# EEC 134
# Application Note

## BASEBAND DESIGN AND IMPLEMENTATION

ROHAN PHADKE
997380053
TEAM HERTZ

## Introduction

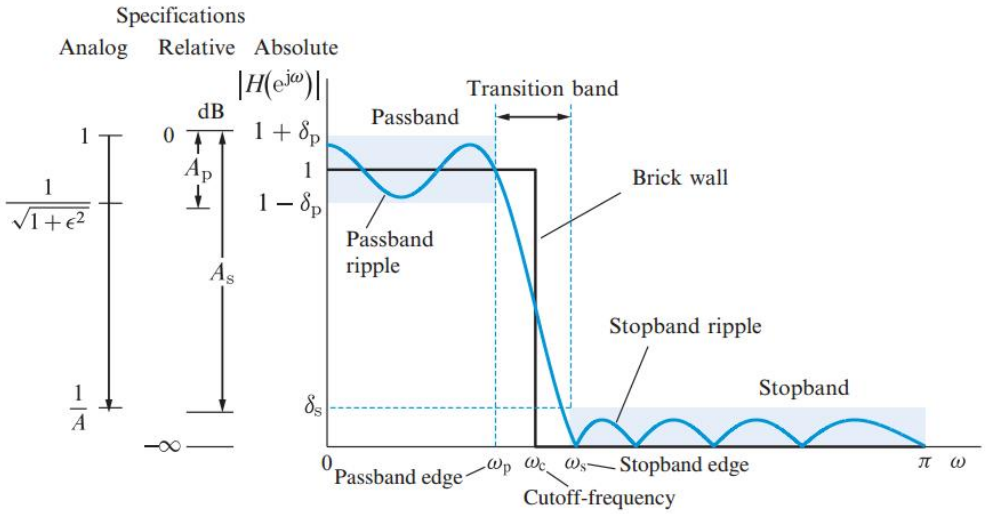
The primary objective of this application note is to outline in detail the process of completing the system level design and PCB layout for the baseband portion of a FMCW/Doppler radar. The baseband system will be analyzed stage by stage and each stage will be determined based on the specifications derived from the competition guidelines. Furthermore, the digital signal processing (DSP) portion of the baseband design will also be evaluated. Once the system level design is thoroughly explained the salient aspects of the PCB design will be discussed.

## Review of Essential Concepts

We briefly discuss some of the fundamental signal processing concepts used in the design of the baseband system, namely: filter design, windowing and sampling/reconstruction.

**Filter design**

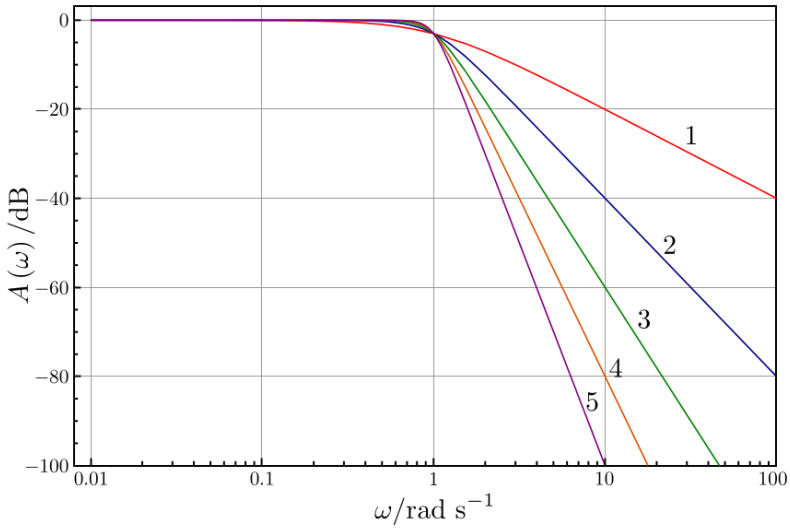
In a nutshell, filter design is the process of designing a system that is designed to pass a set of desired frequency components from a mixture of desired and undesired frequency components or to shape the spectrum of an input signal in a certain way. Filter design is largely governed by frequency response specifications such as magnitude/phase response, passband/stopband ripple, group delay, etc. The lowpass filter (LPF) is the most popular type of filter used, and a variety of other filters (highpass, bandpass, etc. filters) can be synthesized from a knowledge of the characteristics of the corresponding LPF. The order of the filter determines the amount by which the filter approximates the ideal brick wall response. These concepts are summarized in the diagram below which shows an example of the different kinds of specifications for a LPF.




Filters are generally optimized for a certain characteristic, and there are certain classes of filters based on this. For example, a Butterworth filter is the optimal filter if a maximally flat passband

response is desired; a Bessel filter is the optimal filter if a maximally flat group delay is desired. The Butterworth filter has good all-around performance which is adequate for many applications. It has a maximally flat passband magnitude response and a reasonable phase response that does not cause too much distortion.

The figure below shows the magnitude response of the Butterworth filter for various filter orders.

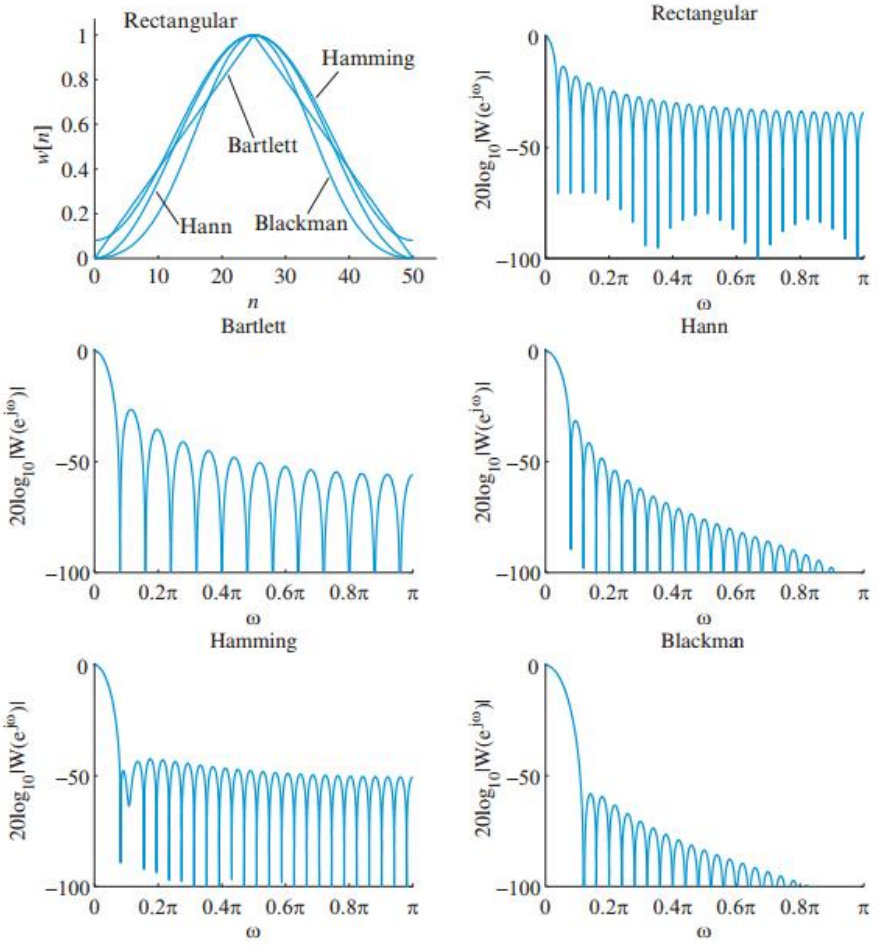



Due to the reasons discussed above, we implement our baseband circuit filter as a Butterworth filter.

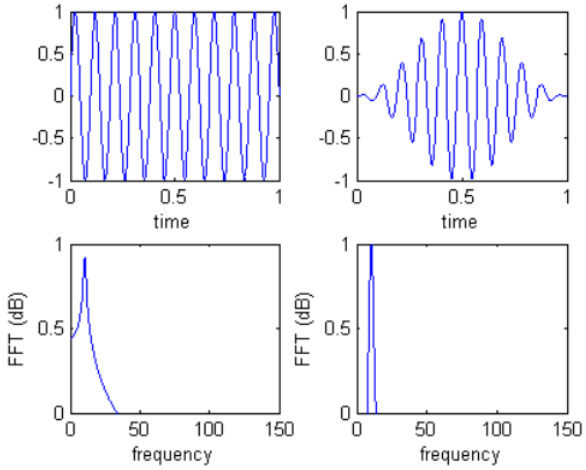
**Windowing**

Fast Fourier Transform (FFT) based measurements are subject to errors from an effect known as leakage. This effect occurs when the FFT is computed from a block of data that is not periodic (since the computation of the FFT assumes a block of data that is periodic). To correct this problem the appropriate windowing function must be applied. Windowing consists of multiplying the block of data by a finite length window with an amplitude that varies gradually and smoothly towards 0. This makes the endpoints of the block of data meet and results in a continuous waveform without sharp transitions, and also forces the data to be periodic.

There are several different types of window functions that you can apply depending on your signal and your requirements. To do this it is important to look at the frequency response of the window function as well as evaluate the frequency content of your signal. The diagram below shows the time domain and frequency domain response of some well known windows.




Generally a Hanning (Hann) window is satisfactory in almost 95% of all cases since it has good frequency resolution and reduced spectral leakage. For example, the picture below compares a non-periodic sine wave and its FFT with leakage (right) to the sine wave with a Hanning window applied to it and its FFT (left).



We can clearly see that after applying the Hanning window the spectral leakage is reduced considerably and the frequency domain representation of the sine wave more closely resembles the ideal response.

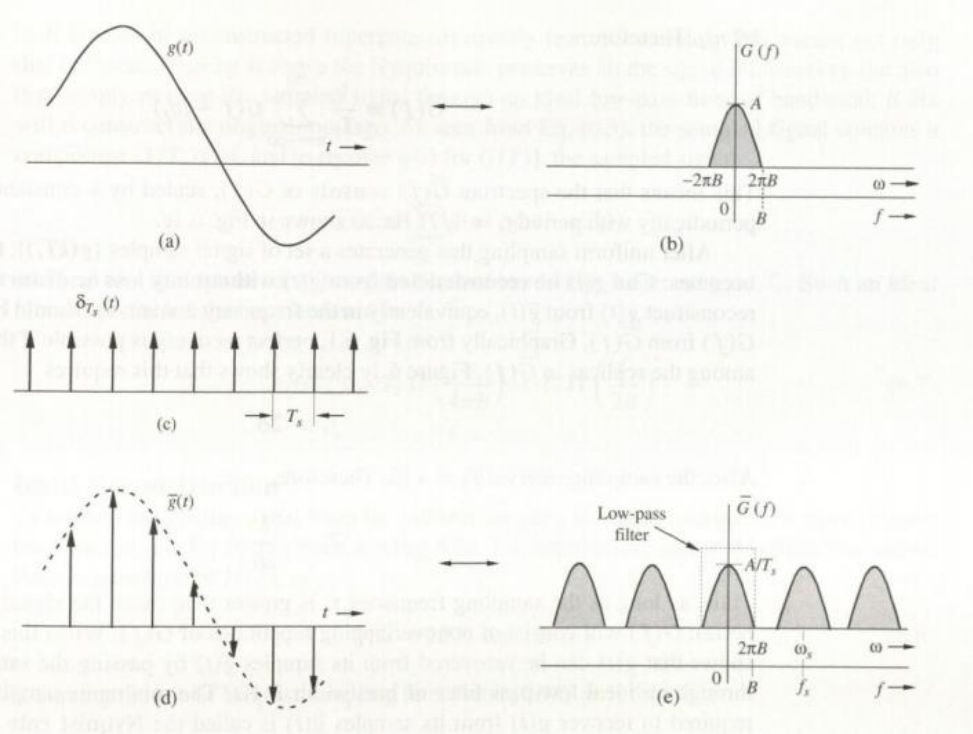
The main lobe of the Hanning window is at very low frequencies (close to 0 Hz). Therefore we can use it to "zoom" in on very low frequencies by applying it to a signal of very low frequency; this will effectively remove any high frequency components from the signal.

Due to the reasons discussed above, we choose to use the Hanning window in our Doppler radar measurements in order to narrow down the frequency band of interest to the Doppler frequency (which is very low, close to 1 Hz).

**Sampling/Reconstruction**

The continuous time signal obtained after filtering must be sent to a computer for further processing. However, a computer can only handle discrete data so the natural solution is to sample the continuous time signal at discrete points in time. The question then arises: how many samples should we take such that we can still uniquely describe the original continuous time signal and if so, how can it be reconstructed from its samples?

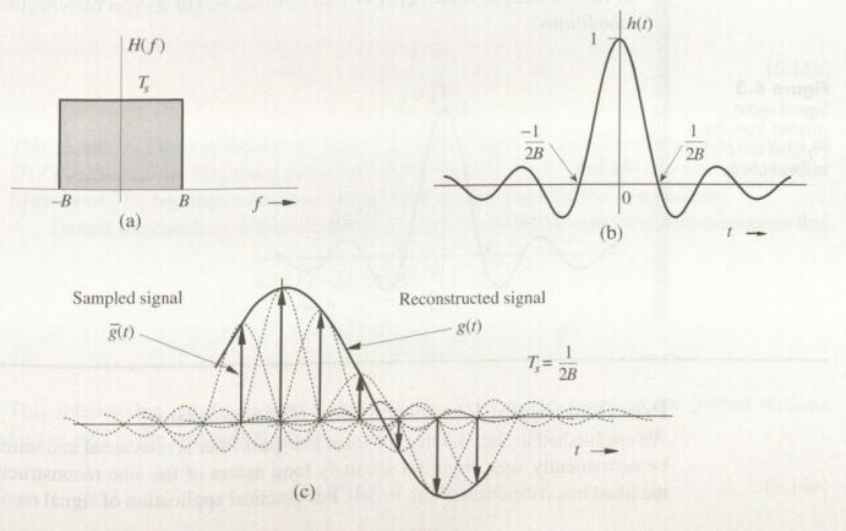
The key to this answer is the Nyquist-Shannon sampling theorem. The diagram below shows a continuous time signal $g(t)$ of bandwidth $B$ Hz (and is bandlimited) and its sampled version $\bar{g}(t)$ (sampled at a rate of $f_S$ Hz), and their frequency domain representations $G(f)$ and $\bar{G}(f)$ respectively.



We can see from the figure above that if $g(t)$ is sampled at a sampling rate $f_s$, then there are certain constraints on both the signal and the sampling rate. The first constraint is that $g(t)$ must be bandlimited, otherwise it will be impossible to find a suitable sampling frequency. Furthermore, from the plot of $\bar{G}(f)$ we can see that if $f_S < 2B$ then the frequency spectra copies will overlap with each other, resulting in aliasing and distortion. However, if $f_S \geq 2B$ then the signal will not be distorted and we will be able to recreate it.

The above explanation is essentially the core of the Nyquist-Shannon sampling theorem. The theorem states that for a bandlimited signal, a signal must be sampled at least twice as fast as the bandwidth $B$ of the signal to accurately reconstruct the waveform. Otherwise, the high-frequency content will alias (overlap) at a frequency inside the spectrum of interest. The Nyquist rate is defined as the minimum rate required to achieve the conditions above; in this case the Nyquist rate would be $2B$.

If a signal is sampled at or above the Nyquist rate it is possible to reconstruct the analog signal from its samples simply by passing it through an ideal lowpass filter of bandwidth $B$ Hz. The ideal lowpass filter will truncate all frequency domain copies except that of the original signal, thus allowing you to exactly recover the original signal from its samples. A picture of this process is shown below.




As will be shown later, the frequencies that we are dealing with will at most be 5 kHz. The filtered signal is sent to the computer via a sound card which has a sampling rate of 44.1 kHz. This sampling rate is far greater than the bandwidth of the maximum signal frequency possible (the resulting bandwidth is 5 x 2 = 10 kHz). Therefore we have ensured that the signal will not be distorted by oversampling.

## Baseband System Design

This section will outline the process of the baseband design. We begin by determining the amplitude of the signal at the output of the mixer as well as the IF frequencies possible for both tests outlined in the competition guidelines. This will give us a sense of how much gain to provide in order to ensure good signal resolution, and what cutoff frequency to use to remove unwanted harmonics or high frequency components. Next, we discuss how to realize the gain and filter stage based on the conclusions made in the previous step. Finally, we evaluate the code used to extract information from the amplified and filtered signal.

### Design specifications

The two main design specifications for the baseband design were the amplitude of the incoming signal and the IF frequency.

The first step is to determine the expected amplitude of the received signal. The RF link budget for the received signal at various distances is shown below.

**Finalized Link Budget (Receive Power)**

| Range | Power Received | Mixer Output Power | Mixer Output Voltage |
|---|---|---|---|
| 1m | -21.58dBm | -3.53dBm | 420.74mVpp |
| 10m | -65.53dBm | -43.53dBm | 4.21mVpp |
| 25m | -81.44dBm | -59.44dBm | 673.78uVpp |
| 50m | -93.28dBm | -71.48dBm | 168.47uVpp |

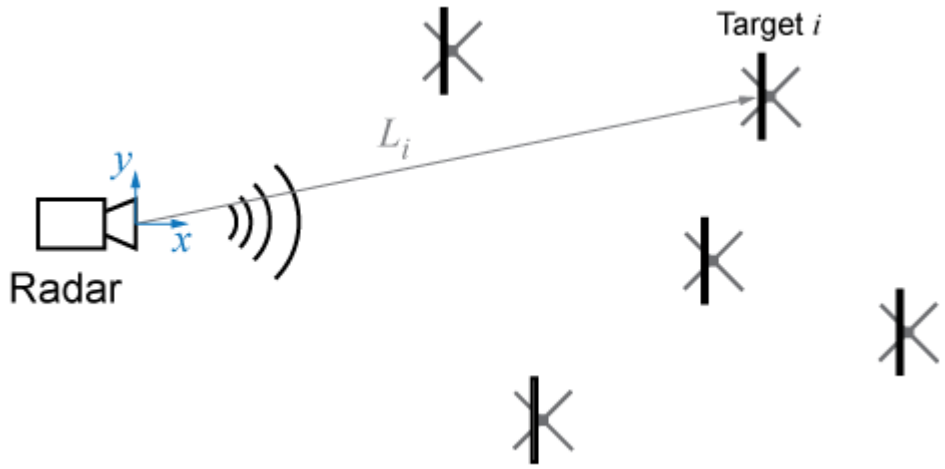
From the table above we can see that the amplitude of the signal entering the baseband system drastically drops as distance increases. We decided to optimize our system for a shorter range. The intensity $I$ of a signal propagating in free space varies with distance from the source $r$ as $I \propto 1/r^2$. Furthermore the amplitude $A$ is related to the intensity by $A^2 \propto I$. Thus we can conclude that $A \propto 1/r$.

Based on this inverse relationship, we decided to optimize our system for a distance of approximately 4-5 m, so the amplitude of the signal will be in the range of 420.74/5 – 420.74/4 ≈ 85-105 $mV_{pp}$. As a reasonable estimate, we decided to design our system assuming an input signal amplitude of roughly 100 $mV_{pp}$.

The next step is to determine the IF frequency for both tests to determine the cutoff frequency of the LPF. This is done below.

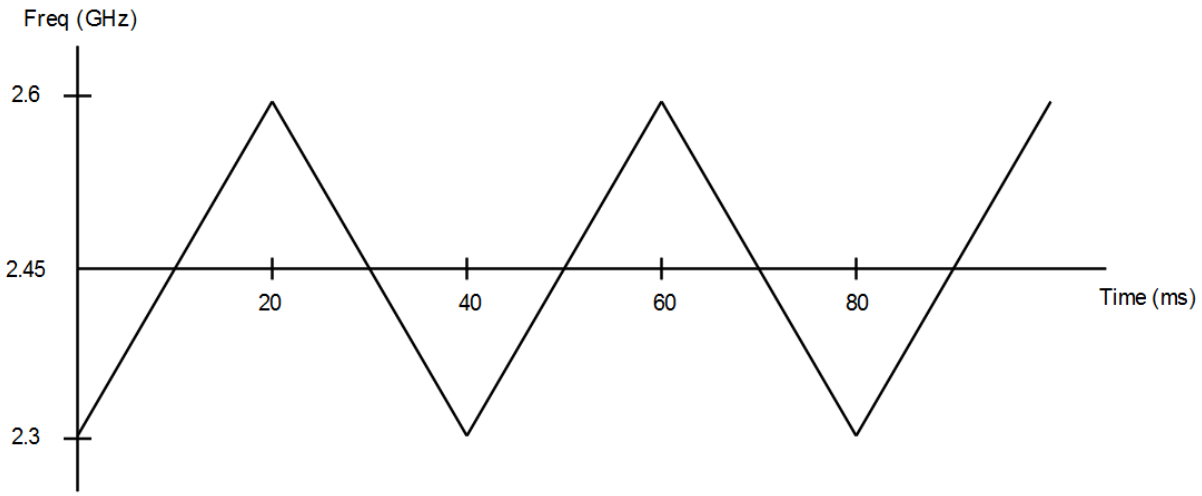
1) Test 1: Fixed Targets

In this test, the FMCW radar is used to measure the distance to various targets. The targets under consideration are 0.3 x 0.3 $cm^2$ metal plates mounted on wooden stands. The radar is configured to work in the FMCW mode. A diagram of this scenario is shown below.




Let $R$ be the range of any given target (for example, the $i^{th}$ target). Then according to the competition guidelines $R_{min} = 5$ m and $R_{max} = 50$ m.

Since the radar is working in the FMCW mode, the voltage of the VCO input is continuously ramping up and down linearly. As a result, the frequency of the output of the VCO (the transmit signal) is also ramping up and down, with modulation rate $f_m$. The plot of frequency modulation vs. time is shown below.




From this plot we can see that the period of modulation $T_m = \frac{1}{f_m} = 40$ ms – this is the period of 1 cycle of the triangle waveform due to the voltage ramping from the modulator circuit. We can also see that for $\Delta t = \frac{T_m}{2} = 20$ ms, $\Delta f = 0.3$ GHz.

The time $t_R$ required for the transmitted signal to make the round trip is $t_R = \frac{2R}{c}$. After time $t_R$ has elapsed, the transmit frequency will be $f_R$. Since the frequency is linearly ramped, we have

$$\frac{\Delta f}{\Delta t} = \frac{f_R}{t_R} \longrightarrow f_R = \frac{\Delta f}{\Delta t} t_R = \frac{\Delta f}{\Delta t} \frac{2R}{c}$$

So we have

$$f_R = \frac{(0.3 \times 10^9)}{(20 \times 10^{-3})} \cdot \frac{2R}{3 \times 10^8}$$

$$= 100R \text{ (Hz)}$$

Now, let $f_1$ be the transmit frequency at time $t = 0$ and $f_2$ the transmit frequency at time $t = t_R$. Then it follows that $f_2 = f_1 + f_R \longrightarrow f_R = f_2 - f_1$. Therefore we can conclude that $f_R = 100R$ (Hz) is the desired IF frequency.

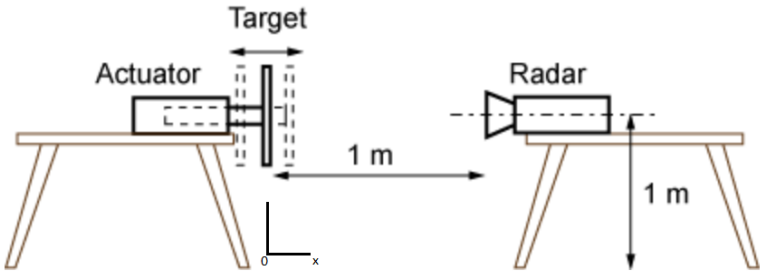
Now, we determine the minimum and maximum IF frequencies possible. To do this, we substitute $R = R_{min}$ and $R = R_{max}$ in the expression for $f_R$. The results are shown in the table below.

| **Range $R$ (m)** | **IF frequency (kHz)** |
|---|---|
| $R_{min} = 5$ | 0.5 |
| $R_{max} = 50$ | 5 |

It is prudent to set the cutoff frequency $f_c$ of the LPF to be somewhat higher than 5 kHz since we do not want any attenuation at 5 kHz. A reasonable estimate for the cutoff frequency of the baseband LPF would therefore be $f_c = 10$ kHz.

2) Test 2: Periodic movement

In this test the radar is to be set to Doppler mode and used to determine the amplitude and frequency of a moving target. This is done by detecting the phase change of the return signal due to the movement of the target. In this case the frequency is no longer ramped up and down and is kept constant instead. The moving target in this case consists of a moving metallic plate with dimensions 10 x 10 cm$^2$ mounted on a linear actuator located exactly 1 m away from the antenna of the radar. A diagram of this scenario is shown below.




The actuator is programmed to move the target back and forth with amplitude $A$ (with $1 \leq A \leq 10$ mm) and frequency $f$ (with $0.2 \leq f \leq 0.8$ Hz). The profile of the movement is either triangular or sinusoidal. Therefore from these competition guidelines, we see that the range $R = 1$ m along with $A_{min} = 1$ mm, $A_{max} = 10$ mm and $f_{min} = 0.2$ Hz, $f_{max} = 0.8$ Hz.

We now develop an expression for the Doppler frequency $f_d$ that will allow us to determine a minimum and maximum IF frequency. This is because when the received signal and LO are mixed the resulting signal has a frequency equal to $f_d$, the Doppler shift resulting from the movement of the target. Hence the Doppler shift is in fact the IF frequency.

We make the following variable definitions:

- $R$ – the distance from the radar to the target
- $f_0$ – the transmit frequency
- $v_r$ – the relative velocity between the radar and the target. Since the radar is not moving, $v_r$ is equal to the velocity of the oscillating target
- $x$ – the position of oscillation (this is defined on the figure above)

We know that if $R$ is the distance from the radar to the target, then the total number of wavelengths contained in the round trip between the radar and target is $2R/\lambda_0$. Therefore the phase change $\phi$ is given by

$$\phi = 2\pi \frac{2R}{\lambda_0}$$

The factor of $2\pi$ is introduced due to the fact that each wavelength corresponds to a phase change of $2\pi$.

If the target is in relative motion with the radar then $R$ and $\phi$ are continuously changing. We can find the Doppler angular frequency shift $\omega_d$ by using the relation $\omega_d = \frac{d\phi}{dt}$. Therefore we have

$$\omega_d = 2\pi f_d = \frac{d\phi}{dt} = \frac{4\pi}{\lambda_0}\frac{dR}{dt} = \frac{4\pi}{\lambda_0} v_r$$

As a result,

$$f_d = \frac{2}{\lambda_0} v_r$$

$$= \frac{2v_r}{c} f_0$$

Now, we assume that the profile of the movement is sinusoidal in all cases. Even if the movement is triangular it can be approximated as sinusoidal because of the low frequency and amplitude of oscillation.

Since the movement is a periodic, sinusoidal function of time we can describe the motion as $x = A\cos(2\pi f t + \theta)$. We can assume that $\theta = 0$ which is equivalent to assuming that the target starts at the equilibrium position. Then we have $x = A\cos(2\pi f t)$ and so

$$v_r = \frac{dx}{dt} = -2\pi f A \sin(2\pi f t)$$

Therefore the Doppler frequency (and therefore the IF frequency) $f_d$ is

$$f_d = \frac{2[-2\pi f A \sin(2\pi f t)]}{c} f_0 = -4\pi f A \sin(2\pi f t)\left(\frac{c}{f_0}\right)$$

In our design, our transmit frequency was $f_0 = 2.4\ \text{GHz}$. Then we have

$$f_d = -4\pi f A \sin(2\pi f t)\left(\frac{3 \times 10^8}{2.4 \times 10^9}\right)$$

$$= -32\pi f A \sin(2\pi f t)$$

Based on this expression we see that $0 \leq |f_d| \leq 32\pi fA$. Therefore the minimum IF frequency possible is 0 Hz (DC) and the maximum IF frequency is $(32\pi fA)$ Hz.

We will now substitute the different minimum and maximum amplitudes ($A_{min}, A_{max}$) and frequencies ($f_{min}, f_{max}$) in the expression for $f_d$ to determine a minimum and maximum IF frequency. The results are shown in the table below.

| Frequency (Hz) | Amplitude (mm) | IF frequency (Hz) |
|---|---|---|
| $f_{min} = 0.2$ | $A_{min} = 1$ | 0.0201 |
| | $A_{max} = 10$ | 0.2011 |
| $f_{max} = 0.8$ | $A_{min} = 1$ | 0.0804 |
| | $A_{max} = 10$ | 0.8042 |

Therefore we can see that the IF frequency will range from approximately 0 Hz (DC) to 1 Hz.

**System realization**

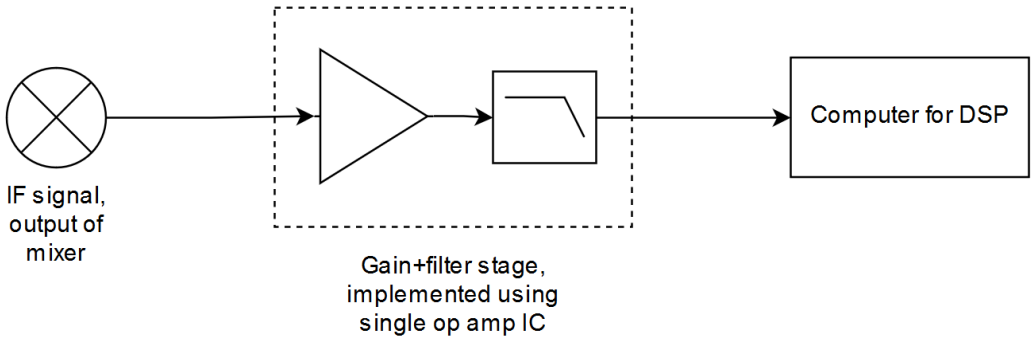
Based on the calculations shown above, a gain and filter stage was designed to process the incoming signal, which is the IF signal produced by the mixer. The gain stage was introduced before the filter in order to amplify the signal a sufficient amount before filtering. The LPF output was then connected to a computer through an audio jack.

