EE120 Fall 2014 PS 5, Quadrature amplitude modulation for binary signal

To ouput python notebook, use ``ipython nbconvert file.ipynb" from command prompt. You may need to install pandoc first.

In [1]:
```
print 4+5  # check to see if iPython is running...
```

```
9
```

In [2]:
```
%pylab
```

```
Using matplotlib backend: Qt4Agg
Populating the interactive namespace from numpy and matplotlib
```

In [3]:
```
import numpy as np
import scipy as sp
print 'imported numpy and scipy'
import matplotlib.pyplot as plt
%matplotlib inline
from scipy.io import wavfile
print 'finished importing'
```

```
imported numpy and scipy
finished importing
```

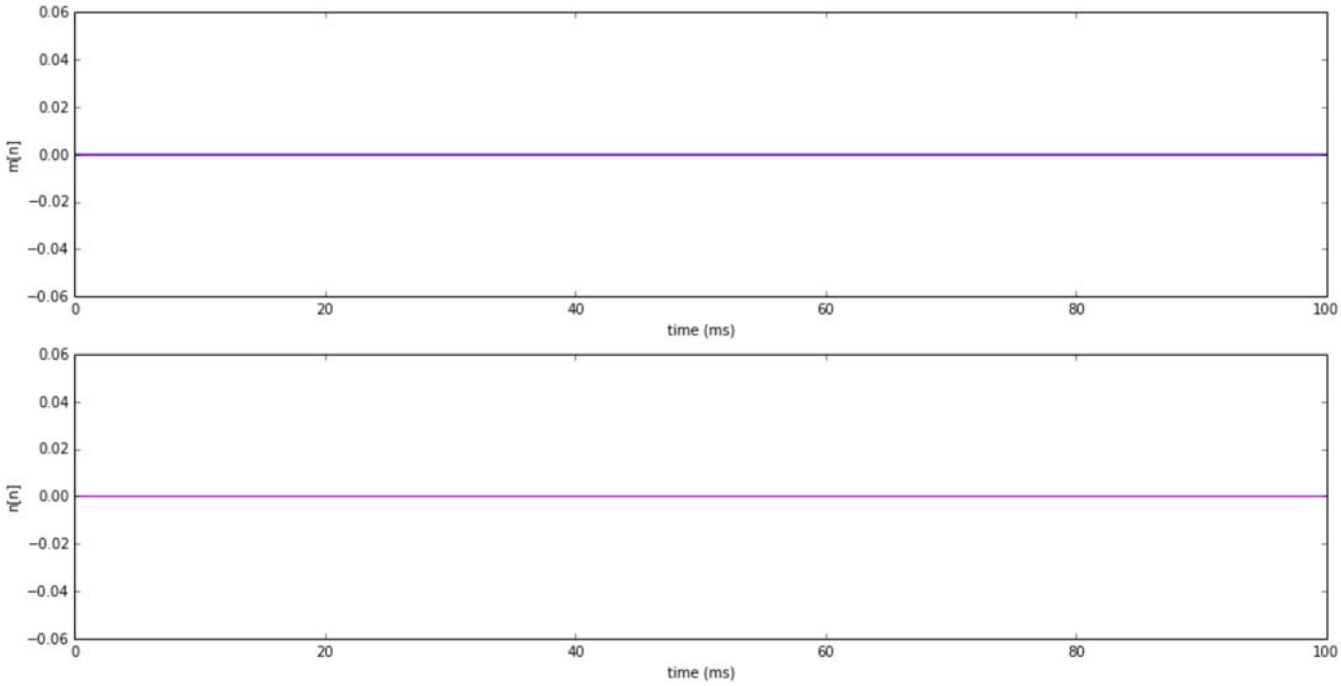
In [4]:
```
# Graphing helper function
def setup_graph(title='', x_label='', y_label='', fig_size=None):
    fig = plt.figure()
    if fig_size != None:
        fig.set_size_inches(fig_size[0], fig_size[1])
    ax = fig.add_subplot(111)
    ax.set_title(title)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
```

In [43]:
```
# import file
from scipy.io import wavfile
rate,data= wavfile.read('xmit-signal.wav')  # 16 bit data from transmitter
omega_c = 2.0 * np.pi * 5e5  # carrier frequency of radio transmitter
print 'rate =', rate
print 'data =', data
length = np.size(data)
print 'length  = ', length
dt = 0.25*1e-6  # sample period
time = dt * np.linspace(0,length,length)

################################################
#
# your code goes here to recover m[n] and n[n]
#
#
#
# m_est = f(data)    - to be determined
# n_est = g(data)   - to be determined
n_est = np.zeros(length)  # dummy initialization
m_est = np.zeros(length)  # dummy initialization
```

```
rate = 44100
data = [  8852  16418  14377 ..., -21190 -14377    865]
length  =  400000
```

In [44]:
```
# now plot data - you should not need to change these functions
# plot estimated m_est[n]
fig = figure(figsize = (16,8))
plt.subplot(2,1,1)
plt.plot(1000.0*time[0:length],m_est[0:length])
xlabel('time (ms)')
ylabel('m[n]')
ax = fig.add_subplot(2,1,1)
ax.axhline(linewidth=1, color='m')

# plot estimated n_est[n]
plt.subplot(2,1,2)
plt.plot(1000.0*time[0:length],n_est[0:length])
xlabel('time (ms)')
ylabel('n[n]')
ax = fig.add_subplot(2,1,2)
ax.axhline(linewidth=1, color='m')
```

Out[44]: <matplotlib.lines.Line2D at 0x129e9ac8>



In []: