# uni-polar-non-return-to-zero

March 29, 2024

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[4]:
```
N = 10;
n = np.random.randint(0, 2, N)
n
```

[4]:
```
array([1, 1, 0, 1, 0, 1, 0, 0, 0, 1])
```

[7]:
```
t = np.arange(0, N, 0.01)
t[:10]
```

[7]:
```
array([0.  , 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09])
```

[8]:
```
nn = [1 if bit== 1 else 0 for bit in n]
nn
```

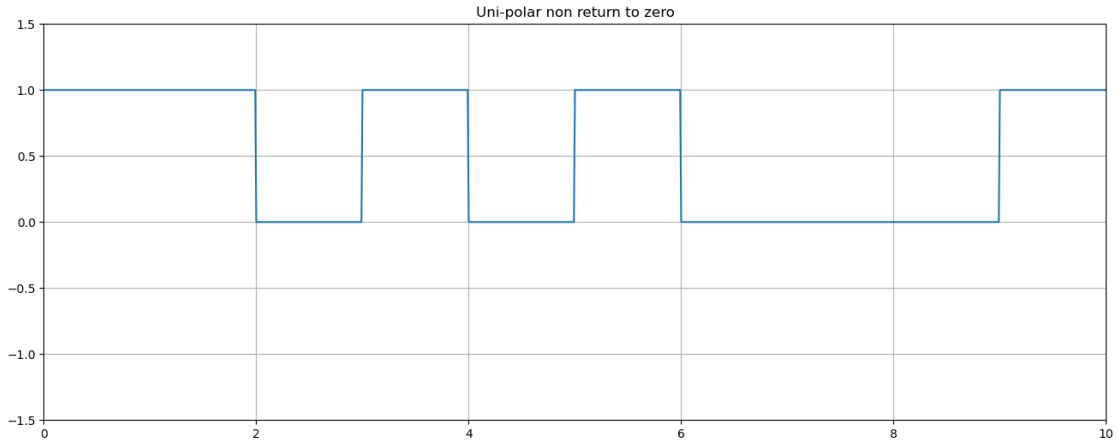
[8]:
```
[1, 1, 0, 1, 0, 1, 0, 0, 0, 1]
```

[12]:
```
nn = np.repeat(n , int(len(t))/N)
nn[800:1000]
```

[12]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1])
```

[20]:
```
plt.figure(figsize=(16, 6))
plt.plot(t, nn)
plt.grid(True)
plt.axis([0, N, -1.5, 1.5])
plt.title("Uni-polar non return to zero")
```

[20]: Text(0.5, 1.0, 'Uni-polar non return to zero')




[17]:
```
n
```

[17]: array([1, 1, 0, 1, 0, 1, 0, 0, 0, 1])

[ ]:

# polar-non-return-to-zero

March 29, 2024

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

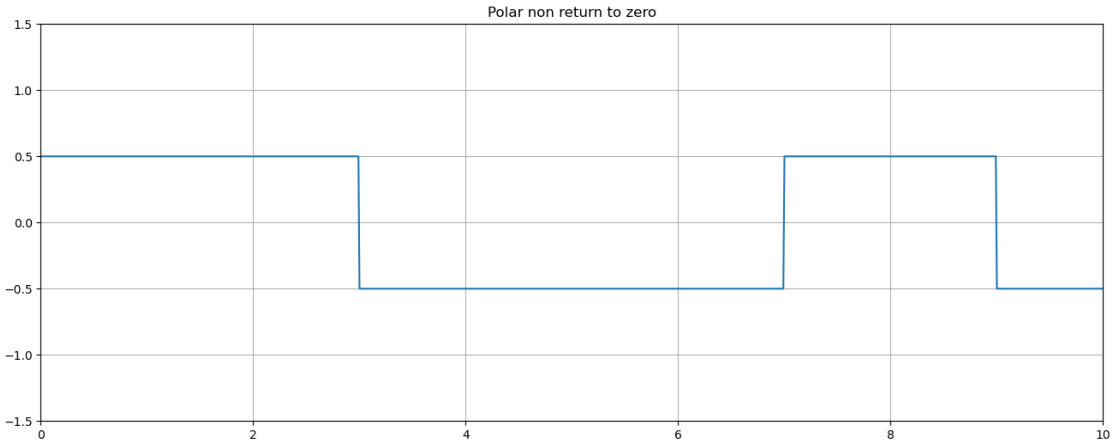
[3]:
```
N = 10
n = np.random.randint(0, 2, N)
n
```

[3]:
```
array([1, 1, 1, 0, 0, 0, 0, 1, 1, 0])
```

[9]:
```
t = np.arange(0, N, 0.01)
nn = [0.5 if bit==1 else -0.5 for bit in n]
nn = np.repeat(nn, int(len(t))/ N)
```

[10]:
```
plt.figure(figsize=(16, 6))
plt.plot(t, nn)
plt.grid(True)
plt.axis([0, N, -1.5, 1.5])
plt.title("Polar non return to zero")
```

[10]:
```
Text(0.5, 1.0, 'Polar non return to zero')
```




[11]:
```
n
```

[11]:
```
array([1, 1, 1, 0, 0, 0, 0, 1, 1, 0])
```

[ ]:

# nipolar-return-to-zero-line-coding

March 29, 2024

[71]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[99]:
```
N = 10
n = np.random.randint(0, 2, N)
n
```

[99]:
```
array([0, 1, 0, 1, 0, 1, 1, 1, 0, 0])
```

[112]:
```
t = np.arange(0, N, 0.01)
nn = [1 if bit == 1 else 0 for bit in n]
nn = np.repeat(nn, int(len(t)/N))
#nn
```

[114]:
```
t = np.arange(0, len(n), 0.01)
print(len(t))
#t
```

```
1000
```

[115]:
```
y = np.zeros(len(t))
len(y)
```

[115]:
```
1000
```

[116]:
```
i = 1
a = 0
b = 0.5
```

[117]:
```
for j in range(len(t)):
    if t[j] >= a and t[j] <= b:
        y[j] = nn[j]
    elif t[j] > b and t[j] < i:
        y[j] = 0
    else:
        i = i + 1
        a = a + 1
        b = b + 1
```

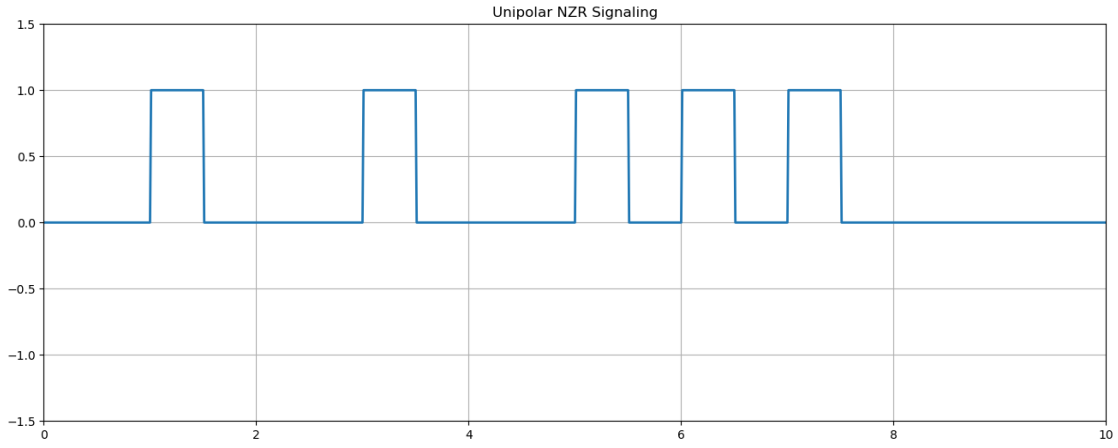
[ ]:

[118]:
```
print(n)
```

```
[0 1 0 1 0 1 1 1 0 0]
```

[119]:
```
plt.figure(figsize=(16, 6))
plt.plot(t, y, linewidth=2)
plt.axis([0, N, -1.5, 1.5])
plt.grid(True)
plt.title('Unipolar NZR Signaling')
```

[119]:
```
Text(0.5, 1.0, 'Unipolar NZR Signaling')
```




[ ]:

[ ]:

# bipolar-return-to-zero-signaling

March 29, 2024

[16]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[17]:
```
N = 10
n = np.random.randint(0, 2, N)
n
```

[17]:
```
array([0, 0, 0, 0, 1, 0, 0, 1, 1, 1])
```

[18]:
```
t = np.arange(0, N, 0.01)
```

[28]:
```
nn = [0 if bit==0 else 1 for bit in n]
nn = np.repeat(n, int(len(t)/N))
for i in range(len(nn)):
    if(nn[i] == 0):
        nn[i] = -1
#nn
```

[27]:
```
y = np.zeros(len(t))
len(y)
#y
```

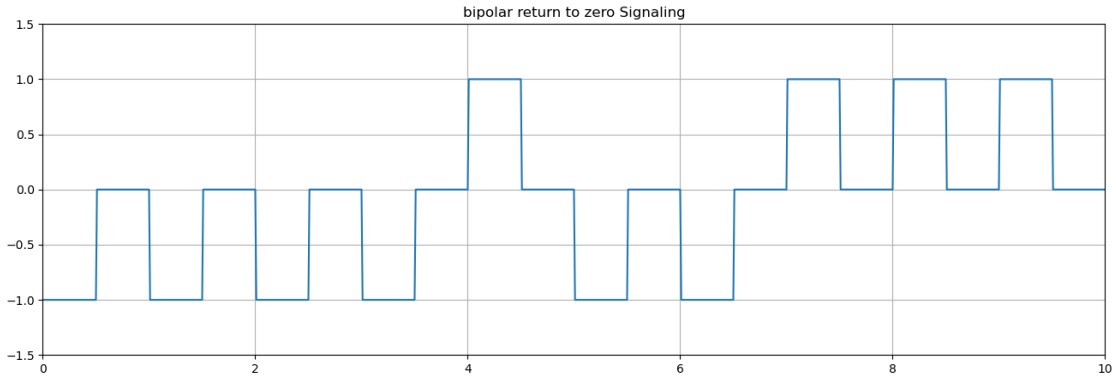
[27]:
```
1000
```

[23]:
```
i = 1
a = 0
b = 0.5
```

[24]:
```
for j in range(len(t)):
    if t[j] >= a and t[j] <= b:
        y[j] = nn[j]
    elif t[j] > b and t[j] < i:
        y[j] = 0
    else:
        i = i + 1
        a = a + 1
        b = b + 1
```

[25]:
```
plt.figure(figsize=(16, 5))
plt.plot(t, y)
plt.axis([0, N, -1.5, 1.5])  # Axis set-up
plt.grid(True)
plt.title('bipolar return to zero Signaling')
```

[25]:
```
Text(0.5, 1.0, 'bipolar return to zero Signaling')
```




[26]:
```
n
```

[26]:
```
array([0, 0, 0, 0, 1, 0, 0, 1, 1, 1])
```

[ ]:
```

```

# split-phase-manchester-coding

March 29, 2024

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[17]:
```
N = 10
n = np.random.randint(0, 2, N)
n
```

[17]:
```
array([1, 0, 1, 0, 1, 1, 1, 0, 1, 0])
```

[18]:
```
nnn = []
for m in range(N):
    if n[m] == 1:
        nn = [1, -1]
    else:
        nn = [-1, 1]
    nnn.extend(nn)
nnn
```

[18]:
```
[1, -1, -1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, -1, -1, 1, 1, -1, -1, 1]
```

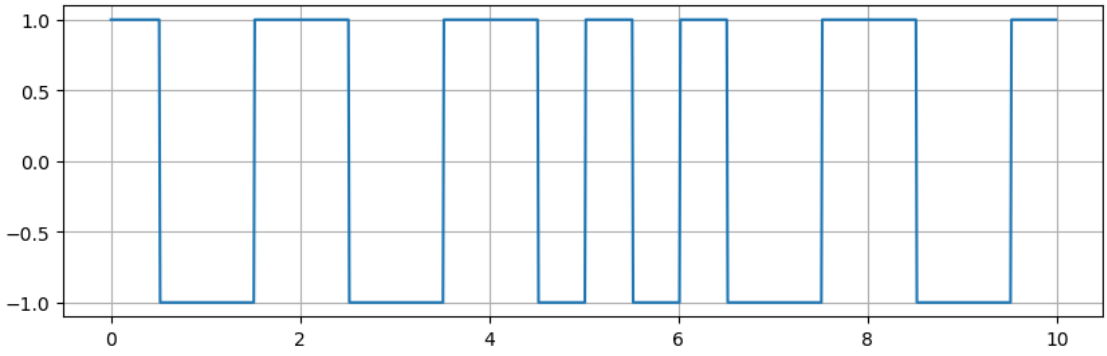
[19]:
```
i = 0
l = 0.5
t = np.arange(0, N, 0.01)
t[: 200]
```

[19]:
```
array([0.  , 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ,
       0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21,
       0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32,
       0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43,
       0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 , 0.51, 0.52, 0.53, 0.54,
       0.55, 0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65,
       0.66, 0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76,
       0.77, 0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87,
       0.88, 0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98,
       0.99, 1.  , 1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07, 1.08, 1.09,
       1.1 , 1.11, 1.12, 1.13, 1.14, 1.15, 1.16, 1.17, 1.18, 1.19, 1.2 ,
       1.21, 1.22, 1.23, 1.24, 1.25, 1.26, 1.27, 1.28, 1.29, 1.3 , 1.31,
```

```
1.32, 1.33, 1.34, 1.35, 1.36, 1.37, 1.38, 1.39, 1.4 , 1.41, 1.42,
1.43, 1.44, 1.45, 1.46, 1.47, 1.48, 1.49, 1.5 , 1.51, 1.52, 1.53,
1.54, 1.55, 1.56, 1.57, 1.58, 1.59, 1.6 , 1.61, 1.62, 1.63, 1.64,
1.65, 1.66, 1.67, 1.68, 1.69, 1.7 , 1.71, 1.72, 1.73, 1.74, 1.75,
1.76, 1.77, 1.78, 1.79, 1.8 , 1.81, 1.82, 1.83, 1.84, 1.85, 1.86,
1.87, 1.88, 1.89, 1.9 , 1.91, 1.92, 1.93, 1.94, 1.95, 1.96, 1.97,
1.98, 1.99])
```

[20]:
```
y = []
for j in range(len(t)):
    if t[j] <= l:
        y.append(nnn[i])
    else:
        y.append(nnn[i])
        i = i + 1
        l = l + 0.5
#y[800:1000]
```

[21]:
```
plt.figure(figsize=(10, 3))
plt.plot(t, y)
plt.grid(True)
```




[ ]:

# ask-modulation-demodulation

March 29, 2024

[22]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import simps
```

[47]:
```
x = np.random.randint(0, 2, 10)
bp = 0.000001
x
```

[47]:
```
array([0, 1, 1, 0, 0, 0, 0, 1, 0, 0])
```

[48]:
```
bit = np.array([])
for n in range(len(x)):
    if x[n] == 1:
        se = np.ones(100)
    else:
        se = np.zeros(100)
    bit = np.concatenate((bit, se))
len(bit)
```

[48]:
```
1000
```

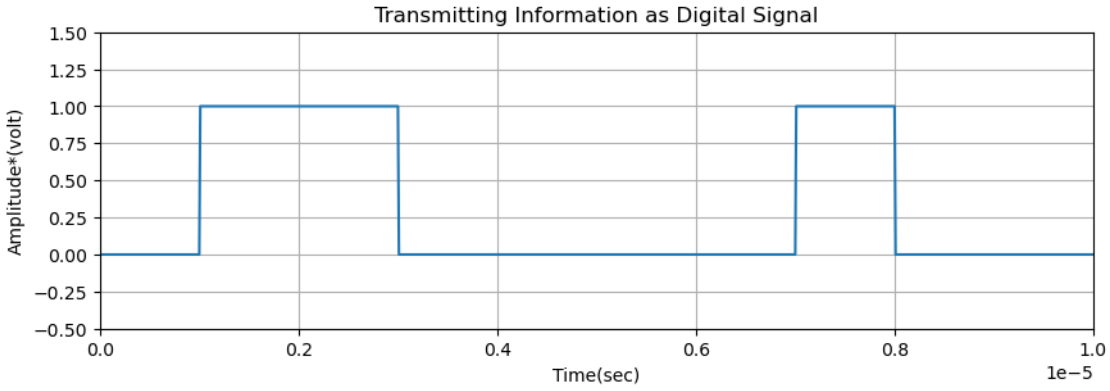
[ ]:
```

```

[49]:
```
t1 = np.arange(bp/100, 100*len(x)*(bp/100) + bp/100, bp/100)
t1 = t1[:len(bit)]
```

[50]:
```
plt.figure(figsize=(10, 3))
plt.plot(t1, bit)
plt.grid(True)
plt.axis([0, bp*len(x), -0.5, 1.5])
plt.ylabel('Amplitude*(volt)')
plt.xlabel('Time(sec)')
plt.title('Transmitting Information as Digital Signal')
```

[50]:
```
Text(0.5, 1.0, 'Transmitting Information as Digital Signal')
```




### Binary ASK Modulation

[51]:
```
A1 = 10
A2 = 5
br = 1/bp
f = br*10
```

[52]:
```
t2 = np.arange(bp/99, bp + bp/99, bp/99)
len(t2)
```

[52]: 99

[53]:
```

```

[56]:
```
m = np.array([])
for i in range(len(x)):
    if x[i] == 1:
        y = A1*np.cos(2*np.pi*f*t2)
    else:
        y = A2*np.cos(2*np.pi*f*t2)
    m = np.concatenate((m, y))

len(m)
```

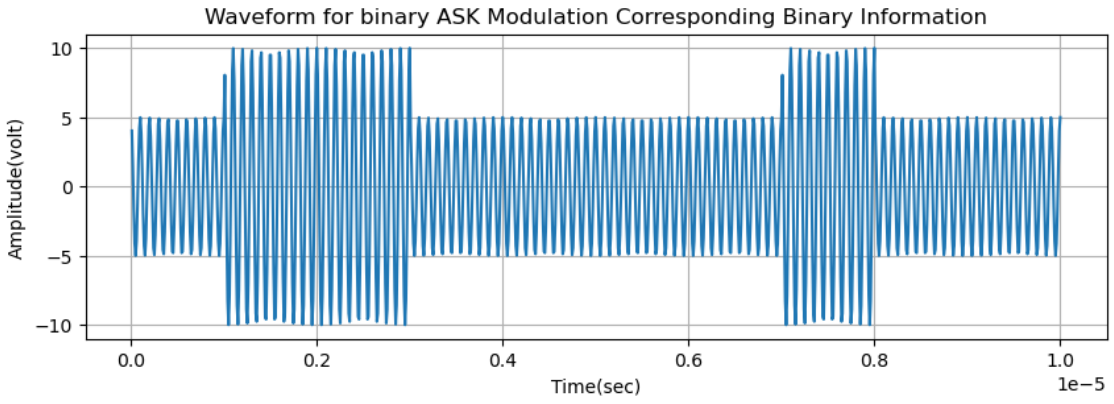
[56]: 990

[57]:
```
t3 = np.arange(bp/99, bp*len(x) + bp/99, bp/99)
t3 = t3[:len(m)]
len(t3)
```

[57]: 990

[58]:
```
plt.figure(figsize=(10,3))
plt.plot(t3, m)
plt.grid(True)
plt.xlabel('Time(sec)')
plt.ylabel('Amplitude(volt)')
plt.title('Waveform for binary ASK Modulation Corresponding Binary Information')
```

[58]:
```
Text(0.5, 1.0, 'Waveform for binary ASK Modulation Corresponding Binary
Information')
```




### Binary ASK Demodulation

[76]:
```
ss = len(t2)
ss
```

[76]:
```
99
```

[86]:
```
mn = np.array([])
for n in range(ss-1, len(m), ss):
    t = np.arange(bp/99, bp + bp/99, bp/99)
    y = np.cos(2*np.pi*f*t)
    mm = y*m[n-ss+1:n+1]
    z = simps(mm, t)
    zz = round((2*z/bp))
    if zz > 7.5:
        a = 1
    else:
        a = 0
    mn = np.append(mn, a)
```

[87]:
```
len(mn)
```

[87]:
```
10
```

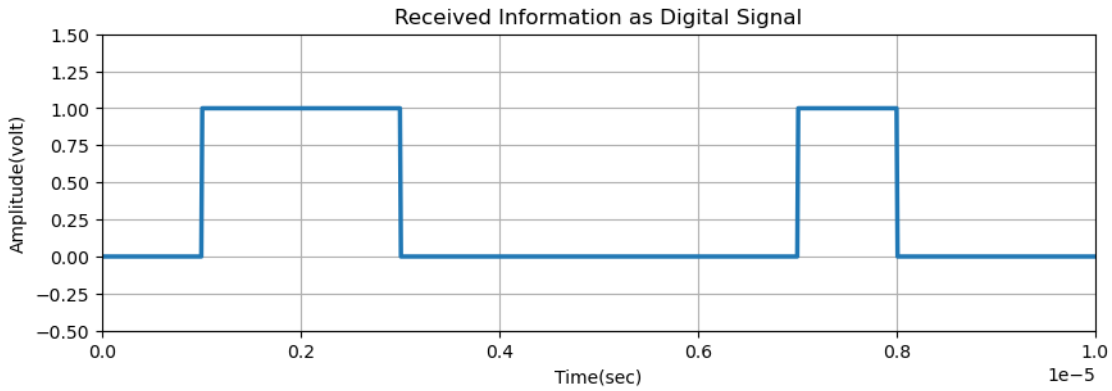
[ ]:

[88]:
```
bit = np.array([])
for n in range(len(mn)):
    if mn[n] == 1:
        se = np.ones(100)
    else:
        se = np.zeros(100)
    bit = np.concatenate((bit, se))
```

[92]:
```
t4 = np.arange(bp/100, 100*len(mn)*(bp/100) + bp/100, bp/100)
t4 = t4[:len(bit)]
```

[93]:
```
plt.figure(figsize=(10, 3))
plt.plot(t4, bit, linewidth=2.5)
plt.grid(True)
plt.axis([0, bp*len(mn), -0.5, 1.5])
plt.ylabel('Amplitude(volt)')
plt.xlabel('Time(sec)')
plt.title('Received Information as Digital Signal')
```

[93]:
```
Text(0.5, 1.0, 'Received Information as Digital Signal')
```



# fsk-modulation-and-demodulation

March 29, 2024

[36]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[75]:
```
x = np.random.randint(0, 2, 10)
x
```

[75]:
```
array([1, 0, 0, 0, 1, 1, 1, 1, 0, 1])
```

[76]:
```
bp = 0.000001
```

[77]:
```
bit = np.array([])
for n in range(len(x)):
    if x[n] == 1:
        se = np.ones(100)
    else:
        se = np.zeros(100)
    bit = np.concatenate((bit, se))
len(bit)
```

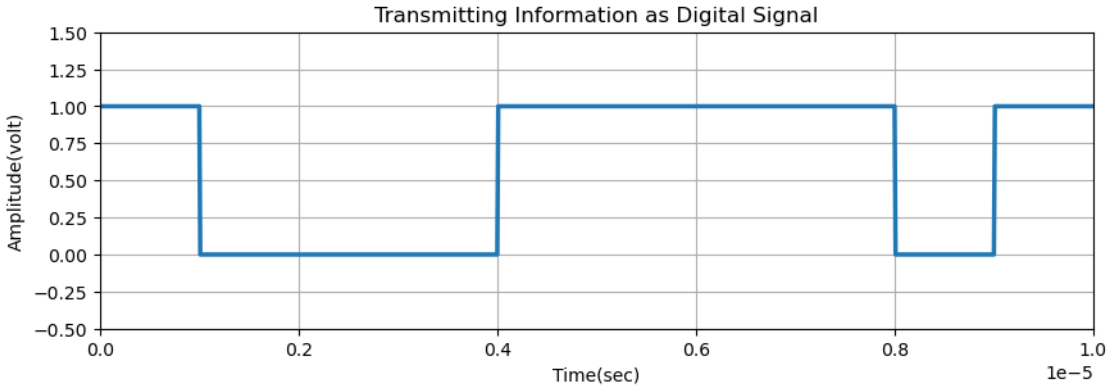
[77]:
```
1000
```

[78]:
```
t1 = np.arange(bp/100, 100*len(x)*(bp/100) + bp/100, bp/100)
t1 = t1[0:len(t1)-1]
len(t1)
```

[78]:
```
1000
```

[79]:
```
plt.figure(figsize=(10, 3))
plt.plot(t1, bit, linewidth=2.5)
plt.grid(True)
plt.axis([0, bp*len(x), -0.5, 1.5])
plt.ylabel('Amplitude(volt)')
plt.xlabel('Time(sec)')
plt.title('Transmitting Information as Digital Signal')
```

[79]:
```
Text(0.5, 1.0, 'Transmitting Information as Digital Signal')
```




### binary FSK Modulation

[80]:
```
A = 5
br = 1/bp
f1 = br*10
f2 = br*1
```

[81]:
```
t2 = np.arange(bp/99, bp + bp/99, bp/99)
len(t2)
```

[81]: 99

[82]:
```
ss = len(t2)
```

[83]:
```
m = np.array([])
for i in range(len(x)):
    if x[i] == 1:
        y = A * np.cos(2*np.pi*f1*t2)
    else:
        y = A * np.cos(2*np.pi*f2*t2)
    m = np.concatenate((m, y))
len(m)
```

[83]: 990

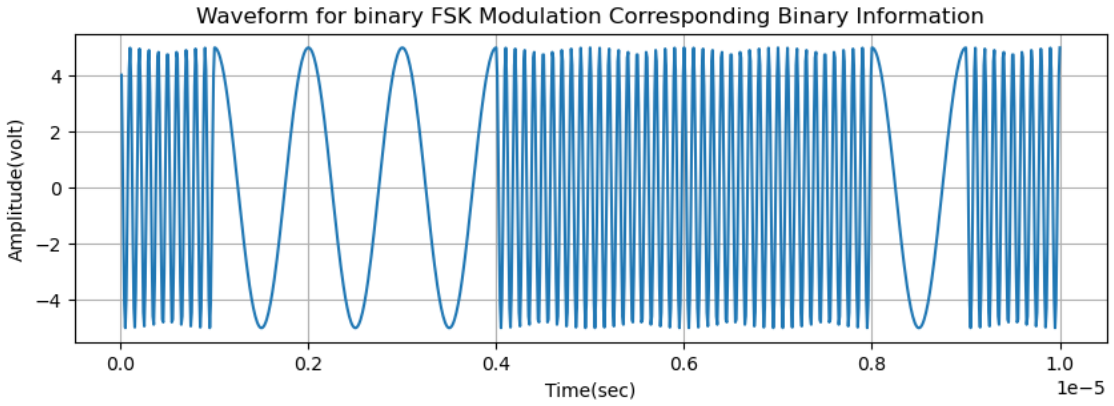
[84]:
```
t3 = np.arange(bp/99, bp*len(x) + bp/99, bp/99)
len(t3)
```

[84]: 990

[85]:
```
plt.figure(figsize=(10,3))
plt.plot(t3, m)
plt.grid(True)
```

```
plt.xlabel('Time(sec)')
plt.ylabel('Amplitude(volt)')
plt.title('Waveform for binary FSK Modulation Corresponding Binary Information')
```

[85]:
```
Text(0.5, 1.0, 'Waveform for binary FSK Modulation Corresponding Binary
Information')
```




### Binary FSK Demodulation

[ ]:
```
```

[110]:
```
mn = np.array([])
for n in range(ss-1, len(m), ss):
    t = np.arange(bp/99, bp + bp/99, bp/99)
    y1 = np.cos(2*np.pi*f1*t)
    y2 = np.cos(2*np.pi*f2*t)
    segment = m[n-ss+1:n+1]
    mm = y1 * segment
    mmm = y2 * segment
    z1 = np.trapz(mm, t)
    z2 = np.trapz(mmm, t)
    zz1 = round((2*z1/bp))
    zz2 = round((2*z2/bp))
    if zz1 > A/2:
        a = 1
    elif zz2 > A/2:
        a = 0
    mn = np.append(mn, a)
```

[111]:
```
len(mn)
```

[111]:
```
10
```

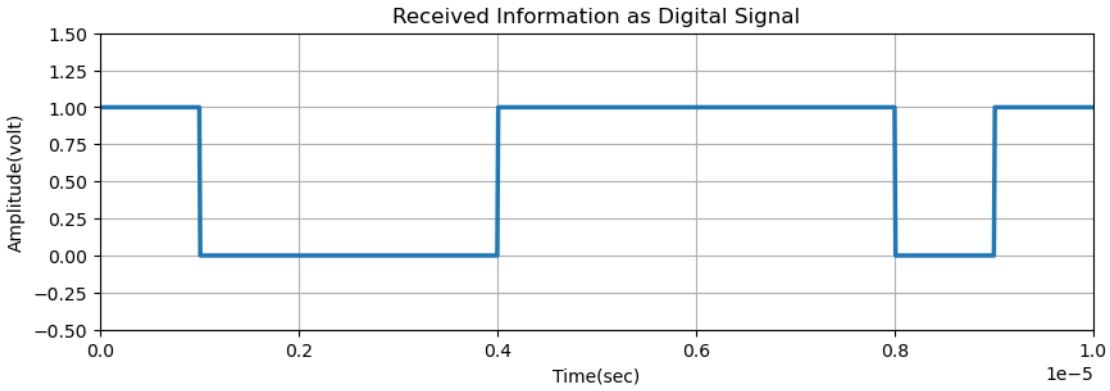
[112]:
```
bit = np.array([])
for n in range(len(mn)):
    if mn[n] == 1:
        se = np.ones(100)
    else:
        se = np.zeros(100)
    bit = np.concatenate((bit, se))
```

[117]:
```
t4 = np.arange(bp/100, 100*len(mn)*(bp/100) + bp/100, bp/100)
t4 = t4[:len(bit)]
len(t4)
```

[117]: 1000

[118]:
```
plt.figure(figsize=(10,3))
plt.plot(t4, bit, linewidth=2.5)
plt.grid(True)
plt.axis([0, bp*len(mn), -0.5, 1.5])
plt.ylabel('Amplitude(volt)')
plt.xlabel('Time(sec)')
plt.title('Received Information as Digital Signal')
```

[118]: Text(0.5, 1.0, 'Received Information as Digital Signal')




[ ]:

# psk-modulation-demodulation

March 29, 2024

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import simps
```

[3]:
```
x = np.random.randint(0, 2, 10)
x
```

[3]:
```
array([1, 1, 1, 1, 0, 1, 0, 1, 1, 0])
```

[4]:
```
bp = 0.000001
```

[5]:
```
bit = np.array([])
for n in range(len(x)):
    if x[n] == 1:
        se = np.ones(100)
    else:
        se = np.zeros(100)
    bit = np.concatenate((bit, se))
len(bit)
```

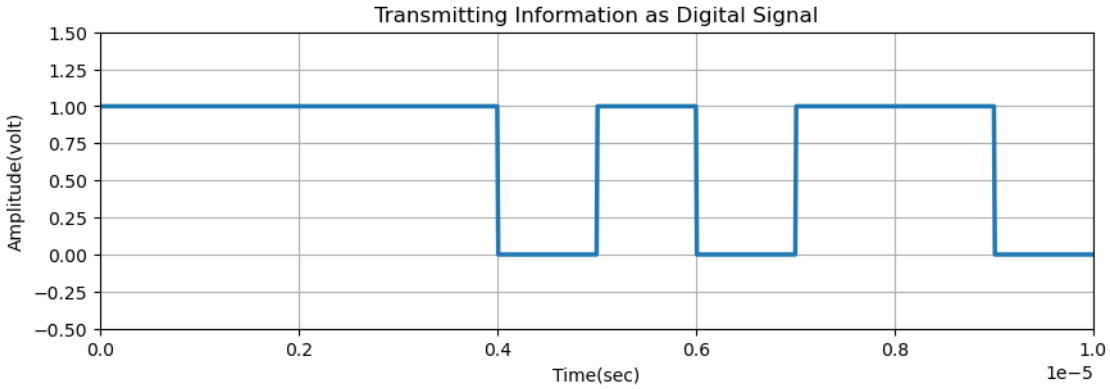
[5]:
```
1000
```

[12]:
```
t1 = np.arange(bp/100, 100*len(x)*(bp/100) + bp/100, bp/100)
t1 = t1[:len(bit)]
len(t1)
```

[12]:
```
1000
```

[13]:
```
plt.figure(figsize=(10,3))
plt.plot(t1, bit, linewidth=2.5)
plt.grid(True)
plt.axis([0, bp*len(x), -0.5, 1.5])
plt.ylabel('Amplitude(volt)')
plt.xlabel('Time(sec)')
plt.title('Transmitting Information as Digital Signal')
```

[13]:
```
Text(0.5, 1.0, 'Transmitting Information as Digital Signal')
```




### Binary PSK Modulation

[31]:
```
A = 5
br = 1/bp
f = br*1
t2 = np.arange(bp/99, bp + bp/99, bp/99)
ss = len(t2)
ss
```

[31]: 99

[32]:
```
m = np.array([])
for i in range(len(x)):
    if x[i] == 1:
        y = A * np.cos(2*np.pi*f*t2)
    else:
        y = A * np.cos(2*np.pi*f*t2 + np.pi)
    m = np.concatenate((m, y))
len(m)
```

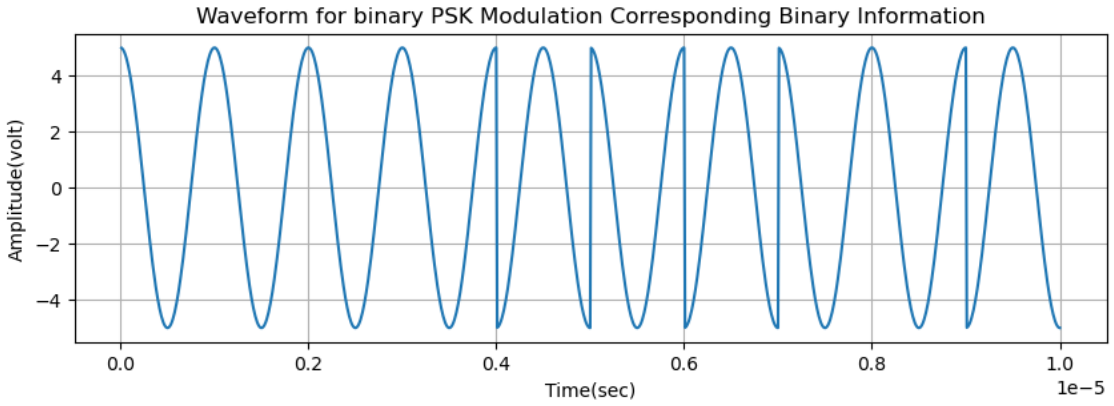
[32]: 990

[33]:
```
t3 = np.arange(bp/99, bp*len(x) + bp/99, bp/99)
len(t3)
```

[33]: 990

[34]:
```
plt.figure(figsize=(10, 3))
plt.plot(t3, m)
plt.grid(True)
plt.xlabel('Time(sec)')
plt.ylabel('Amplitude(volt)')
plt.title('Waveform for binary PSK Modulation Corresponding Binary Information')
```

[34]: Text(0.5, 1.0, 'Waveform for binary PSK Modulation Corresponding Binary Information')




### Binary PSK Demodulation

[37]:
```
mn = np.array([])