The sound card can accurately resolve an incoming signal if the amplitude is around 1-2 $V_{pp}$. Since we estimated that the input amplitude would be around 100 $mV_{pp}$, we need a gain of approximately 10. Furthermore we determined that the maximum range possible in the range test (Test 1) is 5 kHz and the maximum range possible in the Doppler test (Test 2) is 1 Hz. Therefore we need to design our filter to have a cutoff frequency $f_c$ somewhere around 10 kHz, and in the case of the Doppler test employ additional DSP techniques like windowing to accommodate for the extremely low signal frequency.

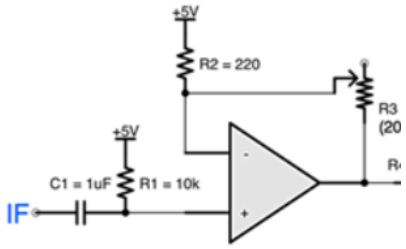
Now that the fundamental building blocks of the baseband system have been introduced we now discuss the two iterations of the baseband system. The first iteration was conceptually simpler to understand and did not require any additional ICs. The second iteration was slightly more complicated and required an additional IC, but was vastly superior to the first iteration.

1) Iteration 1

The block diagram for this iteration of the gain+filter stage is shown below.




In this iteration, both the gain and filter stage were implemented using op amps. The gain stage consisted of two noninverting op amps in cascade. The circuit diagram for this is shown below.

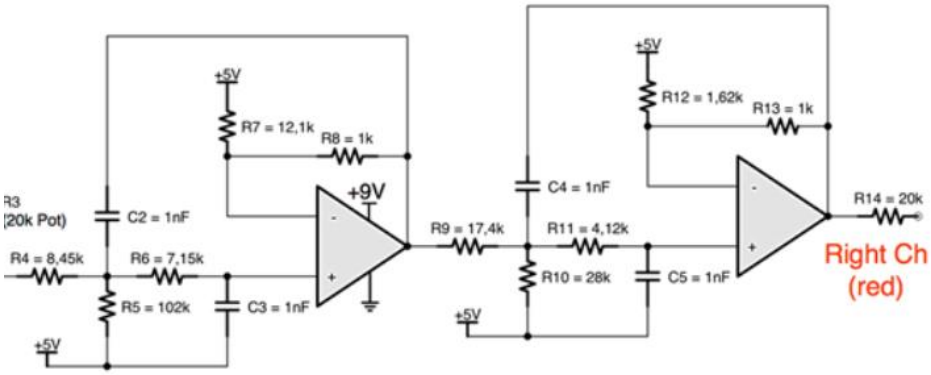



The gain $G$ of such a noninverting op amp configuration is given by

$$G = 1 + \frac{R_{N+1}}{R_N}$$

where $N$ is the identifying number of the feedback resistor connected directly to the inverting pin of the op amp and directly to ground (in the diagram above this would be $R_2$, so $N = 2$). Although not shown in the figure above capacitors were connected at the input and output of each substage. This provided AC coupling and removed any DC bias; this was important to avoid clipping the signal.

The filter stage was a Sallen-Key 4$^{th}$ order active lowpass filter. This 4$^{th}$ order active LPF was constructed using two 2$^{nd}$ order active LPFs in cascade. The circuit diagram for this stage is shown below:




The gain of each stage of the active LPF is the same as that of a noninverting op amp. The cutoff frequency $f_c$ of each 2nd order substage is given by

$$f_c = \frac{1}{2\pi\sqrt{R_N R_{N+2} C_M C_{M+1}}}$$

where $N$ is the identifying number of the feedback resistor connected directly to the inverting pin of the op amp and directly to ground, and $M$ is the identifying number of the feedback capacitor (in the first substage in the diagram above this would be $R_4$ and $C_2$ so $N = 4$ and $M = 2$ respectively).

This gain+filter stage was designed keeping the gain of the overall stage and the cutoff frequency in mind. Each individual noninverting op amp substage and 2nd order LPF substage provided gain. The gain of the noninverting op amp was adjustable through the use of a potentiometer (pot). The pot would allow us to vary the ratio $\frac{R_{N+1}}{R_N}$ of the substage. The pot was adjusted to give an overall gain of 10. The gain of the filter stages was kept at unity. The overall gain of the stage was simply the product of these individual gains. The cutoff frequency of the two 2nd order LPF was kept close to 20 kHz.

This realization of this gain+filter stage is conceptually simple to understand. It is possible to implement it using only one op amp IC (containing 4 individual op amps) and the required passives. This reduces the number of ICs required and reduces the current draw of the overall system. However, this implementation of the gain+filter stage has a multitude of problems. Some of these include:

a) Lack of flexibility

   We can see from the design equations for the LPF that the cutoff frequency $f_c$ is determined by the passive components. This means that to change $f_c$ multiple passive components would need to be replaced. This is impractical in terms of time and could potentially damage the board due to repetitive assembly procedures. Thus the ability to easily change $f_c$ is greatly hindered.

b) Complexity of assembly

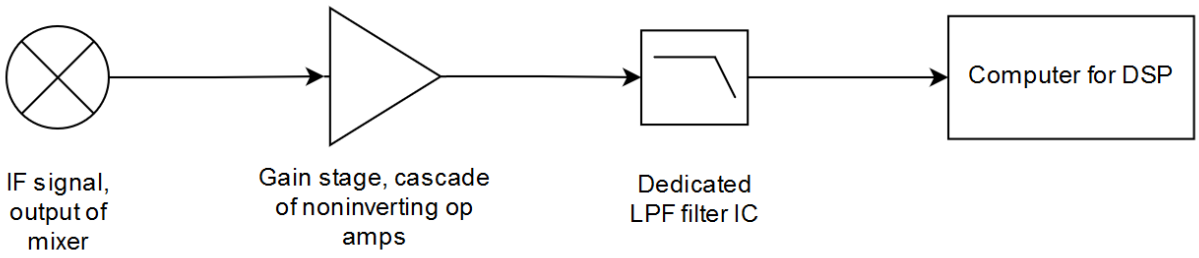
Although the circuit is simple to understand, actually implementing it on a PCB is quite difficult. Due to the form factor of the op amp IC used (specifically, TI OPA4228) it is hard to perform the layout of the passives. They need to be grouped together tightly in order to be laid out properly, meaning that there is not much space to place anything else. Another consequence of this is that it is much harder to place test points at critical parts of the circuit. This means that it is harder to debug the analog portion of the baseband circuit.

c) Parasitics and degradation of signal

Another major concern of placing passives close together is the potential introduction of parasitics as well as the degradation of the signal. Due to the close proximity of the passives, stray capacitances and inductances can easily be introduced even at low frequencies. It is also entirely possible that the signal traveling through one branch of the circuit will interfere with the other, resulting in degradation of the signal. The cascade of op amps also introduces losses and noise in the signal.

2) <u>Iteration 2</u>

The block diagram for this iteration of the gain+filter stage is shown below.



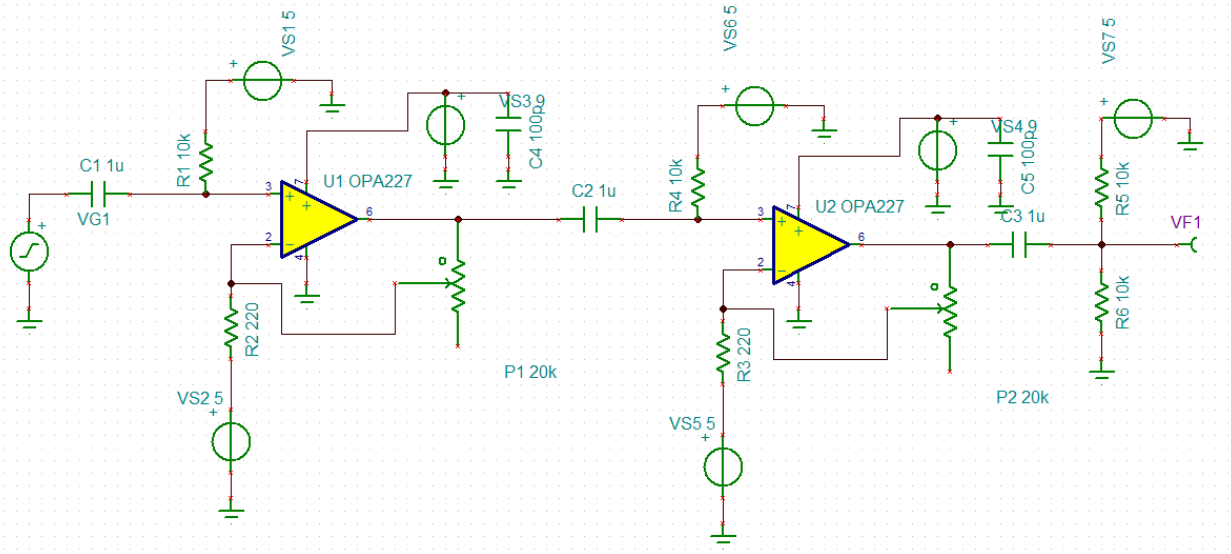

In this iteration the gain and filter stage are separated. The gain section is kept the same as before and a dedicated filter IC is used as the LPF. The cutoff frequency of the filter IC was set to be approximately 12 kHz by using a 22 pF capacitor, as outlined in the datasheet. An extra pad for another capacitor allowed for further flexibility of cutoff frequency adjustment. Furthermore, both the op amp IC and filter IC were operated using single supply techniques.

The advantages of this implementation of the gain+filter stage directly address the drawbacks of the first iteration. The filter IC chosen (Maxim Integrated MAX291) has an adjustable cutoff frequency $f_c$; it can be set either by an external clock signal or external capacitors in conjunction with the built-in internal clock. The filter order is also much higher ($8^{th}$ order Butterworth filter). The complexity of assembly is greatly reduced as the IC

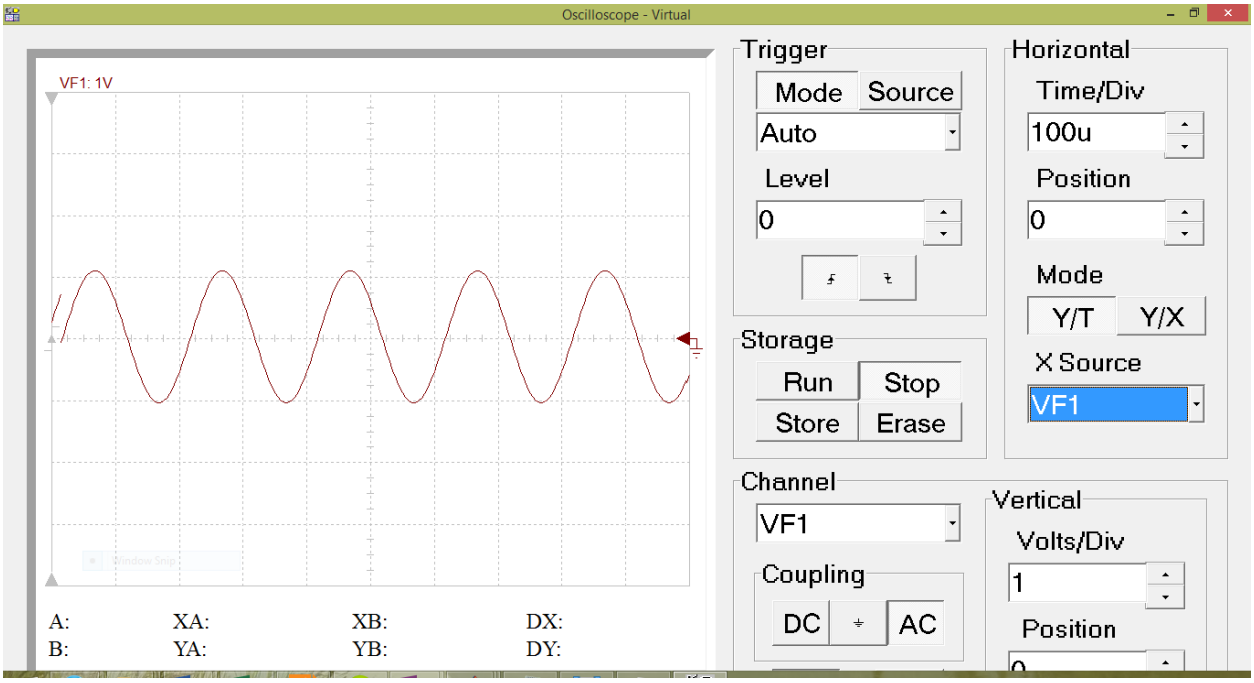
requires no external resistors and capacitors by design. This opens up room to place test points at critical junctions of the circuit to test performance. In addition, since the entire filter is on-chip, parasitics due to passives and degradation of the signal due to crosstalk and noise introduced by op amps is highly diminished.

Due to the reasons mentioned above Iteration 2 was the iteration implemented in the final baseband design. In order to ensure that the iteration would work well, we used the TINA-TI SPICE simulator, a powerful circuit design and simulation tool. TINA-TI is ideal for designing, testing, and troubleshooting a broad variety of basic and advanced circuits, including complex architectures, without any node or number of device limitations.

We simulated the gain stage response using TINA; however we could not simulate the filter response since the component was not available in the component library. This was not a problem however since the performance of the filter was easy to predict based on The op amp used for this simulation was the TI OPA2227 (this choice of op amp will be discussed later). This would give us an idea of how well the gain circuit would work. The TINA schematic is shown below.




The pots were adjusted to give a resistance of 500 Ω. This would mean that for a 100 $mV_{pp}$ signal the overall gain would be $\left(1+\frac{500}{220}\right)^2 = 10.71 \approx 10$, resulting in an output amplitude of approximately 1 $V_{pp}$. The simulation results are shown below.



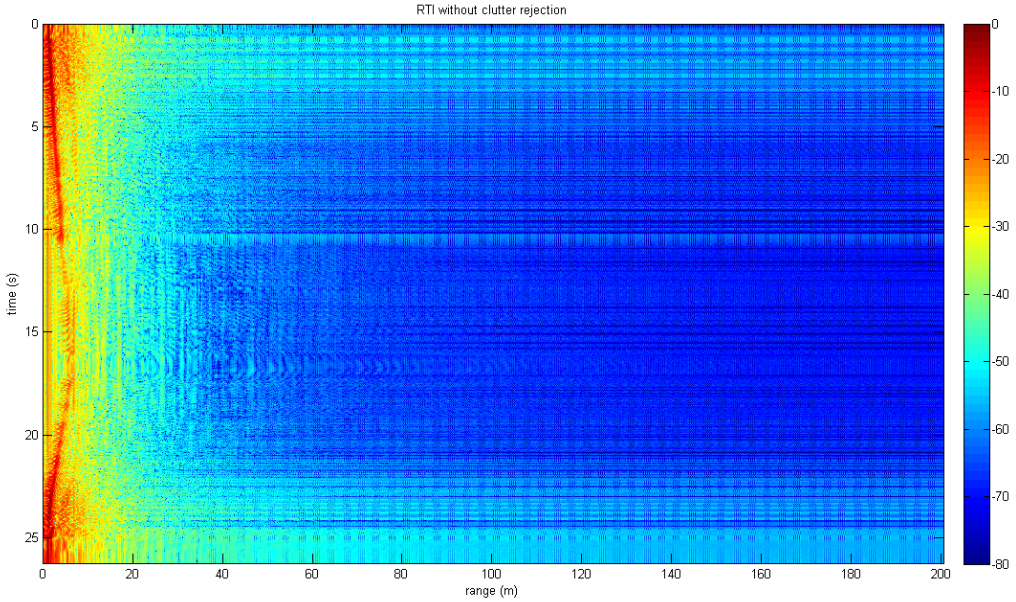

We can see from the picture above that the amplitude of the gain stage output is approximately 1 $V_{pp}$ which is exactly what was desired. This simulation therefore gives us confidence in the fact that the gain circuit will work well.

**DSP code**

Once the signal was appropriately amplified, filtered it was sent to the computer via an audio jack and sampled using the sound card. The sampling rate of the sound card is 44.1 kHz which is more than enough to avoid aliasing. After that, the signal processing was performed on MATLAB. Different code was used to determine the required information (range or frequency/amplitude) based on the test performed. We now discuss the code used for the range test (Test 1) and the Doppler test (Test 2).

1) <u>Test 1: Fixed Targets</u>

   To determine the distance to a fixed target, the radar system was set up to transmit and receive, and the output at the audio jack was saved into a .wav file using Audacity. This .wav file was later opened and processed using MATLAB. RTI clutter rejection was used to remove unwanted echoes from other objects in the vicinity. A sample plot output is shown below. The red trace shows the distance at which the object is present.




The code we used to generate our plots is shown below, and the explanation follows.

```
%MIT IAP Radar Course 20112.5
%Resource: Build a Small Radar System Capable of Sensing Range, Doppler,
%and Synthetic Aperture Radar Imaging
%
%Gregory L. Charvat

%Process Range vs. Time Intensity (RTI) plot

clear all;
close all;

% read the raw data .wav file here
% replace with your own .wav file
[Y,FS,NBITS] = wavread('radar_range.wav');

%constants
c = 3E8; %(m/s) speed of light

%radar parameters
Tp = 20E-3; %(s) pulse time
N = Tp*FS; %# of samples per pulse
fstart = 2260E6; %(Hz) LFM start frequency
fstop = 2590E6; %(Hz) LFM stop frequency
BW = fstop-fstart; %(Hz) transmti bandwidth
f = linspace(fstart, fstop, N/2); %instantaneous transmit frequency

%range resolution
rr = c/(2*BW);
max_range = rr*N/2;
```

```
%the input appears to be inverted
trig = -1*Y(:,1);
s = -1*Y(:,2);
clear Y;

%parse the data here by triggering off rising edge of sync pulse
count = 0;
thresh = 0;
start = (trig > thresh);
for ii = 100:(size(start,1)-N)
    if start(ii) == 1 & mean(start(ii-11:ii-1)) == 0
        %start2(ii) = 1;
        count = count + 1;
        sif(count,:) = s(ii:ii+N-1);
        time(count) = ii*1/FS;
    end
end
%check to see if triggering works
% plot(trig,'.b');
% hold on;si
% plot(start2,'.r');
% hold off;
% grid on;

%subtract the average
ave = mean(sif,1);
for ii = 1:size(sif,1);
    sif(ii,:) = sif(ii,:) - ave;
end

zpad = 8*N/2;

%RTI plot
figure(10);
v = dbv(ifft(sif,zpad,2));
S = v(:,1:size(v,2)/2);
m = max(max(v));
imagesc(linspace(0,max_range,zpad),time,S-m,[-80, 0]);
colorbar;
ylabel('time (s)');
xlabel('range (m)');
title('RTI without clutter rejection');

%2 pulse cancelor RTI plot
figure(20);
sif2 = sif(2:size(sif,1),:)-sif(1:size(sif,1)-1,:);
v = ifft(sif2,zpad,2);
S=v;
R = linspace(0,max_range,zpad);
for ii = 1:size(S,1)
    %S(ii,:) = S(ii,:).*R.^(3/2); %Optional: magnitude scale to range
end
S = dbv(S(:,1:size(v,2)/2));
m = max(max(S));
imagesc(R,time,S-m,[-80, 0]);
colorbar;
ylabel('time (s)');
```

```
xlabel('range (m)');
title('RTI with 2-pulse cancelor clutter rejection');
```

We will explain the code block by block. The first block reads the .wav file and stores the recorded data, the sampling rate and the number of bits per sample in variables. The next block sets the radar parameters such as pulse rate and range resolution. The `fstart` and `fstop` variables are the VCO frequencies corresponding to 0 and 5 V respectively. The next block divides the input data matrix into two vectors – one for the SYNC (trigger) data and the other for the measured signal data.

The next block parses the data by triggering off rising edge of SYNC pulse. The for loop ranges from an arbitrary start time and end time and searches for a SYNC pulse. The SYNC pulse data is used as a trigger to properly sync measurements to the rising edge trigger [`start = (trig > thresh);`]. Next, if the SYNC pulse is high and if it was not high at this point previously (thus establishing that the point we are at is truly a rising edge) [`if start(ii) == 1 & mean(start(ii-11:ii-1)) == 0`] then the data is parsed. The data is then stored in the `sif` matrix with `count` as the counter used to input the signal data into the appropriate row. Then the time corresponding to the rising edge of the SYNC is recorded using the loop counter and the sampling period [`time(count) = ii*1/FS;`].

After that, the average DC term is eliminated from the data by subtracting the average. After that the data is converted to the frequency domain using the FFT algorithm. The maximum amplitude is then found. The data is then plotted without clutter rejection by using the `imagesc` function. The 2-pulse cancelled data is then obtained by subtracting each row of the `sif` matrix by the row before it. The maximum amplitude is once again found. The FFT of this data is obtained and then plotted with clutter rejection using the same `imagesc` function as before.

2) Test 2: Periodic movement

For Doppler radar testing real time code was used. This real time code allowed one to observe the IF frequency obtained in real time. The code is posted below, and the explanation follows.

```
%This code originated from sample code given by Gregory Charvat. It's been
heavily 'developed' into the code
%that you see now primarily by Zach Myers. However intermediate versions
were developed by Jhonnaton Ascate
% and Christopher Young.
clear all;
close all;

%constants
```

```
dbv = @(x) 20*log10(abs(x));
c=3E8; %(m/s) speed of light

%radar parameters
FS = 44.1E3;
Tp = 0.1; %(s) pulse time
N = 8000; %# of samples per pulse
fc = 2590E6; %(Hz) Center frequency (connected VCO Vtune to +5)
recordLength = 0.20;

%filters:
%K: Used to filter incoming signal
%K = Test23;
%K3: Used to window receive signal to reduce sidelobe noise
K3 = hanning(100);


%Recording Setup
r = audiorecorder(44100,16,2);
record(r);
pause(recordLength);
stop(r)
Y= getaudiodata(r);

%Set up shift register buffer
bufferVel = [];
bufferSize = 30;
bufferPosition = 0;

%All of the plots are initialized ahead of time and the set function is
used
%later on instead of recalling plot

%Calling plot continuously is inefficient because it reinitializes memory
whereas
%set simply changes one or two arrays

%Output of Dopplar Radar AFTER windowing
figure(1);
H = plot(Y);
H1 = axis;
ylabel('Amplitude of Doppler Shift (volts)');
xlabel('Time (sec)');
title('Doppler Radar - Received Signal (After Filtering)');
ylim([-5 5]);
xlim([0.005 0.08]);
C = 1;

%Waterfall-ish Diagram
figure(2);
H2 = mesh([0 1;2 3]);
ylim([1 Tp*bufferSize]);
xlim([0 50]);
zlim([-140 10]);
xlabel('Velocity (m/sec)');
ylabel('Time (sec)');
title('Doppler Radar - Velocity');
```

```
%Incredibly Rough Peak Detection Algorithm Display
figure(3);
H3 = plot(0,0);
title('Shift Buffer of Maximum Readings over -40 dB');
xlabel('Position in Buffer');
ylabel('Approximate Velocity (m/s)');

highspeed = zeros(bufferSize,1);
while 1,
    %While everything is processing, have the audio recorder record audio
- This is basically a ping pong buffer - one buffer is processed
    %as another buffer is filled - although MATLAB obsfucates what buffer
is being filled with the recorder wrapper class
    record(r);

    %the input appears to be inverted <- MIT comment
    s = -1*Y(:,2);

    %The signal is windowed by a hanning filter
    K3 = hanning(length(s));
    s = s.*K3;
    %s= filter(K, s);

    %This displays the windowed data to one graph
    set(H, 'YData', s(3:end), 'XData', (3:1:size(s,1))/FS);

    %create doppler vs. time plot data set here
    sif(1,:) = s(1:N);

    %subtract the average DC term here
    sif = sif - mean(s);
    zpad = 8*N/2;

    %doppler vs. time plot:
    %This takes the fourier transform of the signal - I am not sure why
MIT chose to use the IFFT over the FFT
    v = dbv(ifft(sif,zpad,2));
    v = v(:,1:size(v,2)/2);

    %This section attempts to do a very rudimentary form of peak detection
based on the top 3 highest signals
    A = sort(v(3:end), 'descend');
    %The three dopplar shifts that have the largest reflection are
averaged together - this might require significant tweaking
    %as there is significant near-DC components that need to be accounted
for
    B(1) = find(v == A(1));
    B(2) = find(v == A(2));
    B(3) = find(v == A(3));
    A2 = mean(A(1:3));

    %This whole section could be optomized by manually tracking increase
in
    %size - however it was not necessary due to the large amount of
    %processing power
```

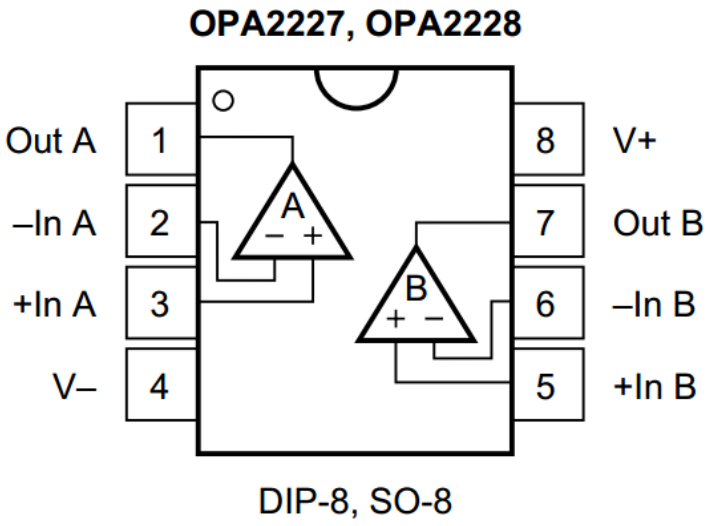
```
    %In fact, this loop does not occupy enough time so I had to insert a
delay later inorder for the number of samples accumulated to be
appropiate.
    bufferVel = [bufferVel; v];
    if size(bufferVel,1) > bufferSize;
        bufferVel = bufferVel((size(bufferVel,1)- bufferSize + 1):end,:);
    end

    %calculate velocity
    delta_f = linspace(0, FS/2, size(bufferVel,2)); %(Hz)
    lambda=c/fc;
    velocity = delta_f*lambda/2;
    avePeakVelo = (velocity(B(1))*A(1) + velocity(B(2))*A(2) +
velocity(B(3))*A(3))/(sum(A(1:3)));

    %calculate time
    time = linspace(1,Tp*size(bufferVel,1),size(bufferVel,1)); %(sec)

    %plot
    set(H2, 'XDATA',velocity(3:700), 'YDATA', time(1:end), 'ZDATA',
bufferVel(1:end,3:700));
    if A2 > -52,%this determines what is considered a detection - it is
set manually (which is bad due to environment dependencies).
        highspeed = [highspeed; avePeakVelo];
        if size(highspeed,1) > bufferSize;
            highspeed = highspeed((size(highspeed,1)- bufferSize +
1):end,:);
        end
        set(H3, 'XDATA', 1:1:bufferSize, 'YDATA', highspeed);
    end

    clear SS S v sif ave count start thresh s trig time velocity;
    %Here is where the delay comes into play - after all the processing is
done
    pause(recordLength);
    %Stop recording and saving the audio data into Y
    stop(r);
    Y = getaudiodata(r);
end
```

This code processes the real time data using the concept of a shift buffer. Essentially what it does is that it first windows the incoming signal and displays this windowed data. It then stores a certain amount of values into a buffer vector. Then, the FFT of the data is taken, the velocity is calculated and the maximum velocity is found using a rudimentary form of peak detection. All this data is then plotted. After plotting, the values are updated by “pushing” out the old values and “pushing” in the next set of values.

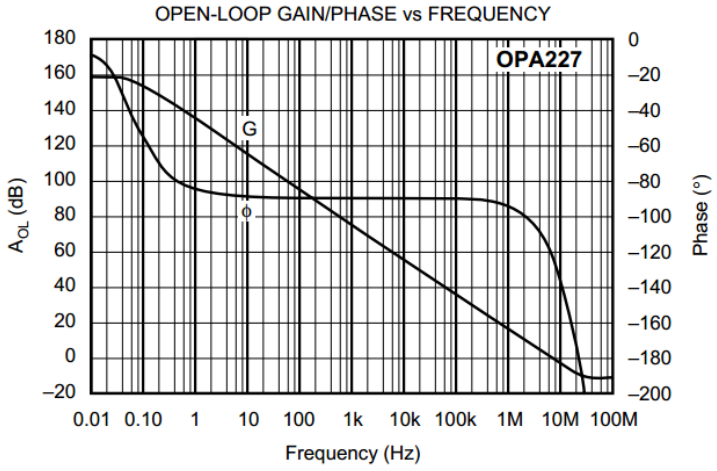
## Baseband Component Selection

After designing the baseband system the next step is to select the appropriate components. Based on Iteration 2, the two main components to be selected are the op amp and the filter IC. The supporting passives to implement these circuits are also required. We will discuss each of these in turn.

**Op amp selection – TI OPA2227**

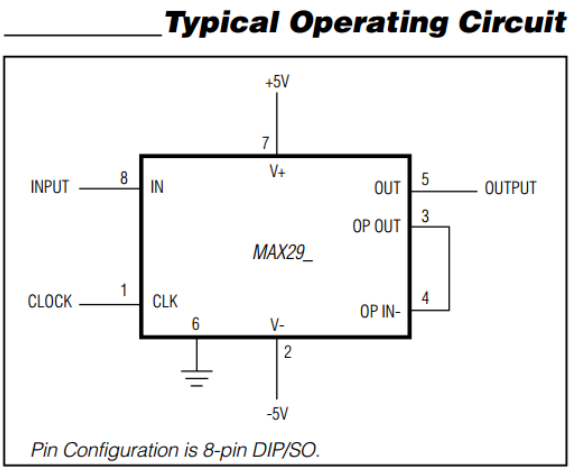


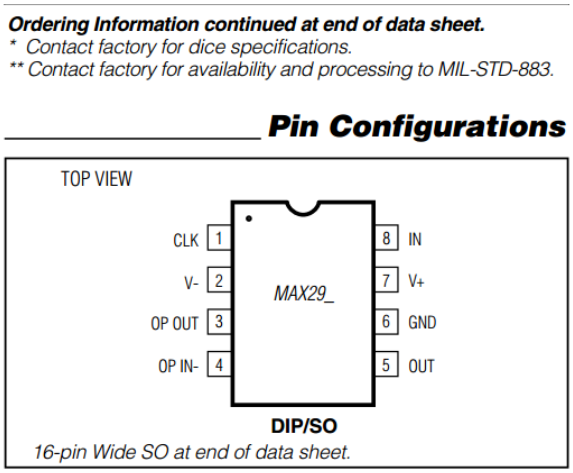

The op amp chosen to implement the gain stage was a TI OPA2227. This is a high precision, low noise operational amplifier that is unity-gain stable, features high slew rate (2.3 V/µs) and a wide bandwidth (8 MHz). It also has a high common mode rejection ratio (CMRR = 138 dB) and low nonidealities, namely low input bias current and low offset voltage. It can also be operated using a single supply. A plot of the typical open loop gain and phase characteristics is shown below.



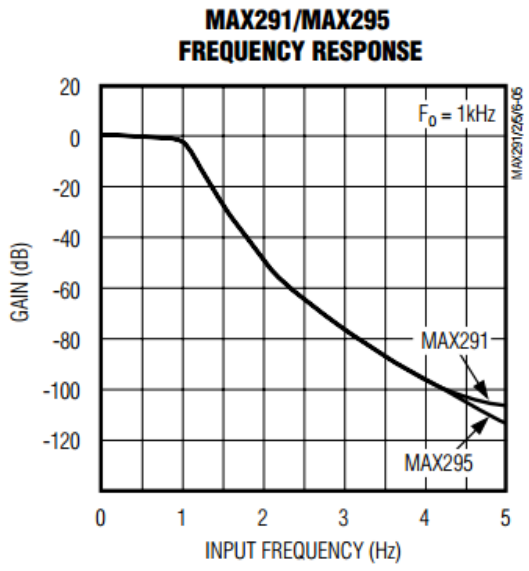
Based on the datasheet specifications we determined that this op amp was robust enough to be used in the gain stage. A DIP-8 package was chosen for ease of replaceability.

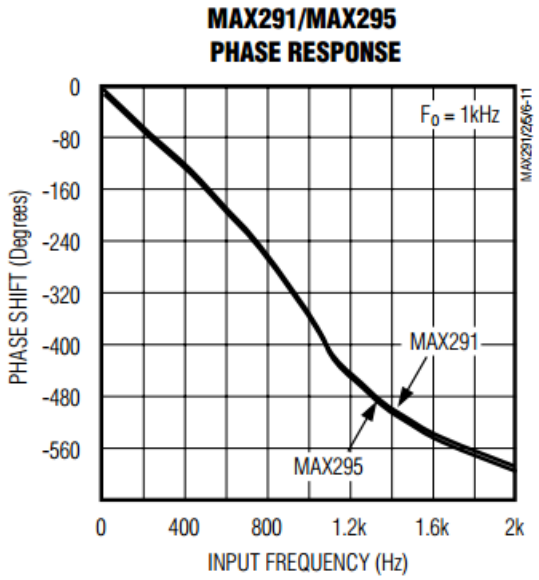
**Filter IC (Maxim Integrated MAX291CSA+-ND)**




The filter IC chosen to implement the lowpass filtering was the Maxim Integrated MAX291CSA+-ND. This is an easy-to-use, 8$^{th}$ order lowpass switched capacitor filter that can be set up with cutoff frequencies ranging from 0.1 Hz to 25 kHz. It is a Butterworth filter that provides maximally flat passband response and has a fixed response, so the only design procedure required is to select the clock frequency that in turn determines the cutoff frequency. The frequency response characteristics are shown below.



Another advantage is that it can be used in a single supply mode, which is ideal for our baseband system which runs only on a 5 V supply. The prescribed circuit for this usage is shown below.

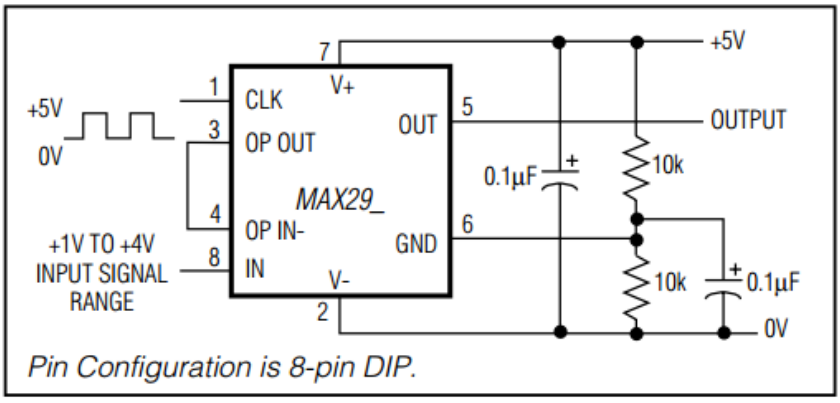



*Figure 3. +5V Single-Supply Operation*

A DIP-8 package was chosen for ease of replaceability.

**Passive components**

The values of the passive components to be used were determined by the requirements of the gain and filter stage. All of the passive components were chosen to be surface mount (SMT) components. SMT components have a large advantage over through hole components in that there is hardly any lead inductance/resistance and no capacitance due to the way the component is constructed. For SMT capacitors, the equivalent series resistance (ESR) and equivalent series inductance (ESL) was much lower and the self resonant frequency (SRF) was much higher than that of the through hole equivalent. This would ensure that the capacitors behaved properly in the IF frequency range.

When picking the passive components, components with a low tolerance were chosen as much as possible to reduce the amount of variation in the nominal value. This was done within the constraint of our budget. Furthermore for consistency a single supplier was chosen for a certain kind of passive component (e.g. one supplier for only resistors, another supplier for only capacitors, etc).

The final thing to consider is the current draw from the op amp IC and the filter IC. The typical values are tabulated below.

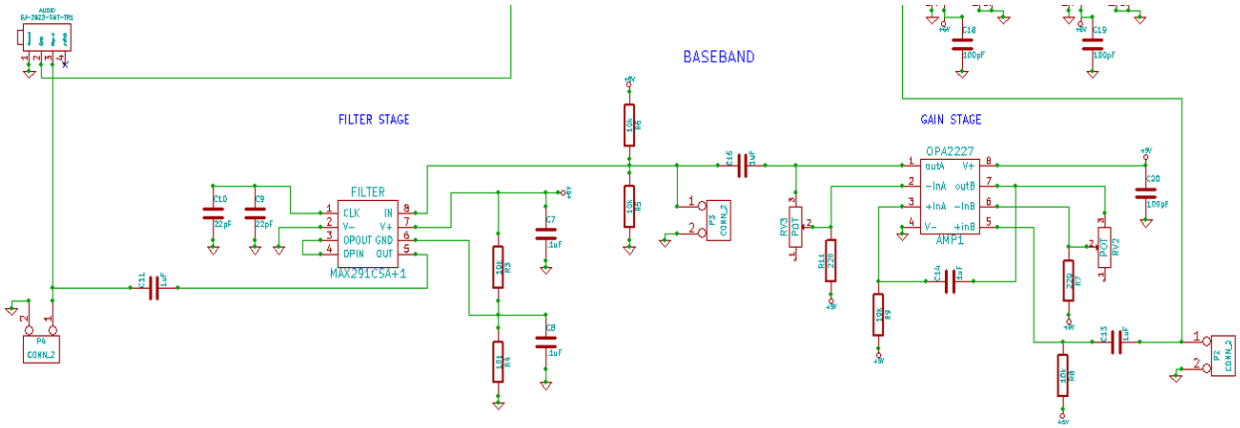
| Component | Current draw (mA) |
|---|---|
| TI OPA2227 | 7.4 |
| Maxim Integrated MAX291CSA+-ND | 15 |

## Baseband PCB Layout

When designing the baseband PCB layout the two key points to keep in mind are proper passive component and test point placement. These two points will be addressed shortly. The PCB design software used is KiCAD. KiCAD is an open source software for electronic design automation (EDA) that is recommended for hobbyists or those who are new to PCB design as the learning curve is much less steeper than a tool like Cadence Allegro.

**Schematic**

The first step in the process is to build the schematic. This is done by using the EESchema tool in KiCAD. The schematic for the baseband system is shown below.



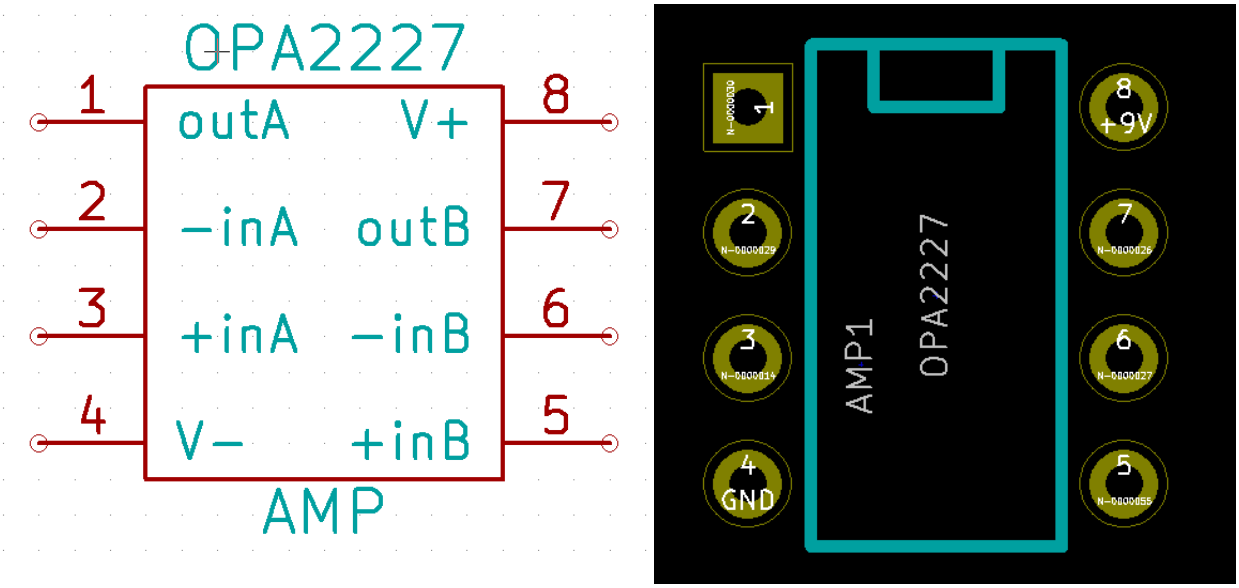

When creating the schematic it is important to keep in mind the strategic placement of test points. The test point itself can be any sort of header pin that you can hook a probe to. This is necessary because it allows for testing the performance of individual subsystems without having to touch any component leads which is an inaccurate way of measuring.

Another useful tip is to place bypass capacitors at points in your circuit where there could potentially be a sudden large amount of current draw. In such a scenario the bypass capacitor will help to fill in the “dip” in the current due to the charge stored on it. The size of the capacitor determines how big of a “dip” it can fill. The larger the capacitor, the larger the “dip” it can handle. A typical bypass capacitor value is 0.1 µF or even 1 µF.

For some components it is necessary to create your own symbols as they may not exist in the KiCAD library. This is highly recommended as it allows for a cleaner schematic layout. It is also necessary to make the component footprint at this time so that in a later step it is easier to link the two. Another handy trick is to define all power supply connections by an equivalent pin. This avoids the problem of making the schematic incoherent by having power supply connections running everywhere.

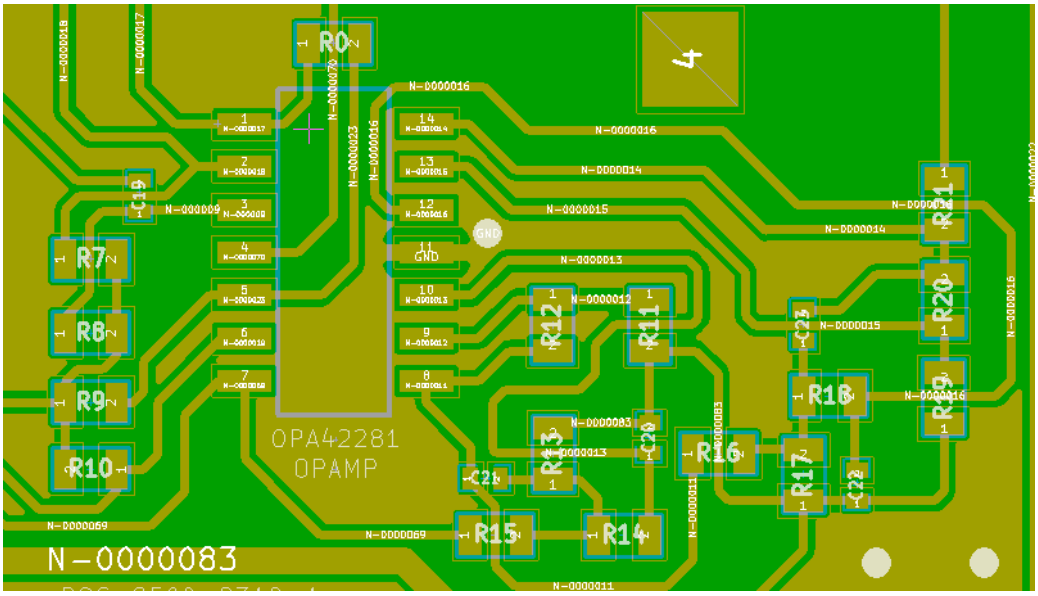
**Component to footprint association**

After the schematic is made it is necessary to associate the components shown in the schematic to their actual footprint layout. To do this, the CvPCB tool is used. The diagrams below show an example of the TI OPA2227 op amp schematic symbol and its associated footprint side by side.



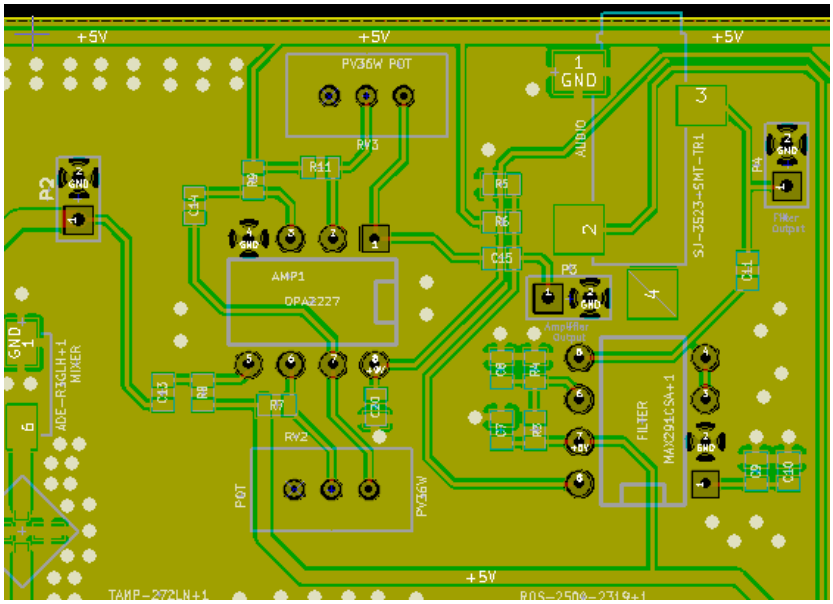

**PCB layout**

The final and most important step in the PCB design process is to perform the layout of the entire board. This step is quite intricate as all components have to be connected together with traces and issues such as too many components in a small area quickly become apparent. For example, below is a screenshot of the layout of the gain+filter stage of Iteration 1:

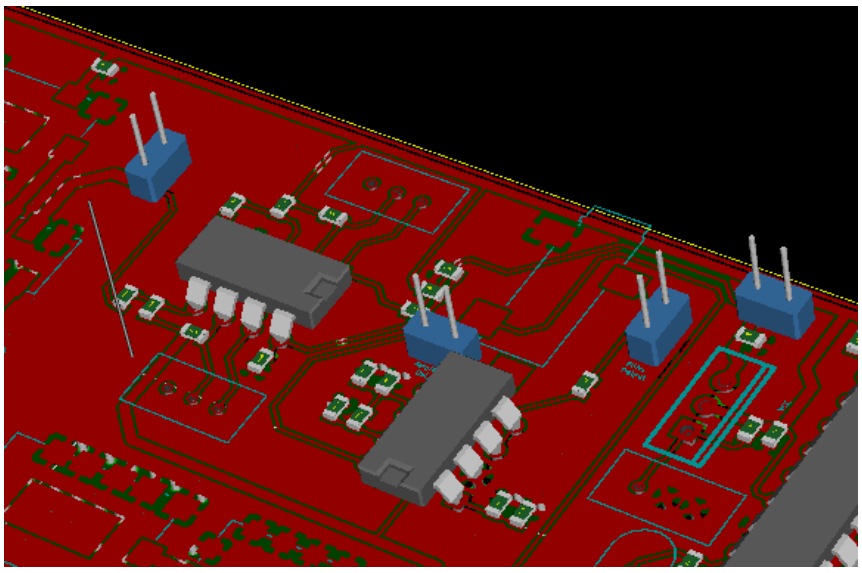


It is apparent from the picture above that there was a great deal of complication in implementing the 4th order filter due to too many passives. In this case a great deal of ingenuity is required to connect all the passives. This density of components leads to undesirable effects as discussed earlier.

The final baseband PCB layout from Iteration 2 is shown below. Note how much cleaner it is compared to that of Iteration 1. Each substage has a test point after it which facilitates easier debugging.




The 3D view of the layout above is shown in the screenshot below.



With the completion of the PCB layout, the Gerber files can then be generated and PCB can be manufactured

## Conclusion

This application note presented a comprehensive outline of the process required to design a full baseband system for a FMCW/Doppler radar. The application note began by first clarifying essential baseband signal processing concepts such as filtering and sampling. Next, the system design was thoroughly examined by determining specifications and selecting the appropriate components. Next the DSP code used to process the incoming signal was analyzed. After outlining the nuances of the system design and code the PCB layout process was presented. The baseband system is crucial in determining parameters such as range. Therefore this application note touches on all the aspects of a successful baseband system design and has hopefully reduced the complexity of designing a functional FMCW/Doppler radar.

## References


[1] *Applied Digital Signal Processing,* Dmitris Manolakis and Vinay Ingle

[2] *RF and Microwave Wireless Systems*, Kai Chang

[3] *Modern Digital and Analog Communication Systems,* B.P. Lathi, Zhi Ding

[4] *Application Note AN104 – Understanding FFT Windows*, http://www.physik.uni-wuerzburg.de/~praktiku/Anleitung/Fremde/ANO14.pdf

[5] *Understanding FFTs and Windowing,* National Instruments (http://www.ni.com/white-paper/4844/en/)