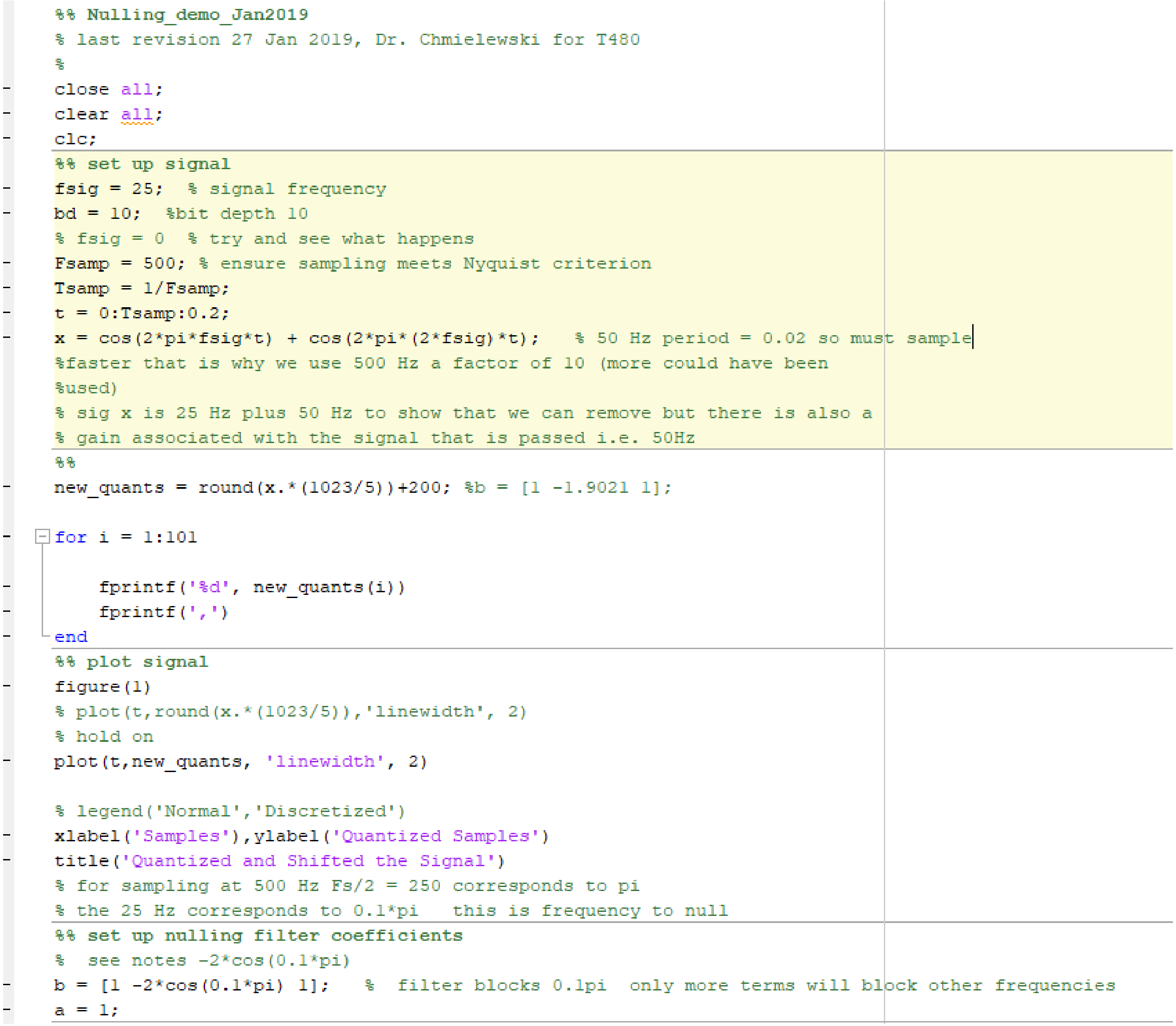
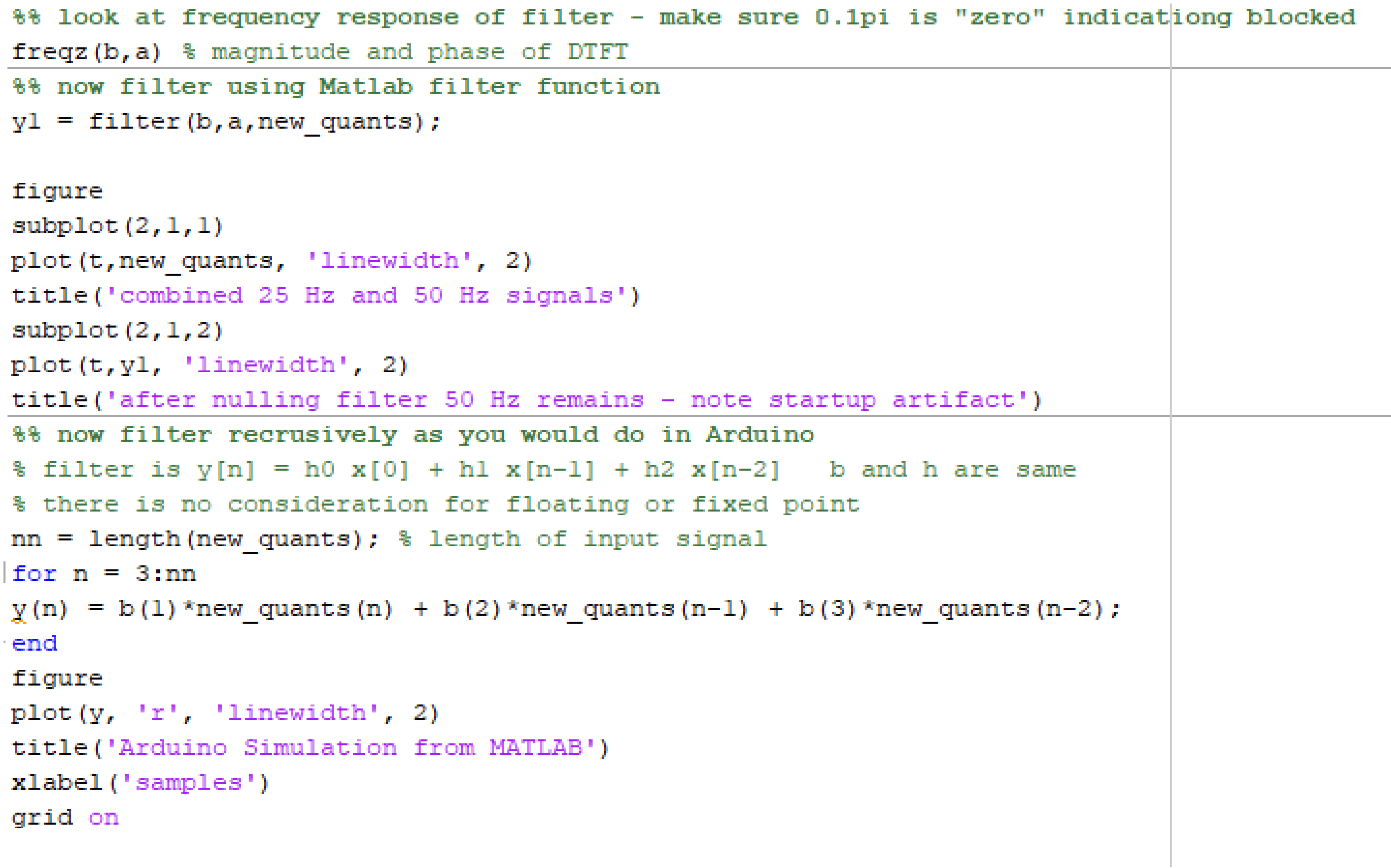
MATLAB

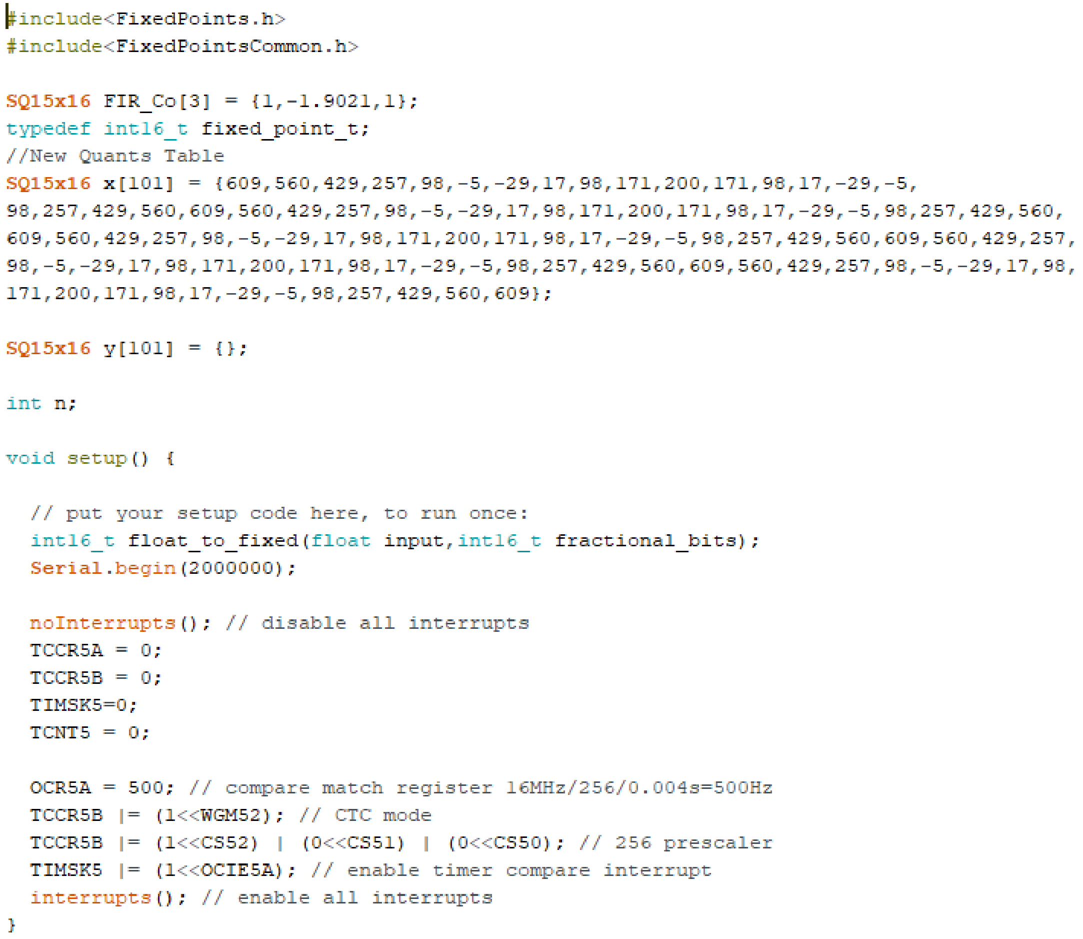


```
%% Nulling_demo_Jan2019
% last revision 27 Jan 2019, Dr. Chmielewski for T480
%
close all;
clear all;
clc;
%% set up signal
fsig = 25;  % signal frequency
bd = 10;  %bit depth 10
% fsig = 0  % try and see what happens
Fsamp = 500; % ensure sampling meets Nyquist criterion
Tsamp = 1/Fsamp;
t = 0:Tsamp:0.2;
x = cos(2*pi*fsig*t) + cos(2*pi*(2*fsig)*t);   % 50 Hz period = 0.02 so must sample
%faster that is why we use 500 Hz a factor of 10 (more could have been
%used)
% sig x is 25 Hz plus 50 Hz to show that we can remove but there is also a
% gain associated with the signal that is passed i.e. 50Hz
%%
new_quants = round(x.*(1023/5))+200; %b = [1 -1.9021 1];

for i = 1:101

    fprintf('%d', new_quants(i))
    fprintf(',')
end
%% plot signal
figure(1)
% plot(t,round(x.*(1023/5)),'linewidth', 2)
% hold on