for n in range(ss-1, len(m), ss):
    t = np.arange(bp/99, bp + bp/99, bp/99)
    y = np.cos(2*np.pi*f*t)
    mm = y * m[n-ss+1:n+1]
    z = simps(mm, t)
    zz = round((2*z/bp))
    if zz > 0:
        a = 1
    else:
        a = 0
    mn = np.append(mn, a)
len(mn)
```

[37]: 10

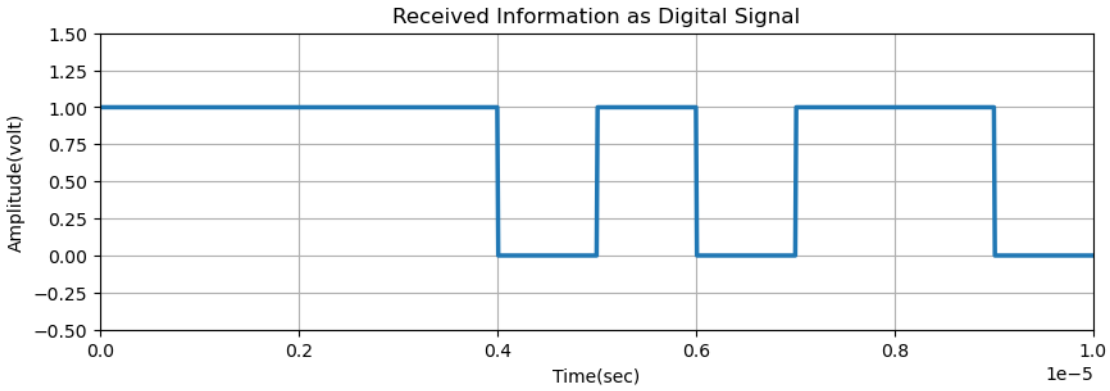
[38]:
```
bit = np.array([])
for n in range(len(mn)):
    if mn[n] == 1:
        se = np.ones(100)
    else:
        se = np.zeros(100)
    bit = np.concatenate((bit, se))
```

[40]:
```
t4 = np.arange(bp/100, 100*len(mn)*(bp/100) + bp/100, bp/100)
t4 = t4[:len(bit)]
len(t4)
```

[40]: 1000

[41]:
```
plt.figure(figsize=(10,3))
plt.plot(t4, bit, linewidth=2.5)
plt.grid(True)
plt.axis([0, bp*len(mn), -0.5, 1.5])
plt.ylabel('Amplitude(volt)')
plt.xlabel('Time(sec)')
plt.title('Received Information as Digital Signal')
```

[41]: Text(0.5, 1.0, 'Received Information as Digital Signal')




[ ]:

# qpsk-modulation

March 29, 2024

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[23]:
```
x = np.random.randint(0, 2, 10)
x
```

[23]:
```
array([0, 0, 1, 1, 1, 0, 1, 1, 0, 1])
```

[24]:
```
p = np.where(x == 0, -1, 1)
p
```

[24]:
```
array([-1, -1,  1,  1,  1, -1,  1,  1, -1,  1])
```

[25]:
```
even_seq = p[::2]
odd_seq = p[1::2]
print(even_seq)
print(odd_seq)
```

