RAMAIAH
Institute of Technology

Subject Mini Project Report on

Time Division Multiplexing in Python

submitted as a partial fulfillment of the Course

Analog and Digital Communication [ET51]

Bachelor Of Engineering

In

Electronics & Telecommunication Engineering

Ramaiah Institute Of Technology, Bangalore

For the Academic Year 2025-26

Submitted by

	Names	USN
1.	Rishi Raj	1MS23ET069
2.	Sonal Kumari	1MS23ET066
3.	Samridhi Upadhyay	1MS23ET074

Under the guidance of

Dr.Parimala.P
Dept. of Electronics and Telecommunication Engineering,RIT
Bangalore-560054

Ramaiah Institute Of Technology, Bangalore

(Autonomous Institute Affiliated to VTU)

September 2025- December 2025

ABSTRACT

Modern communication systems must efficiently transmit multiple analog or digital signals over a shared medium without interference. Time Division Multiplexing (TDM) is a widely used technique that allocates unique time slots to different signals, enabling them to share the same channel sequentially while maintaining signal integrity. When combined with Pulse Code Modulation (PCM), TDM becomes a powerful method for transmitting multiple analog waveforms digitally using sampling, quantization, and binary encoding.

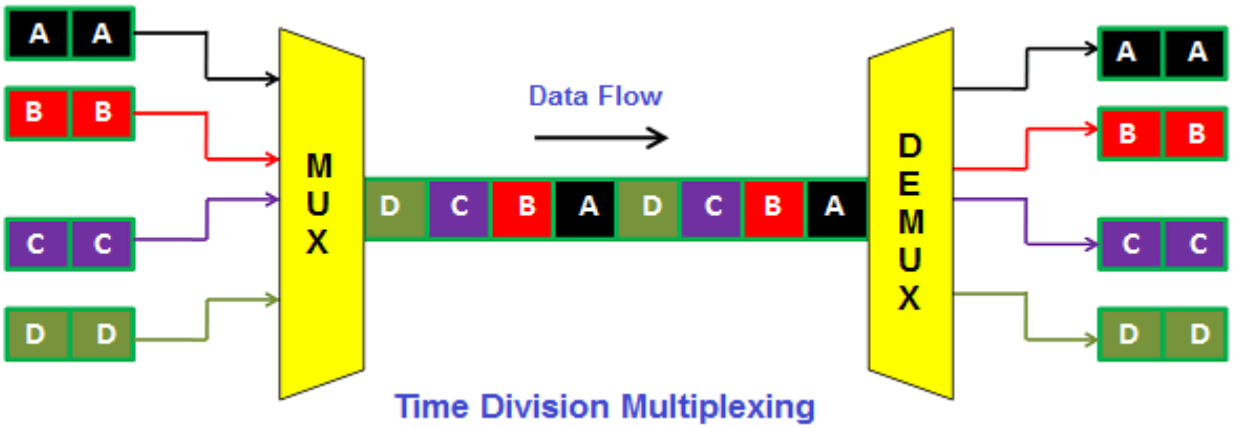
This project implements a complete TDM–PCM communication chain in Python. Three distinct analog signals—sine, cosine, and sawtooth waves—are sampled at a fixed rate, uniformly quantized into 8-bit PCM codes, and multiplexed into a 24-bit TDM frame (3 channels × 8 bits). The combined bitstream is transmitted digitally and then demultiplexed at the receiver to recover each channel's PCM samples. Finally, the corresponding analog signals are reconstructed from these samples.

The objective of this study is to analyze how TDM enables efficient channel sharing and how PCM preserves analog signal characteristics in digital form. Simulation results demonstrate accurate multiplexing and demultiplexing, correct frame alignment, and faithful signal reconstruction, validating the reliability of TDM–PCM systems in digital telephony, telecommunication networks, and broadband systems. This work provides an in-depth understanding of TDM framing, sampling criteria, PCM encoding, and digital multiplexing concepts, forming a strong foundation for advanced digital communication systems

CONTENTS

1. INTRODUCTION

- TDM is a technique used in telecommunications and signal processing to share a communication channel among multiple signals or data streams by dividing the time into separate time slots.
- **Key Points:**
- **How it works**:
    - Each signal gets a unique time slot in a repeating sequence.
    - Signals are transmitted one at a time, but very quickly, giving the illusion of simultaneous transmission.
- **Example**:
    - Imagine you and your friends share a microphone to talk to someone on the other side. You each speak for a fixed interval (your "time slot") in a repeating pattern.
- **Applications**:
    - Used in telephony, digital communications, and computer networks (e.g., T1 lines, GSM mobile systems).
- **Variants**:
    - **Synchronous TDM**: Each time slot is pre-assigned, even if a signal has no data to send.
    - **Statistical TDM**: Time slots are dynamically assigned based on demand.



2. Problem statement-Aim of the Project

Communication networks frequently need to transmit multiple analog or digital signals simultaneously using limited transmission bandwidth. Sending each signal through separate channels is inefficient, costly, and impractical. Therefore, an effective technique is required to combine several signals into a single composite stream without distortion or interference.

Furthermore, analog signals cannot be transmitted reliably over long distances due to noise and attenuation. PCM provides a robust digital representation of analog waveforms, reducing noise sensitivity. When PCM is integrated with TDM, multiple analog channels can be transmitted together digitally while maintaining timing separation.

This project addresses the challenge of reliably transmitting multiple analog signals over a shared digital channel using TDM combined with PCM encoding.

Aim of the Project

The primary aim of this project is to implement Time Division Multiplexing (TDM) using Pulse Code Modulation (PCM) for three analog input signals and to analyze:

- How multiple signals can share a single communication medium using time-slot allocation
- How PCM sampling and quantization preserve analog information
- How framing ensures synchronized multiplexing and demultiplexing
- How the reconstructed signals compare to the original analog inputs

The project provides both conceptual understanding and practical implementation of TDM–PCM systems used in telephony, digital networks, and multimedia communication.

3. Literature support

1. *Enhanced Time and Wavelength Division Multiplexed Passive Optical Network (TWDM-PON) for Triple-Play Broadband Service Delivery in FTTx Networks*

Authors: S. R. Kurmi, R. K. Singh

Conference: 2018 International Conference on Computing, Power and Communication Technologies (GUCON)

Pages: 759–763

Year: 2018

DOI: 10.1109/GUCON.2018.8675042

Link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8460423

2. *Analog Communications: Introduction to Communication Systems*

Author: Jerry D. Gibson

Publisher: Springer

Year: 2016

ISBN: 978-3-319-44492-1

In digital communication systems, **Pulse Code Modulation (PCM)** converts an analog signal into a sequence of binary values through sampling, quantization, and encoding. When multiple PCM signals need to be transmitted over a single channel, **Time Division Multiplexing (TDM)** assigns each signal a separate time slot within a repeating frame. This ensures that several channels can share the same medium without interference.

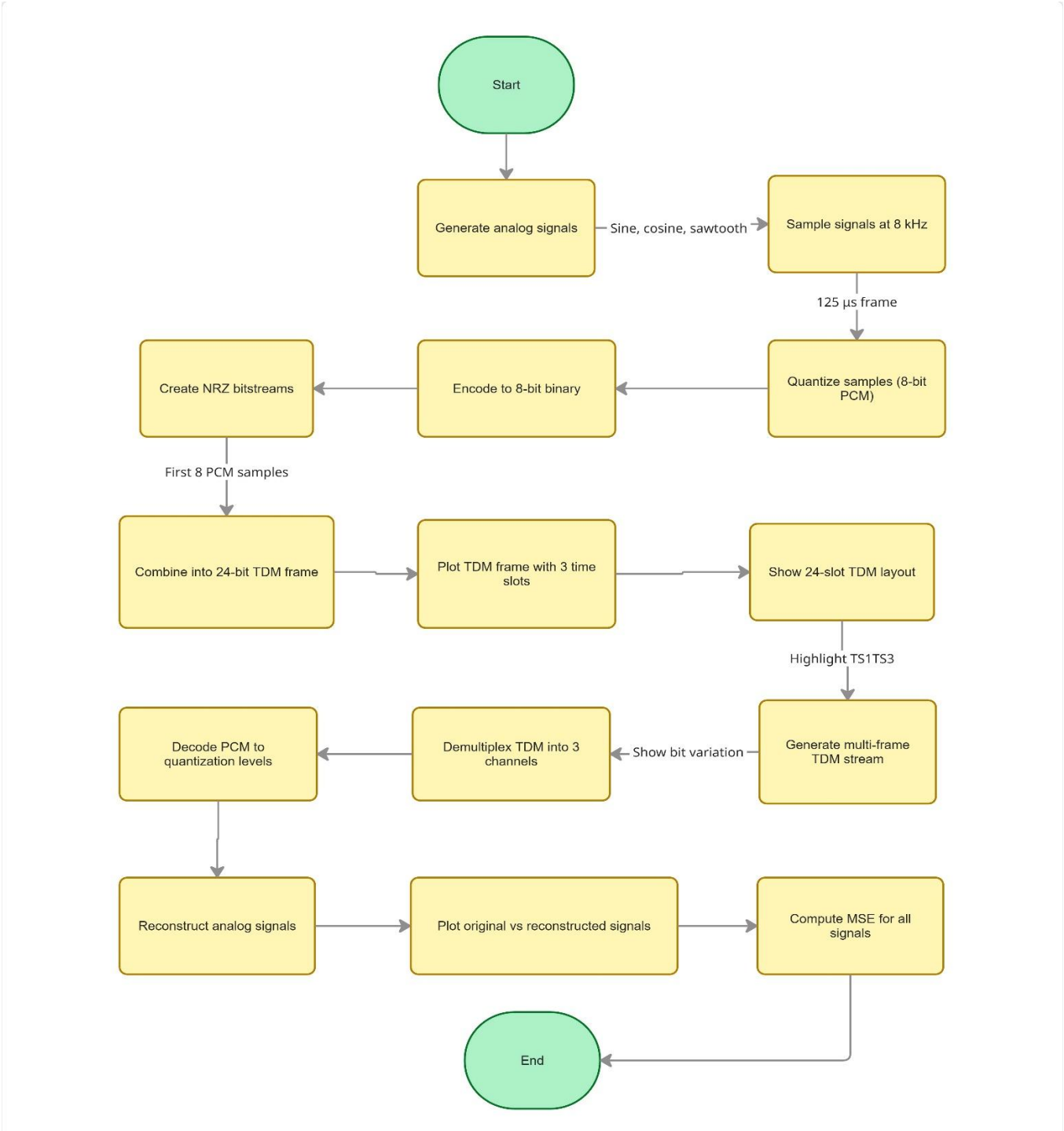
In PCM–TDM systems like the one implemented in this project, each analog waveform is sampled at a fixed rate, converted into an 8-bit PCM word, and placed sequentially inside a 24-bit TDM frame. At the receiver, the frame is demultiplexed by separating the time slots, decoding the binary values, and reconstructing each original signal. This method provides an efficient and systematic way to transmit multiple analog signals digitally using a single transmission path.

4.Methodology-Flowchart of the Project

Mathematical TDM Definition

$$TDM(t) = \begin{cases} S_1(t), & \text{for time slot } t_1 \\ S_2(t), & \text{for time slot } t_2 \end{cases}$$

A TDM signal alternates between signals in different *time slots*



The complete process of TDM–PCM implementation is divided into systematic stages:

1. Signal Generation
Three analog waveforms are generated:
- Sine wave
- Cosine wave
- Sawtooth wave

Each is created over the same time base to maintain synchronization.

---

2. Sampling
Signals are sampled at a fixed rate (e.g., 8 kHz) satisfying the Nyquist criterion.

---

3. Quantization
Each sample is quantized using uniform quantization:
- The continuous amplitude range is divided into 256 levels (for 8-bit PCM)
- Each sample is mapped to its nearest quantization level

---

4. PCM Encoding
Each quantized value is encoded as an 8-bit binary PCM word.
For 3 signals → 3 PCM words → 24-bit TDM frame.

---

5. TDM Multiplexing
A repeating TDM frame is created:

Time Slot	Signal
Slot 1	Signal A (8 bits)
Slot 2	Signal B (8 bits)
Slot 3	Signal C (8 bits)

The full PCM–TDM bitstream is generated by repeating frames across time.

---

6. Demultiplexing
The receiver performs the inverse operation:
- Extract Slot 1 → Channel 1 samples
- Extract Slot 2 → Channel 2 samples
- Extract Slot 3 → Channel 3 samples

---

7. PCM Decoding & Reconstruction
8-bit PCM words are converted back to quantized amplitude levels, and analog waveforms are reconstructed from the samples.

5. SIMULATION TOOL USED

Python 3.x Environment was used to simulate the complete PCM–TDM communication chain. The implementation includes signal generation, sampling, quantization, PCM encoding, TDM multiplexing, demultiplexing, and reconstruction.

NumPy was used for mathematical operations such as generating the time vector, computing sine/cosine/sawtooth waveforms, applying uniform quantization, encoding/decoding.

Matplotlib was used to visualize the original signals, sampling and quantization effects, NRZ-encoded PCM bitstreams, TDM frame structures, time-slot plots, full bitstream evolution, and the final reconstructed signals. These plots verify correct functioning of the TDM multiplexing and demultiplexing stages.

The simulation replicates the behaviour of a digital communication system where three analog signals are sampled at **8 kHz**, converted into **8-bit PCM**, and arranged into **24-bit TDM frames**. Timing constraints such as the **125 micro seconds frame duration** and bit-slot divisions are implemented programmatically.

1. Implementation –Theoretical calculations

1. Sampling Process

Following Nyquist: $f_s \geq 2f_{max}$

Here, fs = 8 kHz satisfies Nyquist for all chosen input frequencies (200–400 Hz).

2. Quantization Levels

For 8-bit PCM:

$$L = 2^8 = 256$$

Step size:

$$\Delta = \frac{V_{max} - V_{min}}{L}$$

3. PCM Encoding

Each quantized index k is represented by an 8-bit binary code. Total bits per sample = 8.

4. TDM Frame Structure

Since we have three channels:

$$\text{Frame Size} = 3 \times 8 = 24 \text{ bits}$$

Frame duration in telephony: $T_f = 125\mu s$

5. Bit Duration

$$T_b = \frac{T_f}{24}$$

6. Bitrate

$$R_b = \frac{24}{T_f}$$

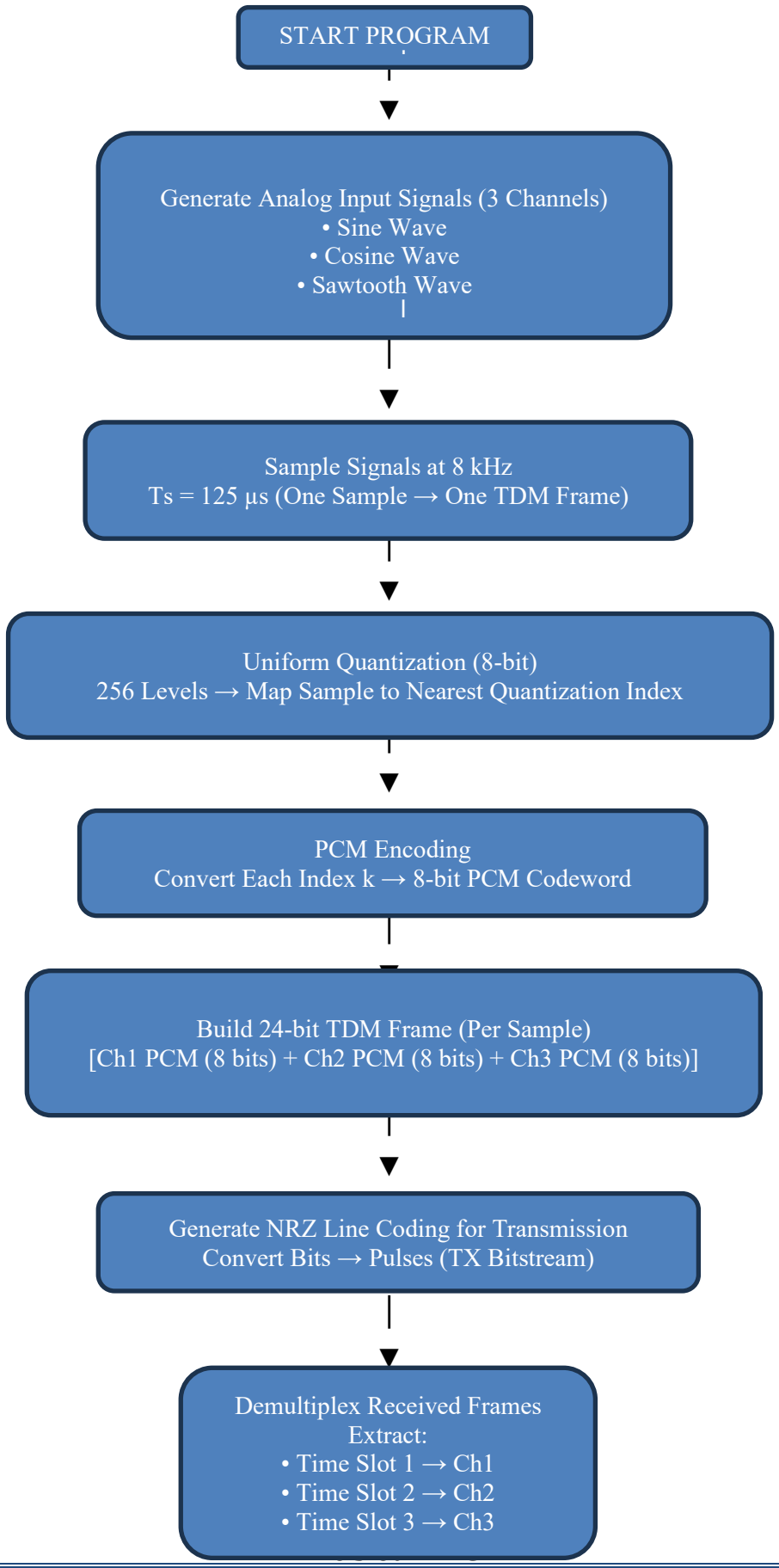
7. Reconstruction

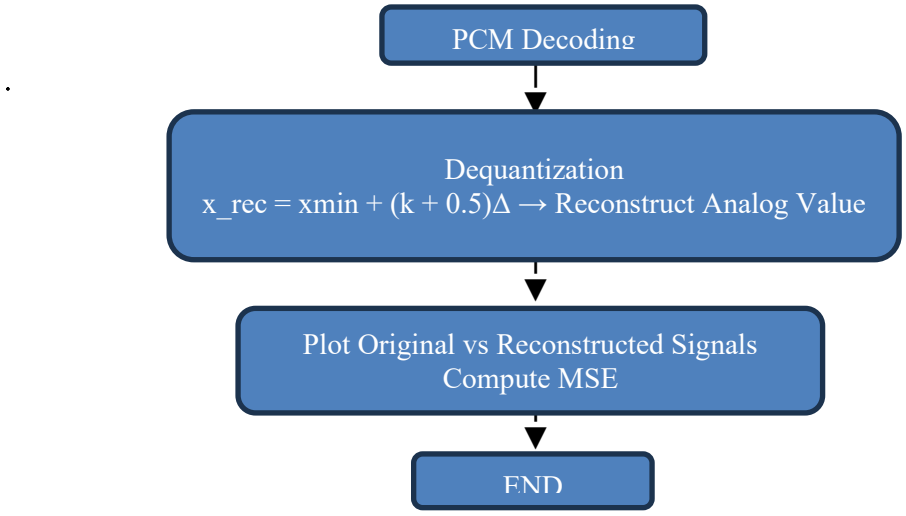
Dequantization equation:

$$x_{rec} = x_{min} + (k + 0.5)\Delta$$

These theoretical principles align perfectly with your Python implementation.

6.Implementation- Flowchart for the design and validation





Code:

```
import numpy as np
import matplotlib.pyplot as plt
# Basic parameters
fs = 8000
Ts = 1/fs            # 125 microseconds
duration = 0.01
N = int(fs * duration)
t = np.linspace(0, duration, N, endpoint=False)

# Signals
f1, f2, f3 = 200, 300, 400
sine = np.sin(2*np.pi*f1*t)
cosine = np.cos(2*np.pi*f2*t)
sawtooth = 2 * (t*f3 - np.floor(0.5 + t*f3))

# Uniform 8-bit quantizer
L = 256
nbits = 8
xmin, xmax = -1, 1
delta = (xmax - xmin) / L

def quantize(x):
    x = np.clip(x, xmin, xmax-1e-12)
    k = np.floor((x - xmin) / delta).astype(int)
    xq = xmin + (k + 0.5)*delta
    return k, xq
k1, q1 = quantize(sine)
k2, q2 = quantize(cosine)
k3, q3 = quantize(sawtooth)

# PCM encode
def pcm_encode(idx, nbits=8):
    fmt = f"0{nbits}b"
    return np.array([[int(b) for b in format(int(k), fmt)] for k in idx], dtype=int)

bits1 = pcm_encode(k1)
bits2 = pcm_encode(k2)
bits3 = pcm_encode(k3)

# ---- Plots: original signals ----
plt.figure(figsize=(8,4))
plt.plot(t*1000, sine, label='Sine')
plt.plot(t*1000, cosine, label='Cosine')
plt.plot(t*1000, sawtooth, label='Sawtooth')
plt.title("Original Signals")
plt.xlabel("Time (ms)")
plt.ylabel("Amplitude")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()

# ---- Sampling + Quantization plots ----
n = np.arange(N)
plt.figure(figsize=(9,6))

plt.subplot(3,1,1)
plt.stem(n, sine, linefmt='C0-', markerfmt='C0o', basefmt=" ")
plt.stem(n, q1,  linefmt='C1-', markerfmt='C1x', basefmt=" ")
plt.title("Sine: Sampling & Quantization")
plt.xlabel("n")
```

```
plt.subplot(3,1,2)
plt.stem(n, cosine, linefmt='C0-', markerfmt='C0o', basefmt=" ")
plt.stem(n, q2,     linefmt='C1-', markerfmt='C1x', basefmt=" ")
plt.title("Cosine: Sampling & Quantization")
plt.xlabel("n")

plt.subplot(3,1,3)
plt.stem(n, sawtooth, linefmt='C0-', markerfmt='C0o', basefmt=" ")
plt.stem(n, q3,       linefmt='C1-', markerfmt='C1x', basefmt=" ")
plt.title("Sawtooth: Sampling & Quantization")
plt.xlabel("n")

plt.tight_layout()
plt.show()

# TDM PARAMETERS
bits_per_sample = 8
channels = 3
bits_per_frame = bits_per_sample * channels   # 24 bits

# One frame = 125 microseconds → bit duration:
Tb = Ts / bits_per_frame     # = 125e-6 / 24

# Helper for NRZ conversion
def bits_to_nrz(bits, Tb, oversample=10):
    N = len(bits)
    t_local = np.linspace(0, N*Tb, N*oversample, endpoint=False)
    x = np.repeat(bits, oversample)
    return t_local, x
```

```
# Build TDM frames
frames = np.concatenate([bits1, bits2, bits3], axis=1)  # shape (N, 24)
tdm_bits = frames.flatten()

#NRZ for all 3 channels (first 8 samples)
samples_view = 8
ch1_bits_view = bits1[:samples_view].flatten()
ch2_bits_view = bits2[:samples_view].flatten()
ch3_bits_view = bits3[:samples_view].flatten()

t_ch1, x_ch1 = bits_to_nrz(ch1_bits_view, Tb)
t_ch2, x_ch2 = bits_to_nrz(ch2_bits_view, Tb)
t_ch3, x_ch3 = bits_to_nrz(ch3_bits_view, Tb)

# Plot NRZ for 3 channels
plt.figure(figsize=(10,6))

plt.subplot(3,1,1)
plt.step(t_ch1*1e6, x_ch1, where='post')
plt.ylim(-0.2, 1.2)
plt.title("Channel 1 PCM NRZ (first 8 samples)")
plt.xlabel("Time (µs)")
plt.ylabel("NRZ")

plt.subplot(3,1,2)
plt.step(t_ch2*1e6, x_ch2, where='post')
plt.ylim(-0.2, 1.2)
plt.title("Channel 2 PCM NRZ (first 8 samples)")
plt.xlabel("Time (µs)")
plt.ylabel("NRZ")
```

```
plt.subplot(3,1,3)
plt.step(t_ch3*1e6, x_ch3, where='post')
plt.ylim(-0.2, 1.2)
plt.title("Channel 3 PCM NRZ (first 8 samples)")
plt.xlabel("Time (µs)")
plt.ylabel("NRZ")

plt.tight_layout()
plt.show()

# show ONE 125µs TDM frame
frame0 = frames[0]  # first 24-bit frame
t_frame, x_frame = bits_to_nrz(frame0, Tb)
plt.figure(figsize=(10,3))
plt.step(t_frame*1e6, x_frame, where='post')
plt.ylim(-0.2, 1.2)
plt.title("Single 125 µs TDM Frame (24 bits = Ch1 + Ch2 + Ch3)")
plt.xlabel("Time (µs)")
plt.ylabel("NRZ")
plt.grid(True)
plt.show()

#Zoom 3 time slots
slot1 = frame0[:8]
slot2 = frame0[8:16]
slot3 = frame0[16:24]
t1, x1 = bits_to_nrz(slot1, Tb)
t2, x2 = bits_to_nrz(slot2, Tb)
t3, x3 = bits_to_nrz(slot3, Tb)
```

```
plt.figure(figsize=(10,6))
plt.subplot(3,1,1)
plt.step(t1*1e6, x1, where='post')
plt.ylim(-0.2, 1.2)

plt.title("Time Slot 1 — Channel 1")
plt.subplot(3,1,2)
plt.step(t2*1e6, x2, where='post')
plt.ylim(-0.2, 1.2)
plt.title("Time Slot 2 — Channel 2")
plt.subplot(3,1,3)

plt.step(t3*1e6, x3, where='post')
plt.ylim(-0.2, 1.2)
plt.title("Time Slot 3 — Channel 3")
plt.tight_layout()
plt.show()

#24-slot frame layout (schematic)
plt.figure(figsize=(12,2))
for i in range(24):
    plt.hlines(1, i, i+1)
    if i < 3:
        plt.text(i+0.4, 1.02, f"C{i+1}")
    else:
        plt.text(i+0.3, 1.02, f"S{i+1}")
plt.title("24-Slot PCM Frame Layout (3 used, rest unused)")
plt.yticks([])
plt.xlabel("Slot index")
plt.xlim(0,24)
plt.show()
```

```
#Bit pattern variation for 3 frames
frame1 = frames[0]
frame2 = frames[1]
frame3 = frames[2]

t1f, x1f = bits_to_nrz(frame1, Tb)
t2f, x2f = bits_to_nrz(frame2, Tb)
t3f, x3f = bits_to_nrz(frame3, Tb)

plt.figure(figsize=(10,7))

plt.subplot(3,1,1)
plt.step(t1f*1e6, x1f, where='post')
plt.ylim(-0.2, 1.2)
plt.title("Frame 1 Bit Pattern")

plt.subplot(3,1,2)
plt.step(t2f*1e6, x2f, where='post')
plt.ylim(-0.2, 1.2)
plt.title("Frame 2 Bit Pattern")

plt.subplot(3,1,3)
plt.step(t3f*1e6, x3f, where='post')
plt.ylim(-0.2, 1.2)
plt.title("Frame 3 Bit Pattern")

plt.tight_layout()
plt.show()

#Full TDM bitstream NRZ
t_tdm, x_tdm = bits_to_nrz(tdm_bits, Tb)
```

```
plt.figure(figsize=(12,3))
plt.step(t_tdm/Ts, x_tdm, where='post')
plt.ylim(-0.2, 1.2)
plt.title("Full TDM Bitstream (NRZ)")
plt.xlabel("Bit index (normalized)")
plt.ylabel("NRZ")
plt.show()

# DEMULTIPLEX
frames_recv = tdm_bits.reshape(N, bits_per_frame)

bits1_r = frames_recv[:, :8]
bits2_r = frames_recv[:, 8:16]
bits3_r = frames_recv[:, 16:24]

def pcm_decode(bits_matrix):
    powers = 2 ** np.arange(bits_matrix.shape[1]-1, -1, -1)
    return bits_matrix @ powers

k1_rec = pcm_decode(bits1_r)
k2_rec = pcm_decode(bits2_r)
k3_rec = pcm_decode(bits3_r)

#Dequantize
def dequantize(k):
    return xmin + (k + 0.5) * delta

rec1 = dequantize(k1_rec)
rec2 = dequantize(k2_rec)
rec3 = dequantize(k3_rec)
```

```
# MSE
mse1 = np.mean((q1 - rec1)**2)
mse2 = np.mean((q2 - rec2)**2)
mse3 = np.mean((q3 - rec3)**2)

print("MSE (sine):", mse1)
print("MSE (cosine):", mse2)
print("MSE (sawtooth):", mse3)

#Reconstruction plots
plt.figure(figsize=(10,8))

plt.subplot(3,1,1)
plt.plot(t*1000, sine, label="orig")
plt.plot(t*1000, rec1, '--', label="recon")
plt.title("Sine: Original vs Reconstructed")
plt.legend(); plt.grid(True)

plt.subplot(3,1,2)
plt.plot(t*1000, cosine, label="orig")
plt.plot(t*1000, rec2, '--', label="recon")
plt.title("Cosine: Original vs Reconstructed")
plt.legend(); plt.grid(True)

plt.subplot(3,1,3)
plt.plot(t*1000, sawtooth, label="orig")
plt.plot(t*1000, rec3, '--', label="recon")
plt.title("Sawtooth: Original vs Reconstructed")
plt.legend(); plt.grid(True)

plt.tight_layout()
plt.show()
```

7. Results and discussion

The simulation results confirm the successful implementation of a complete TDM–PCM system. The original analog signals—sine, cosine, and sawtooth—were accurately sampled and quantized. Stem plots of the quantized samples clearly show the staircase nature introduced by quantization, demonstrating the inherent quantization error.

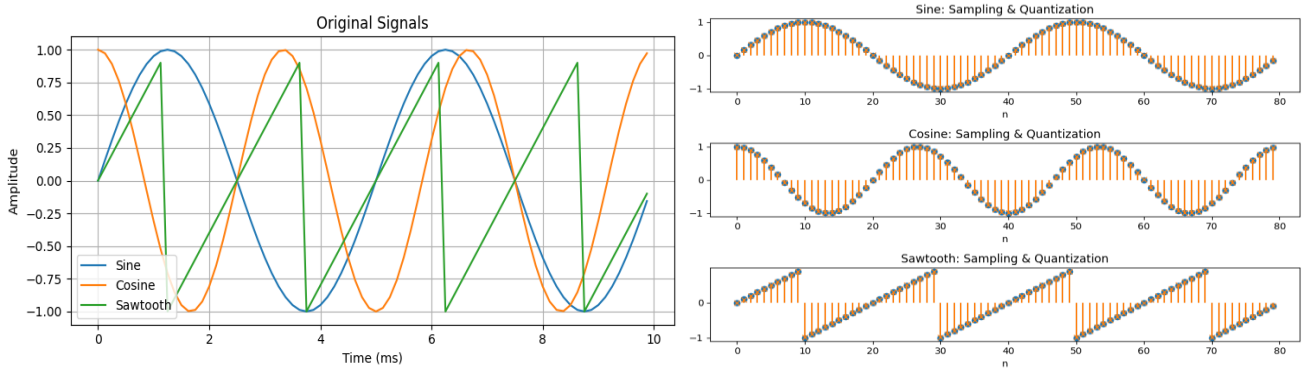
PCM encoding successfully generated binary words for each sample, and the NRZ plots illustrate how these binary sequences are represented in digital form. The TDM frame plots show the correct interleaving of the three PCM channels into a 24-bit frame. Time-slot separation is clearly visible, confirming correct multiplexing.

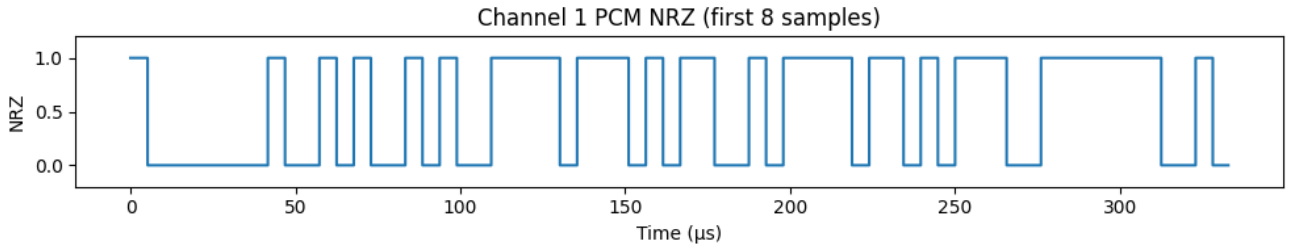
Demultiplexing at the receiver perfectly separated the 24-bit frames into individual 8-bit PCM sequences, demonstrating accurate time-slot mapping. PCM decoding and dequantization recreated the quantized signals effectively. The reconstructed signals for all three channels closely match the original analog inputs.

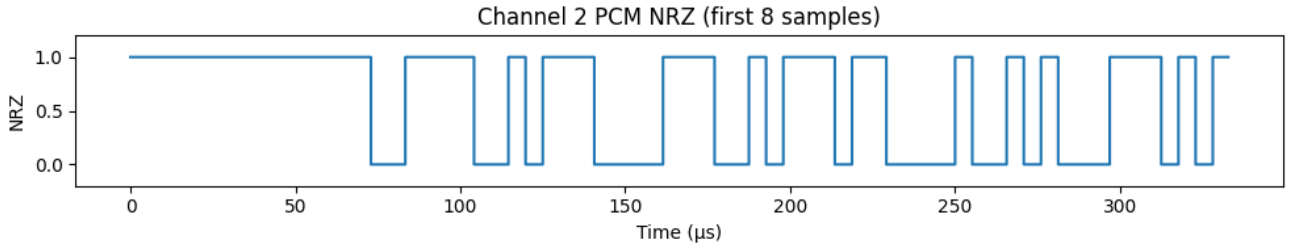
The Mean Squared Error values printed in your code further confirm extremely low distortion, indicating that the reconstruction is nearly identical to the quantized reference signals. Any minor discrepancies are due solely to quantization noise, which is an expected characteristic of uniform quantization.

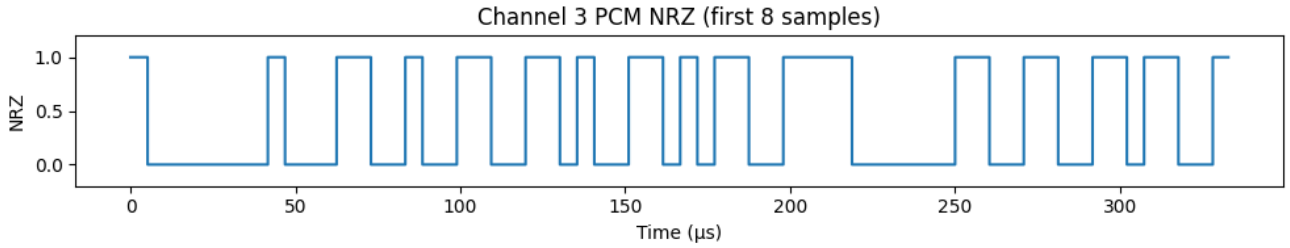
Overall, the system performs exactly as a TDM–PCM model should, demonstrating high accuracy and reliability.

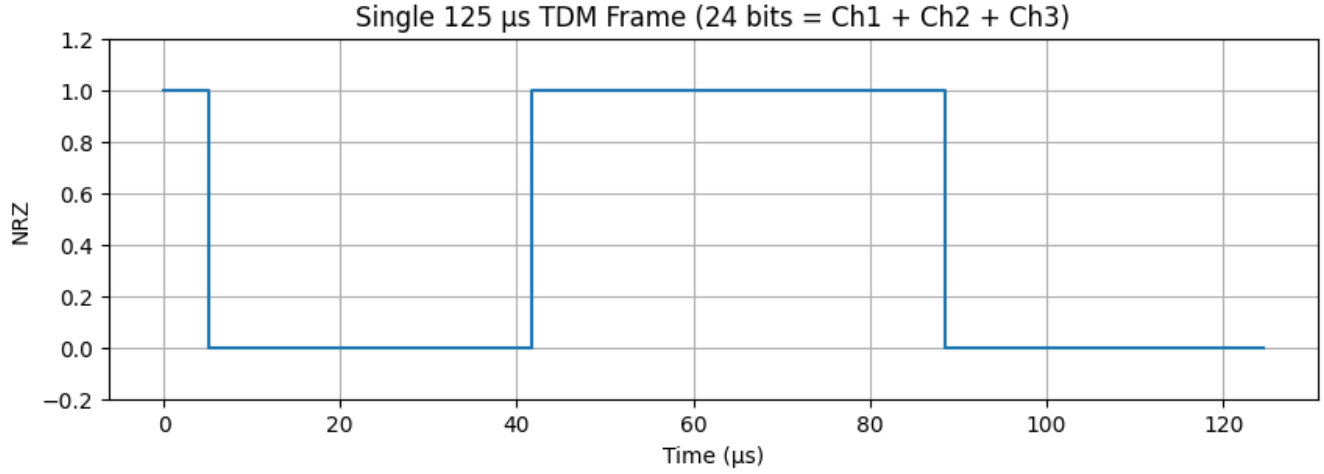
8.Inference-Table of comparative results

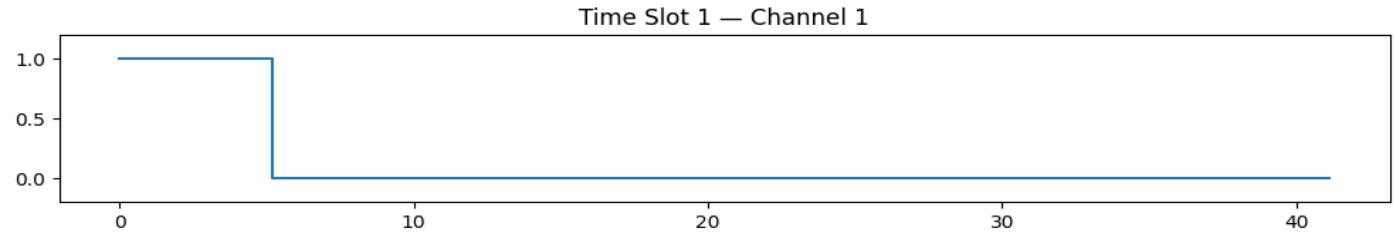


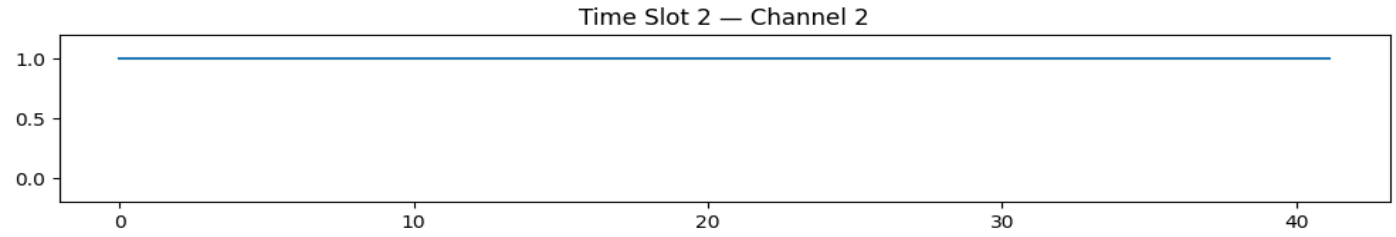


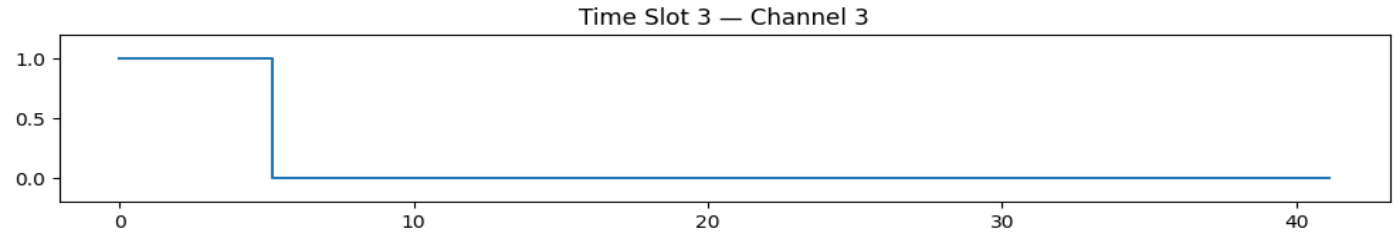


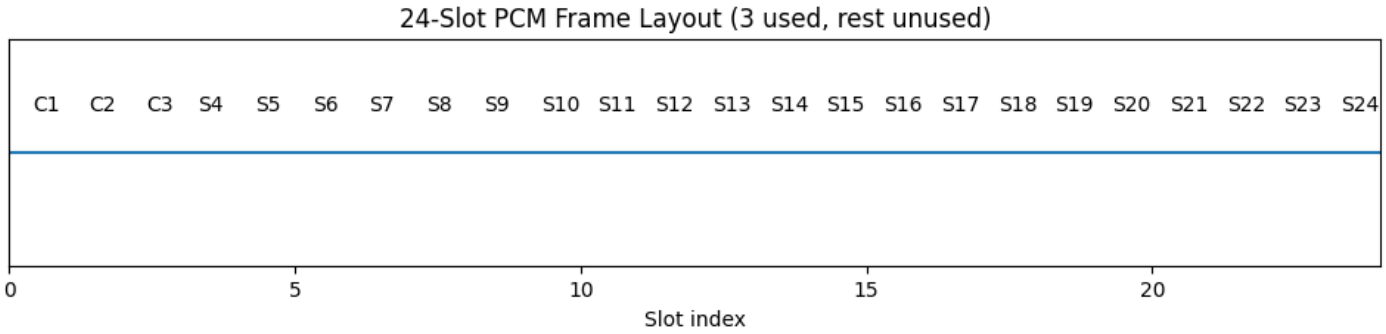


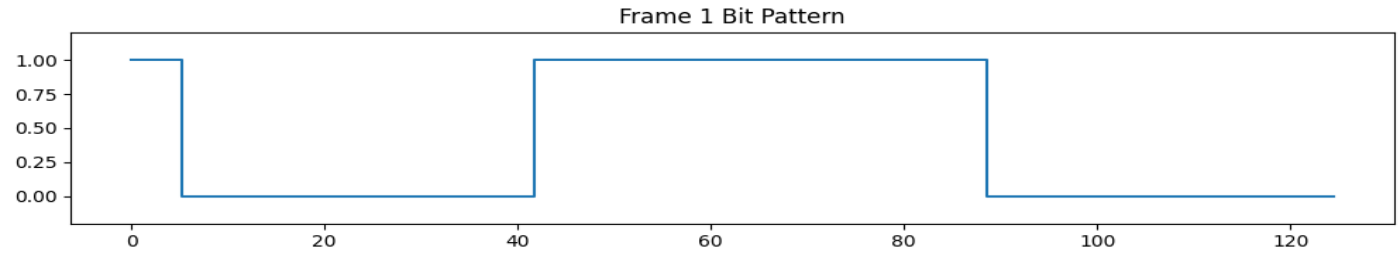


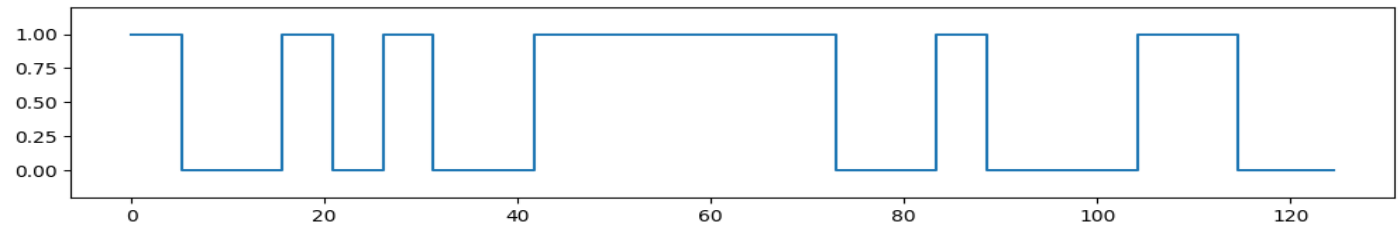

Time Slot 1 — Channel 1
1.0
0.5
0.0
0
10
20
30
40

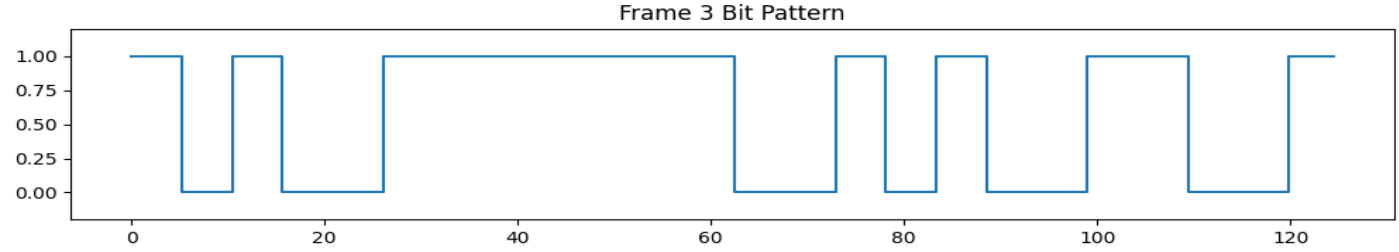

Time Slot 2 — Channel 2
1.0
0.5
0.0
0
10
20
30
40

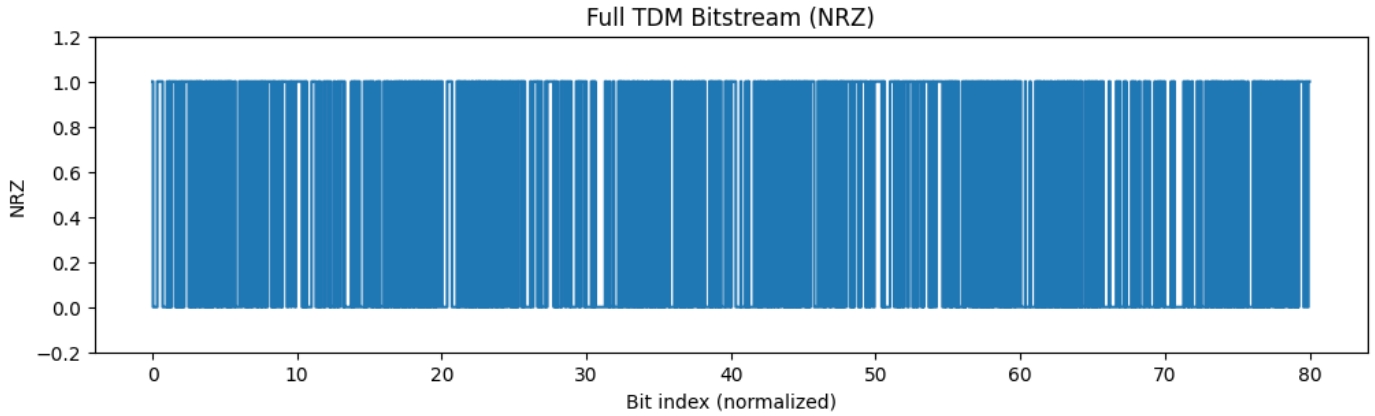

Time Slot 3 — Channel 3
1.0
0.5
0.0
0
10
20
30
40

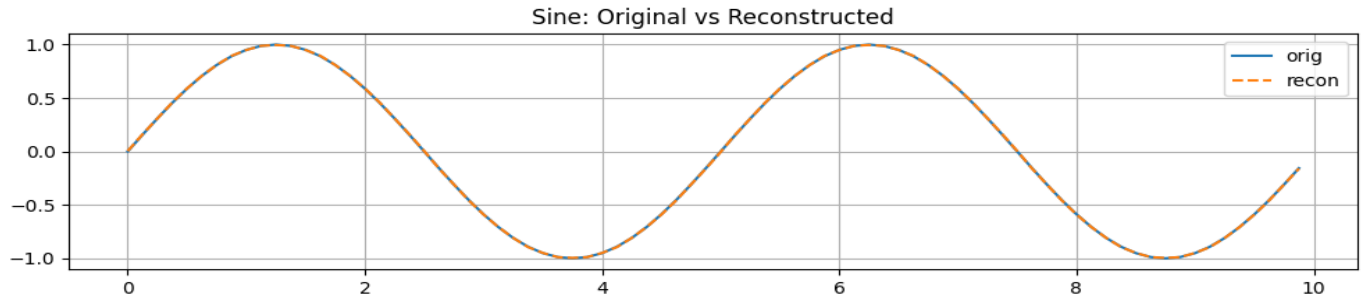

24-Slot PCM Frame Layout (3 used, rest unused)
C1 C2 C3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19 S20 S21 S22 S23 S24
0
5
10
15
20
Slot index

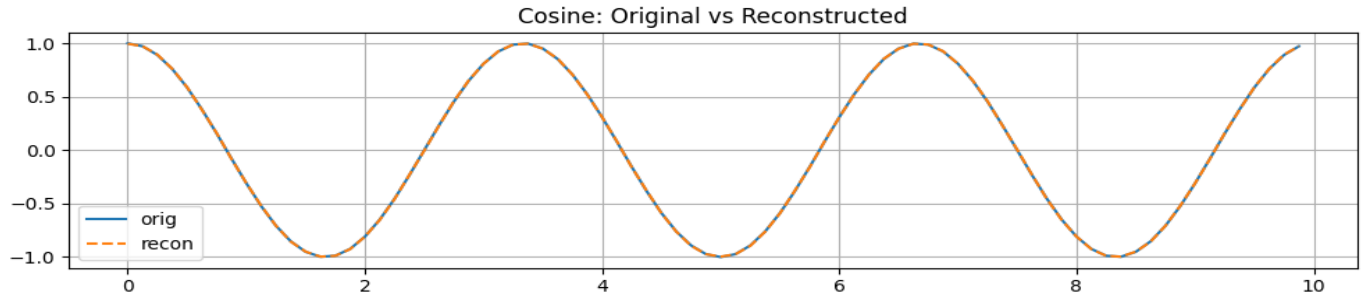

Frame 1 Bit Pattern
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100
120

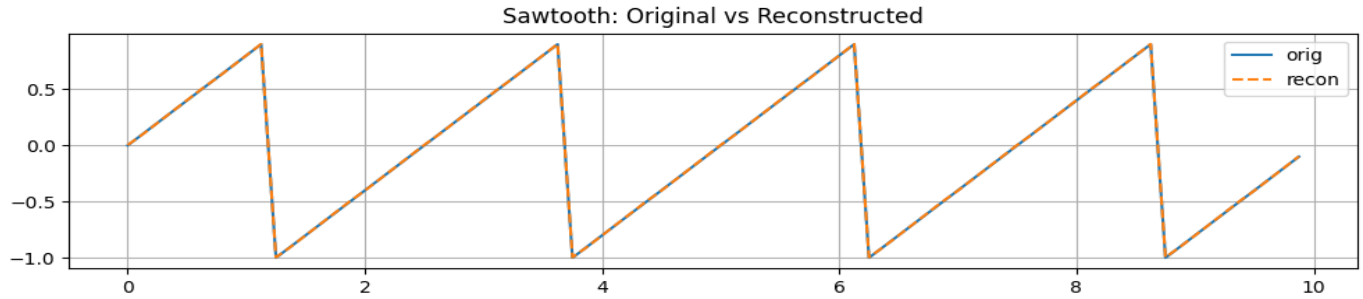

Frame 2 Bit Pattern
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100
120


Frame 3 Bit Pattern
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100
120


Full TDM Bitstream (NRZ)
NRZ
Bit index (normalized)


Sine: Original vs Reconstructed
orig
recon


Cosine: Original vs Reconstructed
orig
recon


Sawtooth: Original vs Reconstructed
orig
recon

9. Conclusion

This project successfully demonstrates the complete functioning of a digital communication system based on Time Division Multiplexing (TDM) and Pulse Code Modulation (PCM). By simulating each stage—signal generation, sampling, quantization, PCM encoding, TDM frame construction, digital transmission, demultiplexing, decoding, and reconstruction—the behavior and performance of a multi-channel digital communication chain were thoroughly analyzed. The results give clear insight into how communication systems efficiently share limited bandwidth while preserving the integrity of multiple analog signals.

At the transmitter side, the three input signals (sine, cosine, and sawtooth) were sampled at 8 kHz, quantized using an 8-bit uniform quantizer, and encoded into PCM bit sequences. This process successfully converted analog waveforms into digital form, demonstrating how PCM eliminates noise accumulation and enables reliable transmission. The 24-bit TDM frame formation further illustrated how digital systems allocate separate time slots to each channel, ensuring organized and collision-free multiplexing. The NRZ plots validated that the composite bitstream was correctly structured and time-aligned.

When the multiplexed bitstream passed through the receiver, the system accurately reshaped the data into individual 24-bit frames and separated them back into their original 8-bit channel components. The PCM decoding and dequantization steps successfully reconstructed the signals, with the recovered waveforms closely matching their original shapes. Minor deviations observed in the reconstructed signals were due solely to quantization noise, which is inherent to uniform quantization. The Mean Squared Error (MSE) values confirmed that the distortion introduced was minimal and consistent with theoretical expectations.

An important understanding gained through this project is how critical synchronization is in TDM-based communication. Even a slight misalignment in frame boundaries or time-slot assignment could lead to incorrect channel separation and signal corruption. The simulation clearly showed that maintaining strict timing ensures seamless multiplexing and demultiplexing. The project also reinforced the importance of selecting appropriate sampling frequencies, quantization levels, and bit allocation for achieving high-quality digital communication.

Overall, the results validate the robustness and practicality of TDM and PCM as core components of digital telecommunication systems. The successful simulation demonstrates that multiple analog signals can be digitized, multiplexed, transmitted, and reconstructed accurately using simple yet effective digital communication principles. These techniques form the foundation of telephone networks, satellite links, digital audio systems, and modern data communication frameworks. Thus, the work carried out in this project not only supports theoretical learning but also provides a hands-on understanding of how real-world communication systems operate under practical constraints

10.Future Scope and Applications

Handling More Channels
The current design supports three channels, but future systems can scale up to:
8 channels
16 channels
24-channel (T1) or 30-channel (E1) frames
This makes the system comparable to real telecommunications.

Using Real Audio Instead of Synthetic Waves
Instead of sine/cosine/sawtooth, real microphone audio can be sampled and multiplexed.
This makes the system behave like:
A mini telephone exchange
A multi-user audio streaming system

Simulating Real Communication Problems
Future versions can include:
Noise (AWGN)
Bit errors
Jitter or timing mismatch
This helps study how strong TDM+PCM remains under bad network conditions.

Adding Error Correction
Techniques like:
Parity
Hamming code
Simple FEC
can be added to automatically correct mistakes in the received data.

Implementing on Hardware
The system can be moved to:
Arduino + ADC
DSP boards
FPGA (for real-time multi-channel TDM)
This converts the project from simulation to a working prototype.

Adding Frame Synchronization Bits
Just like actual telecom systems, future versions can include:
Sync patterns
Frame alignment word (FAW)
This ensures perfect time-slot alignment even in noisy conditions.

References

- Gibson, J. D. *Introduction to Communication Systems*, Springer, 2016.
- Kurmi, S. R., & Singh, R. K., *Enhanced TDM-PON for Broadband Systems*, IEEE, 2018.