plot(t,new_quants, 'linewidth', 2)

% legend('Normal','Discretized')
xlabel('Samples'),ylabel('Quantized Samples')
title('Quantized and Shifted the Signal')
% for sampling at 500 Hz Fs/2 = 250 corresponds to pi
% the 25 Hz corresponds to 0.1*pi   this is frequency to null
%% set up nulling filter coefficients
%  see notes -2*cos(0.1*pi)
b = [1 -2*cos(0.1*pi) 1];   %  filter blocks 0.1pi  only more terms will block other frequencies
a = 1;
```

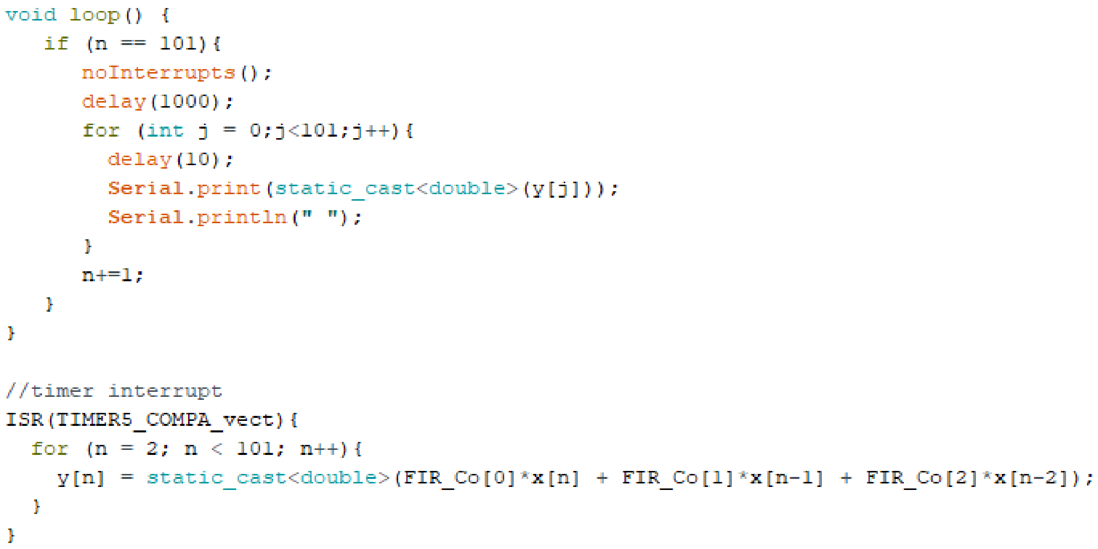


```
%% look at frequency response of filter - make sure 0.1pi is "zero" indicationg blocked
freqz(b,a) % magnitude and phase of DTFT
%% now filter using Matlab filter function
y1 = filter(b,a,new_quants);

figure
subplot(2,1,1)
plot(t,new_quants, 'linewidth', 2)
title('combined 25 Hz and 50 Hz signals')
subplot(2,1,2)
plot(t,y1, 'linewidth', 2)
title('after nulling filter 50 Hz remains - note startup artifact')
%% now filter recrusively as you would do in Arduino
% filter is y[n] = h0 x[0] + h1 x[n-1] + h2 x[n-2]    b and h are same
% there is no consideration for floating or fixed point
nn = length(new_quants); % length of input signal
for n = 3:nn
y(n) = b(1)*new_quants(n) + b(2)*new_quants(n-1) + b(3)*new_quants(n-2);
end
figure
plot(y, 'r', 'linewidth', 2)
title('Arduino Simulation from MATLAB')
xlabel('samples')
grid on
```

Arduino



```
#include<FixedPoints.h>
#include<FixedPointsCommon.h>

SQ15x16 FIR_Co[3] = {1,-1.9021,1};
typedef int16_t fixed_point_t;
//New Quants Table
SQ15x16 x[101] = {609,560,429,257,98,-5,-29,17,98,171,200,171,98,17,-29,-5,
98,257,429,560,609,560,429,257,98,-5,-29,17,98,171,200,171,98,17,-29,-5,98,257,429,560,
609,560,429,257,98,-5,-29,17,98,171,200,171,98,17,-29,-5,98,257,429,560,609,560,429,257,
98,-5,-29,17,98,171,200,171,98,17,-29,-5,98,257,429,560,609,560,429,257,98,-5,-29,17,98,
171,200,171,98,17,-29,-5,98,257,429,560,609};

SQ15x16 y[101] = {};

int n;

void setup() {

  // put your setup code here, to run once:
  int16_t float_to_fixed(float input,int16_t fractional_bits);
  Serial.begin(2000000);

  noInterrupts(); // disable all interrupts
  TCCR5A = 0;
  TCCR5B = 0;
  TIMSK5=0;
  TCNT5 = 0;

  OCR5A = 500; // compare match register 16MHz/256/0.004s=500Hz
  TCCR5B |= (1<<WGM52); // CTC mode
  TCCR5B |= (1<<CS52) | (0<<CS51) | (0<<CS50); // 256 prescaler
  TIMSK5 |= (1<<OCIE5A); // enable timer compare interrupt
  interrupts(); // enable all interrupts
}
```



```
void loop() {
   if (n == 101){
      noInterrupts();
      delay(1000);
      for (int j = 0;j<101;j++){
        delay(10);
        Serial.print(static_cast<double>(y[j]));
        Serial.println(" ");
      }
      n+=1;
   }
}

//timer interrupt
ISR(TIMER5_COMPA_vect){
  for (n = 2; n < 101; n++){
    y[n] = static_cast<double>(FIR_Co[0]*x[n] + FIR_Co[1]*x[n-1] + FIR_Co[2]*x[n-2]);
  }
}
```