```
[-1  1  1  1 -1]
[-1  1 -1  1  1]
```

[26]:
```
t = np.arange(0, len(x), 0.01)
len(t)
```

[26]:
```
1000
```

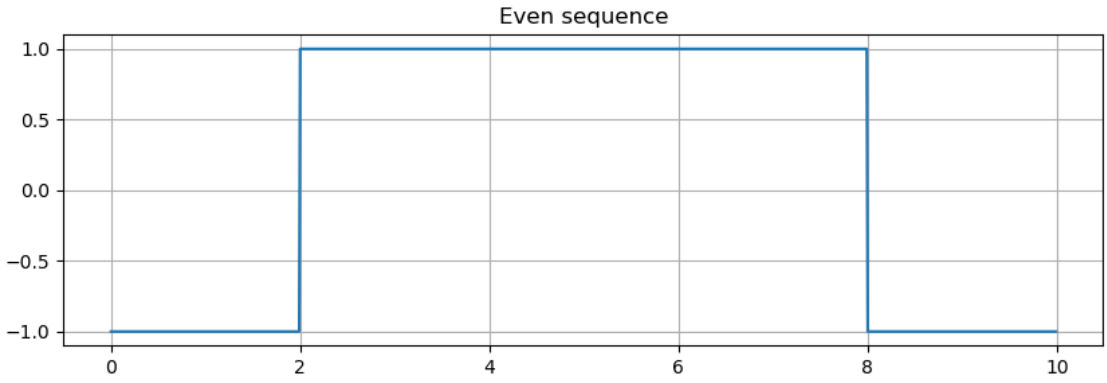
[27]:
```
even_ps = np.repeat(even_seq, int(len(t) / len(even_seq)))
odd_ps = np.repeat(odd_seq, int(len(t) / len(odd_seq)))
```

[28]:
```
c1 = np.cos(2 * np.pi * 1 * t)
c2 = np.sin(2 * np.pi * 1 * t)
```

[29]:
```
r1 = even_ps * c1
r2 = odd_ps * c2
qpsk_sig = r1 - r2
```

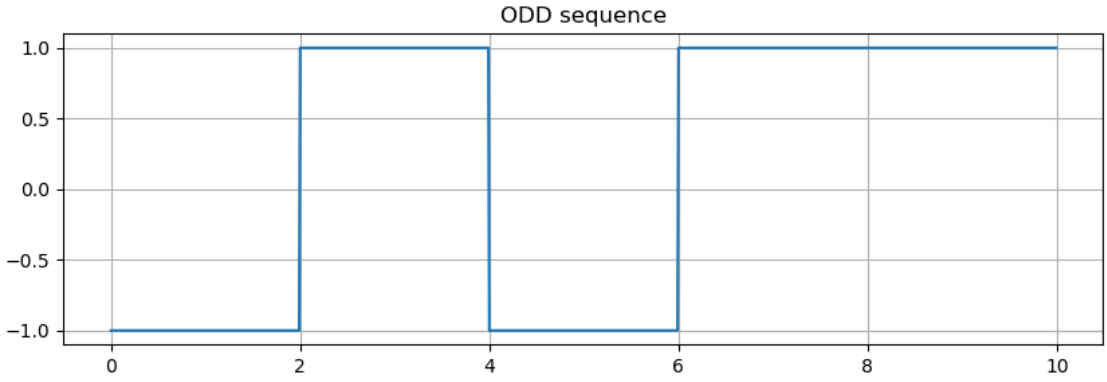
[30]:
```
plt.figure(figsize=(10, 3))
plt.plot(t, even_ps)
plt.grid(True)
plt.title("Even sequence")
```

[30]:
```
Text(0.5, 1.0, 'Even sequence')
```

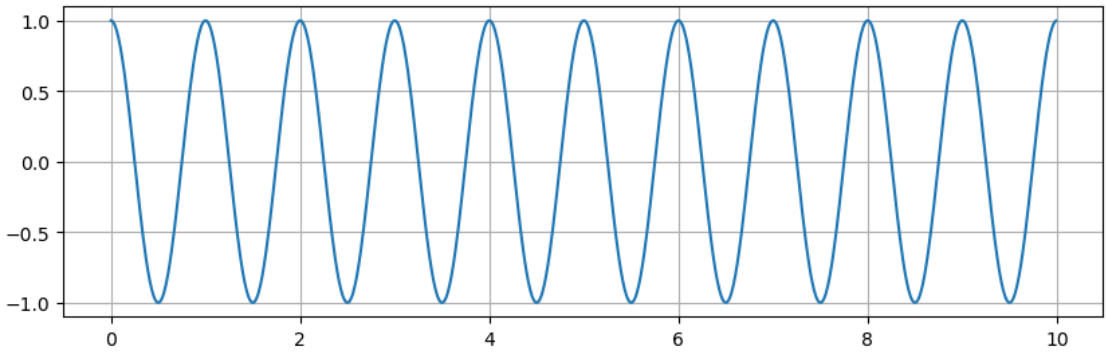



[31]:
```
plt.figure(figsize=(10, 3))
plt.plot(t, odd_ps)
plt.grid(True)
plt.title("ODD sequence")
```

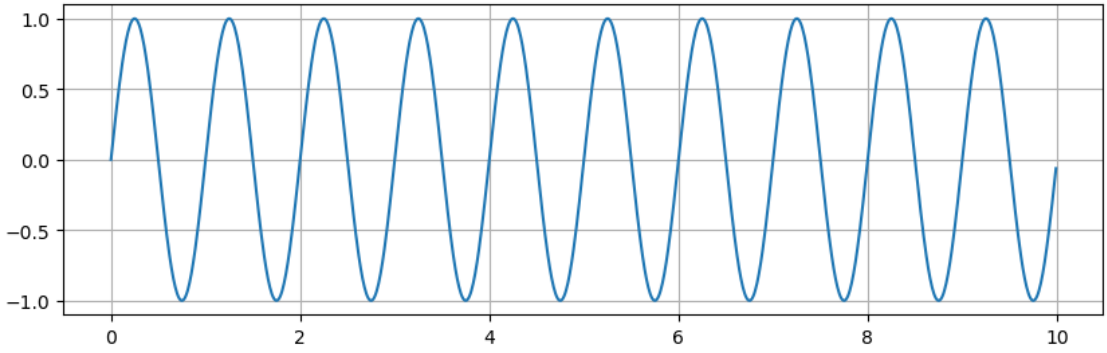
[31]:
```
Text(0.5, 1.0, 'ODD sequence')
```



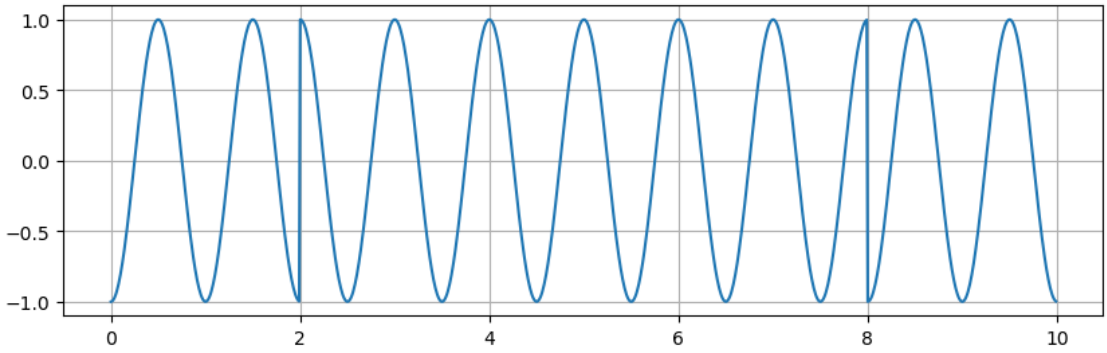

[33]:
```
plt.figure(figsize=(10, 3))
plt.plot(t, c1)
plt.grid(True)
```



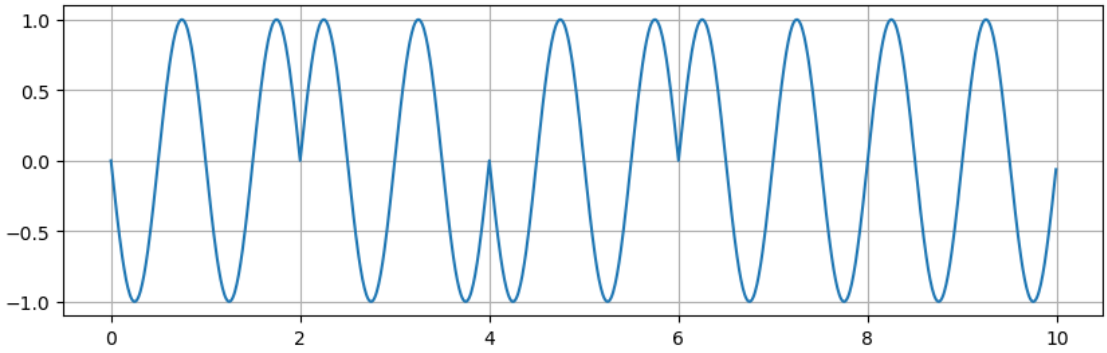

[35]:
```
plt.figure(figsize=(10, 3))
plt.plot(t, c2)
plt.grid(True)
```



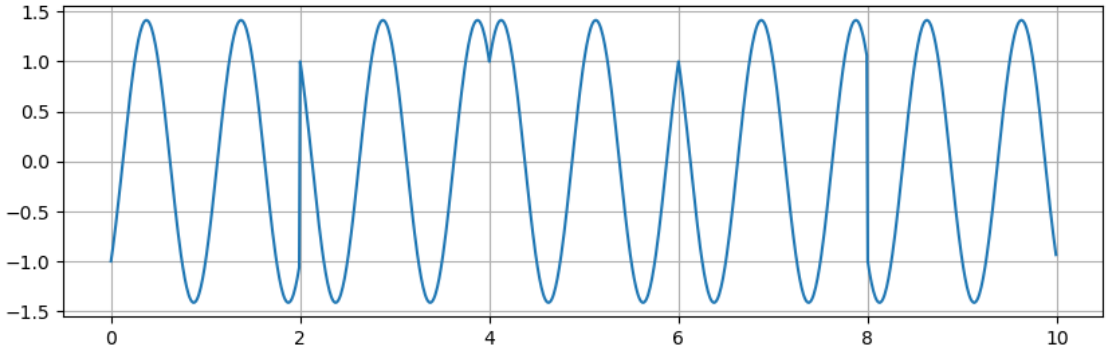

[40]:
```
plt.figure(figsize=(10,3))
plt.plot(t, r1)
plt.grid(True)
```



[41]:
```
plt.figure(figsize=(10, 3))
plt.plot(t, r2)
plt.grid(True)
```




[42]:
```
plt.figure(figsize=(10, 3))
plt.plot(t, qpsk_sig)
plt.grid(True)
```



# pulse-amplitude-modulation

March 29, 2024

[11]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[70]:
```
fc = 50
fm = 2
fs = 1000
t = 1
duty  = 20
n = np.arange(0, t, 1/fs)
```

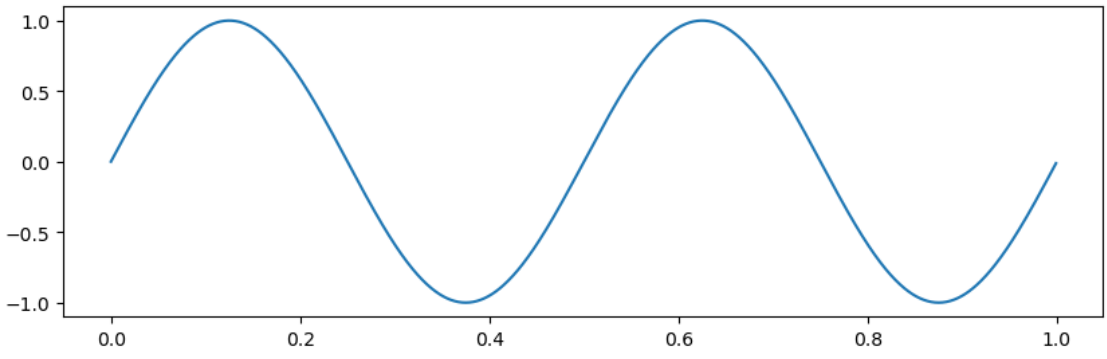
[71]:
```
s = np.sin(2 * np.pi * fc * n)
s[s < 0] = 0
```

[ ]:
```

```

[82]:
```
m = np.sin(2 * np.pi * fm * (n - 1))
plt.figure(figsize=(10, 3))
plt.plot(n, m)
```

[82]:
```
[<matplotlib.lines.Line2D at 0x26ae4bd1dd0>]
```




[83]:
```
period_sample = len(n) / fc
period_sample
```

[83]:
```
20.0
```

[84]:
```
index = np.arange(0, len(n), int(period_sample))
index
```

[84]:
```
array([  0,  20,  40,  60,  80, 100, 120, 140, 160, 180, 200, 220, 240,
       260, 280, 300, 320, 340, 360, 380, 400, 420, 440, 460, 480, 500,
       520, 540, 560, 580, 600, 620, 640, 660, 680, 700, 720, 740, 760,
       780, 800, 820, 840, 860, 880, 900, 920, 940, 960, 980])
```

[85]:
```
on_sample = int(np.ceil(period_sample * duty / 100))
on_sample
```

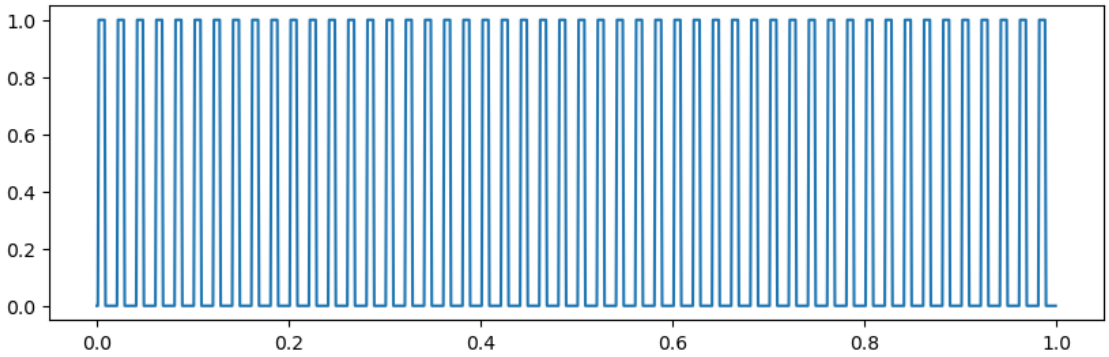
[85]:
```
4
```

[86]:
```
pam = np.zeros_like(n)
```

[87]:
```
carrier_pulse_train = np.where(s > 0.35, 1, 0)
```

[88]:
```
plt.figure(figsize=(10, 3))
plt.plot(n, carrier_pulse_train[: len(n)])
```

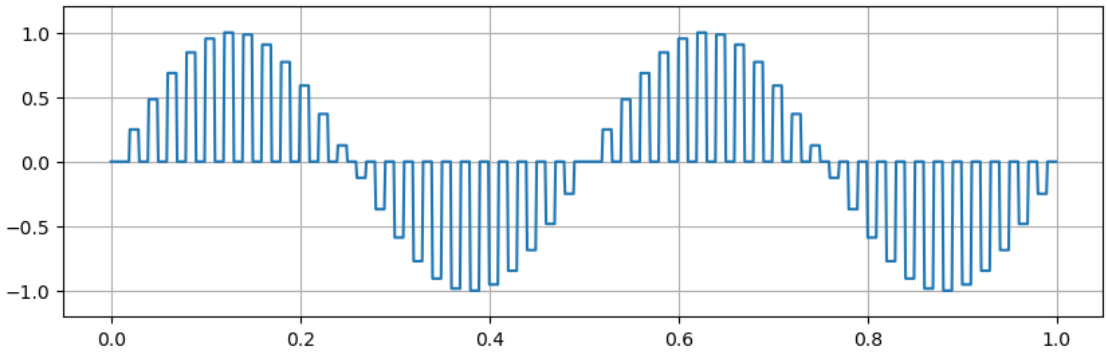
[88]:
```
[<matplotlib.lines.Line2D at 0x26ae4c5a390>]
```




[89]:
```
for i in range(len(index)):
    pam[index[i] : index[i] + on_samp] = m[index[i]]
```

[90]:
```
plt.figure(figsize=(10, 3))
plt.plot(n, pam)
plt.grid(True)
plt.ylim([-1.2, 1.2])
```

[90]:
```
(-1.2, 1.2)
```


1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0

[ ]: