int(input("Εισάγετε τον αριθμό των σταθμών κβαντοποίησης: "))


measurements = list(map(float, input("Εισάγετε τις μετρήσεις (χωρισμένες με κενά): ").split()))
while any(m < volt_range[0] or m > volt_range[1] for m in measurements):
    print(f"Σφάλμα: Όλες οι μετρήσεις πρέπει να είναι μεταξύ {volt_range[0]} και {volt_range[1]}.")
    measurements = list(map(float, input("Εισάγετε τις μετρήσεις (χωρισμένες με κενά): ").split()))


# Υπολογισμός παραμέτρων
fs = 2 * f  # Θεώρημα Nyquist
bits_per_sample = int(math.log2(quantums))
bit_rate = fs * bits_per_sample  # Υπολογισμός Bit Rate


# Κβαντοποίηση & Κωδικοποίηση
```

quantized_values = quantize_signal(measurements, volt_range[0], volt_range[1], quantums)

binary_sequence = encode_to_binary(quantized_values, quantums)

 # Εμφάνιση αποτελεσμάτων

print("\n **Αποτελέσματα ")**

print(f"Συχνότητα Δειγματοληψίας: {fs} Hz")

print(f"Bits ανά δείγμα: {bits_per_sample}")

print(f"Bit Rate: {bit_rate} bits/sec")

print(f"Ψηφιακή Ακολουθία: {binary_sequence}")

Ενδεικτικά Αποτελέσματα

Εισάγετε το εύρος ζώνης του αναλογικού σήματος (Hz): 20

Εισάγετε το πεδίο τιμών (min max): 0 7

Εισάγετε τον αριθμό των σταθμών κβαντοποίησης (δύναμη του 2): 8

Εισάγετε τις μετρήσεις (χωρισμένες με κενά): 5.5 6.9 6.5 4.4 3.7 3.9 2.5 1.6 6.4 0.4

🔹 **Αποτελέσματα** 🔹

Συχνότητα Δειγματοληψίας: 40 Hz

Bits ανά δείγμα: 3

Bit Rate: 120 bits/sec

Ψηφιακή Ακολουθία: 110111110100100100010010110000