

EEC 134 --- WINTER 2016

Instructor: Xiaoguang "Leo" Liu

Application Note

Baseband Design

Duyen Tran
ID#: 999246920
Team DMK

This application note provides the process to design the baseband of the radar system.

Table of Contents:

- 1) *Baseband Components*
- 2) *Schematic*
- 3) *PCB Design*
- 4) *PCB Fabrication*
- 5) *Testing*

1) Baseband Components

1.1 Power supply/voltage regulator

We will use a voltage regulator (LM317 IC – Figure 7) to provide a relatively stable dc voltage supply.

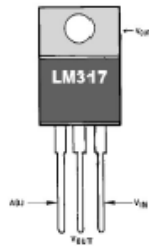


Figure 7: Pin out of the LM317 IC.

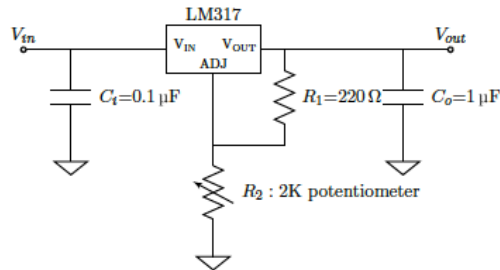


Figure 8: Schematic of the voltage regulator circuit using LM317.

We will supply an input of 8V to get an output of 5V.

1.2 Precision voltage reference

In order to present a precise and stable reference for the DAC and the ADC, we will build a precision voltage reference circuit. We will use the TI LT1009 reference IC with 2.5V output voltage.

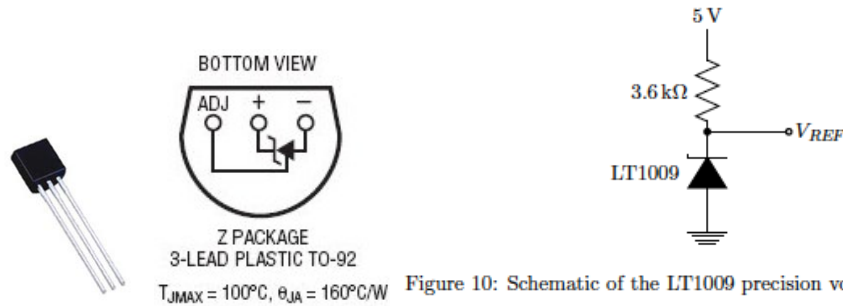


Figure 10: Schematic of the LT1009 precision voltage reference circuit.

1.3 Function generator

We will use a micro-controller (Teensy 3.1) and a DAC IC (MCP4921 DAC).

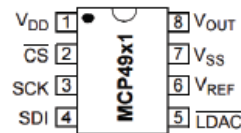
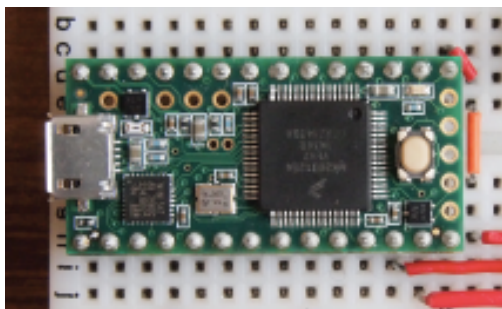


Figure 12: (a) Pinout diagram and (b)

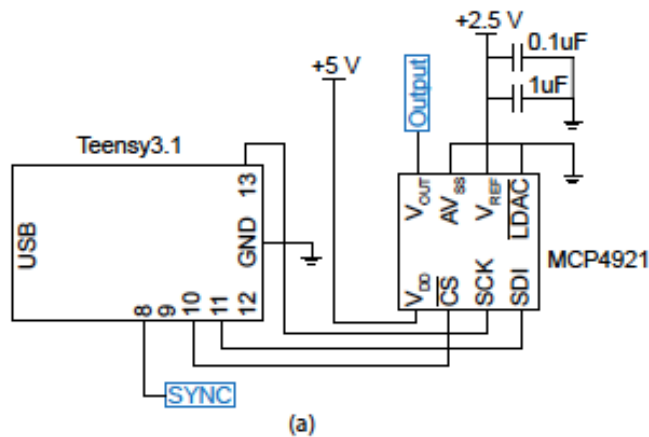


Figure 13: Schematic of the connection between the Teensy 3.1 and the MCP4921.

1.4 Gain Stage + Active low-pass filter

We will implement a gain stage and an active low pass filter (LPF) with an adjustable gain using the TI TL974IN quad Op-Amp.

For the 5V supply, we will use the output of LM317 IC; and for 2.5V supply, we will use the reference voltage of LT1009.

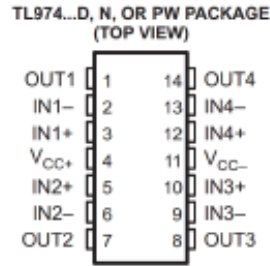


Figure 14: Pinout diagram of the TL974 Op-Amp.

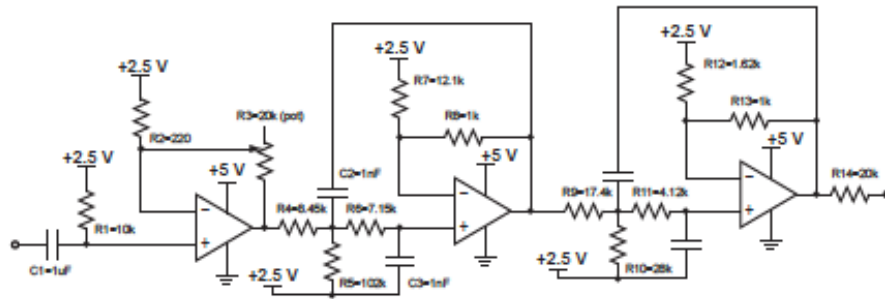


Figure 15: Schematic of the active low-pass filter.

2) Schematic

We will use KiCad to design the schematic and the PCB layout for the Baseband by following step by step on *The Tutorial on PCB Design* provided by Prof. Liu. In this tutorial, I will just show the tips on this Baseband PCB Design.

In the default KiCad library, we could find all the schematic symbols for the Baseband's components. However, there is not a proper schematic symbol of Teensy 3.1, so we have to create a schematic symbol for it using "Schematic library editor":

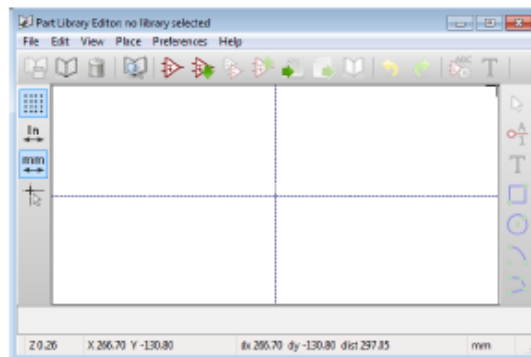


Figure 8: The schematic library editor program

- Click the “Create a new component” button.
- Click the “Add graphic rectangle to the component body” button and use the mouse cursor to draw a rectangle as the Figure 11 below.
- Click the “Add pins to the component” button and then add the component pins as follow the Teensy 3.1 schematic symbol below.

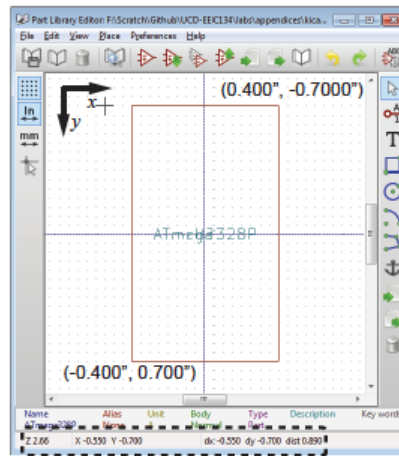
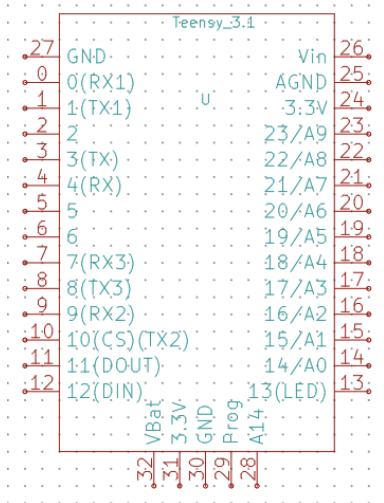


Figure 11: Drawing a rectangle in the library editor main window with the help of the built-in coordinate system.

- Then finally follow the following block diagram and the schematic for each component provided in part 1) *Baseband Components* to draw the completed schematic for the Baseband.

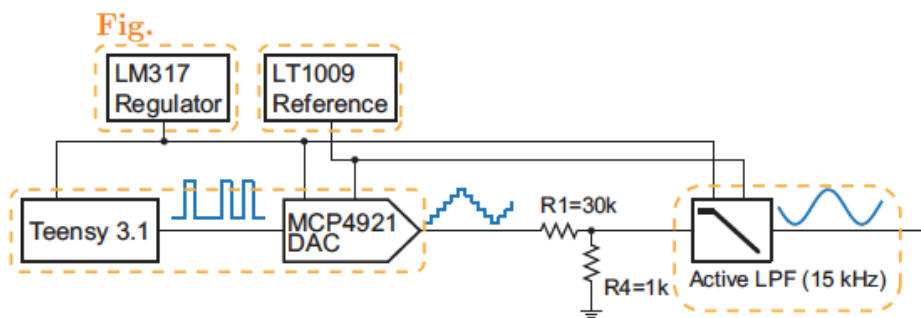


Figure 18: Block diagram of the PCB design.

- Remember to add the test points at pin #8 of the Teensy 3.1, output of MCP4921, and input of the gain stage. This will help to be easier to test the Baseband at the end.

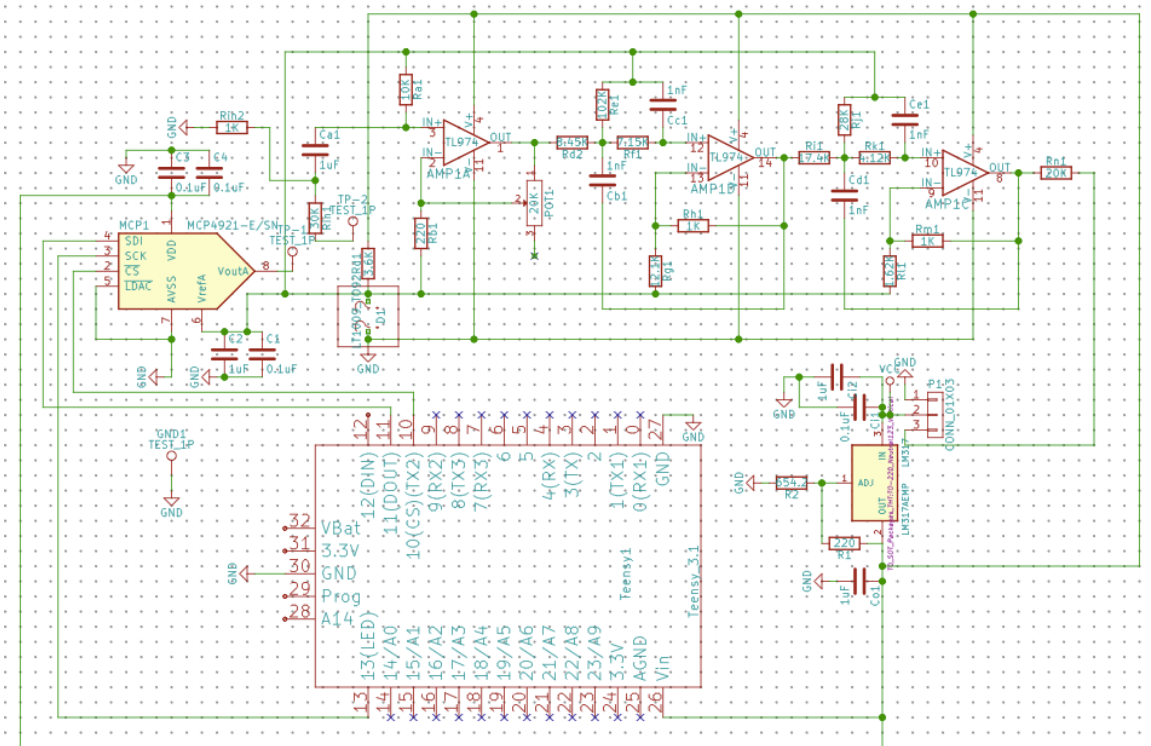
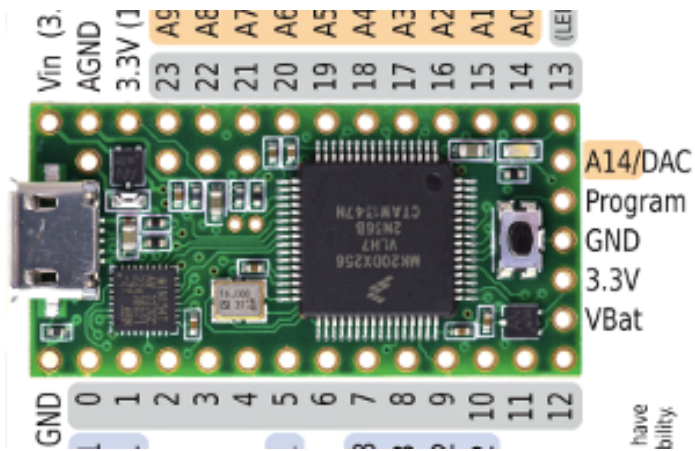


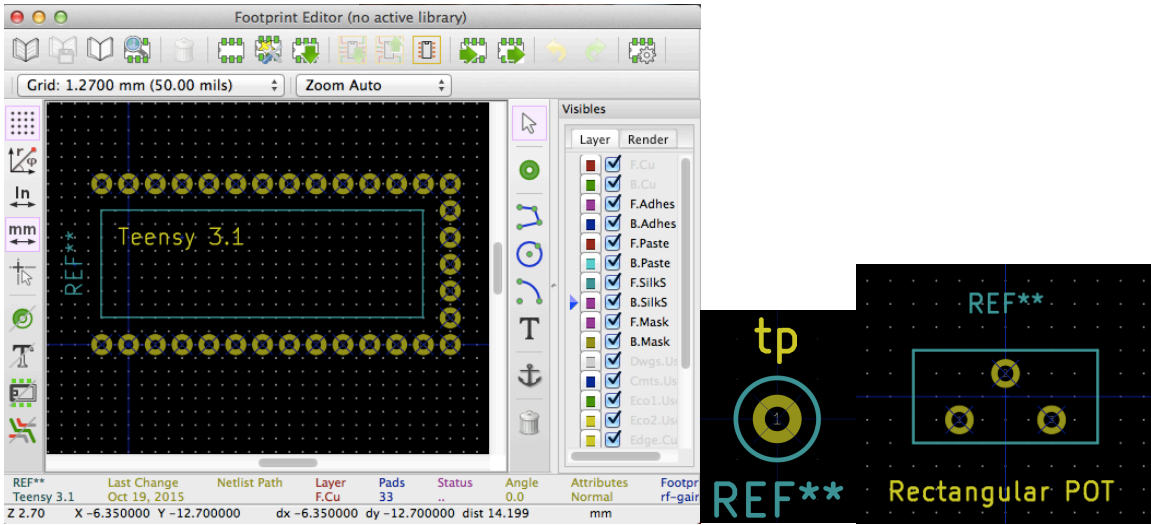
Figure A: The Final Schematic of The Baseband

3) PCB Design

In the KiCad library, Teensy 3.1 footprint is missing, so we have to create our own using “PCB footprint editor”. Follow the steps in the PCB Tutorial and make sure that all the pads of Teensy 3.1 have the correct dimensions (drill size, pitch and row spacing).



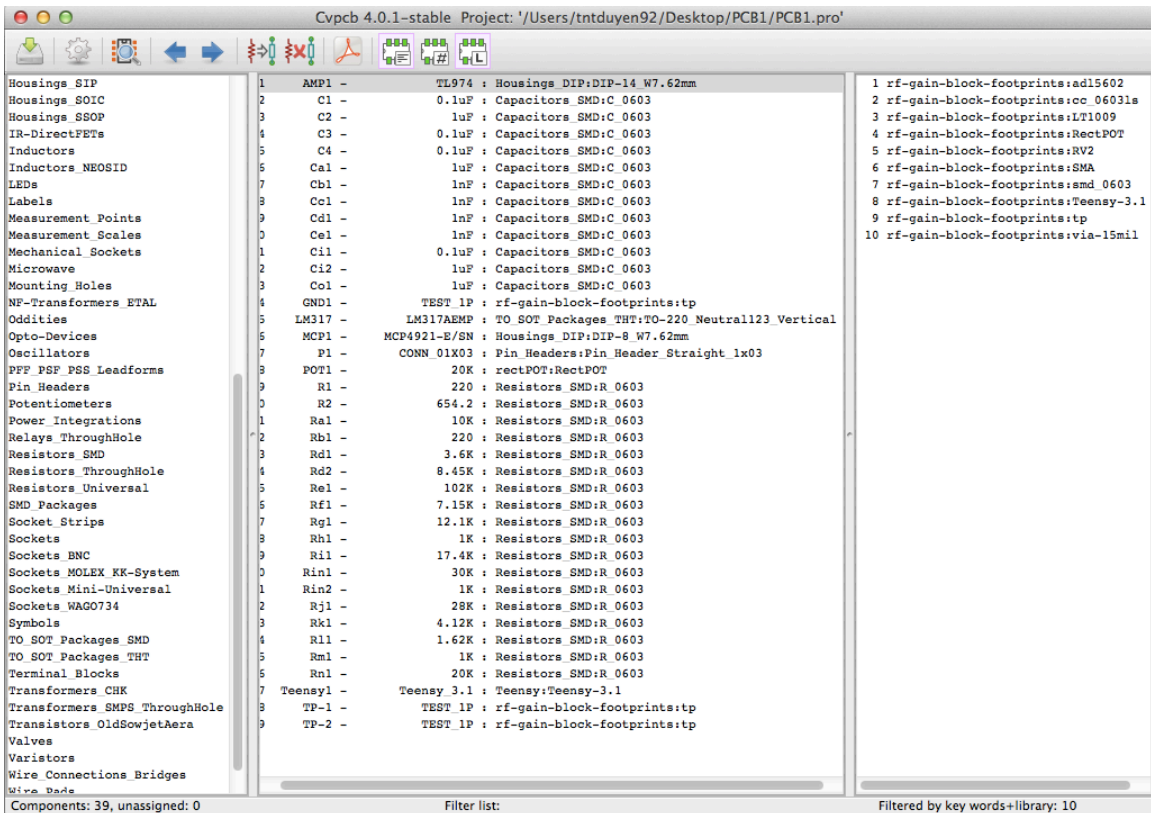
Below is the completed footprint of the Teensy 3.1 in the Footprint Editor:



And also footprints of test point and rectangular POT.

Make sure to save it in the same directory as our project after finishing.

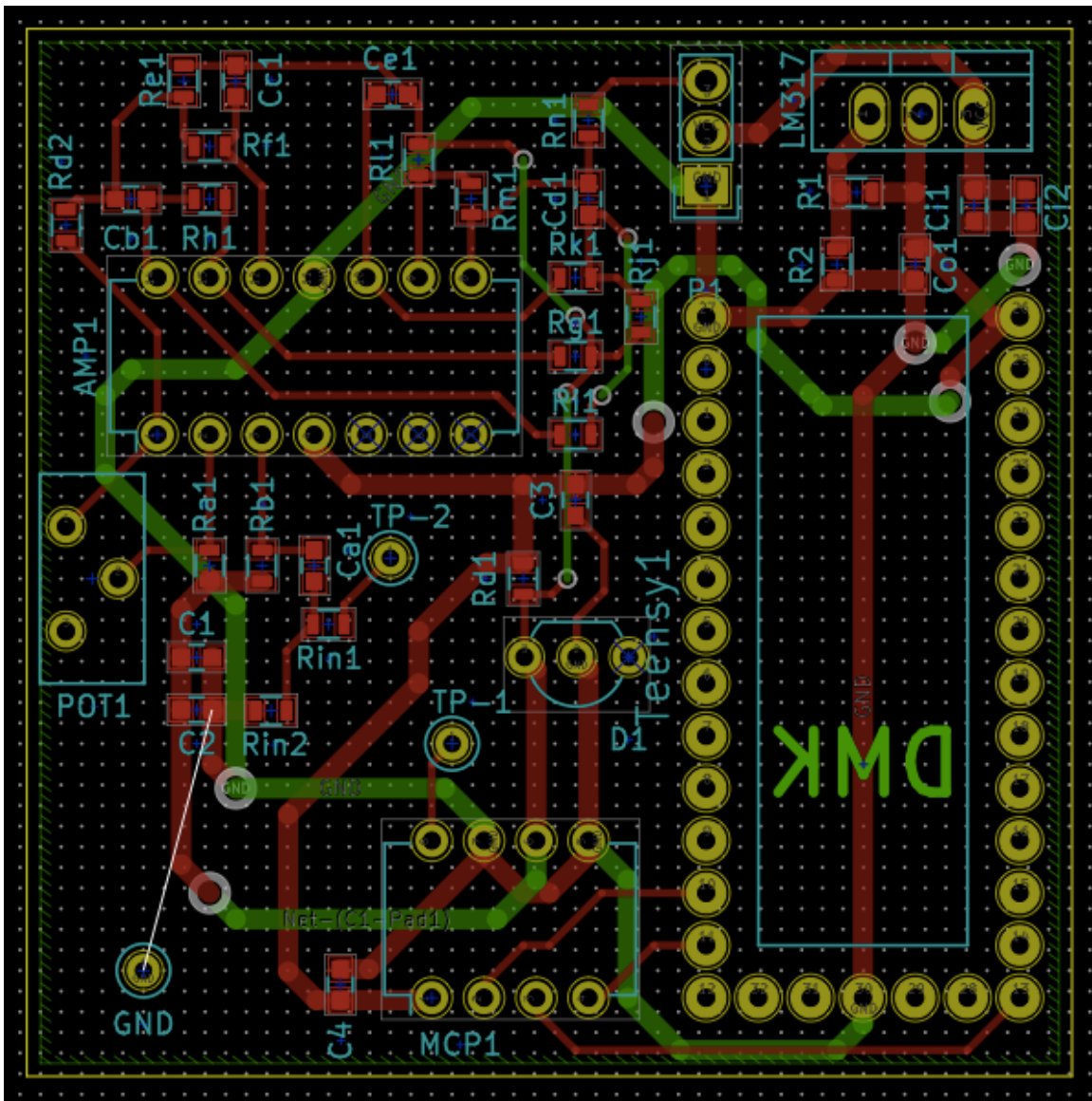
The following is the final schematic symbol and footprint association in CvPcb:



Then now is laying out the PCB:

- Click the "Generate netlist" button to generate a netlist file. Simply use the default file name and location.
- Start Pcbnew.

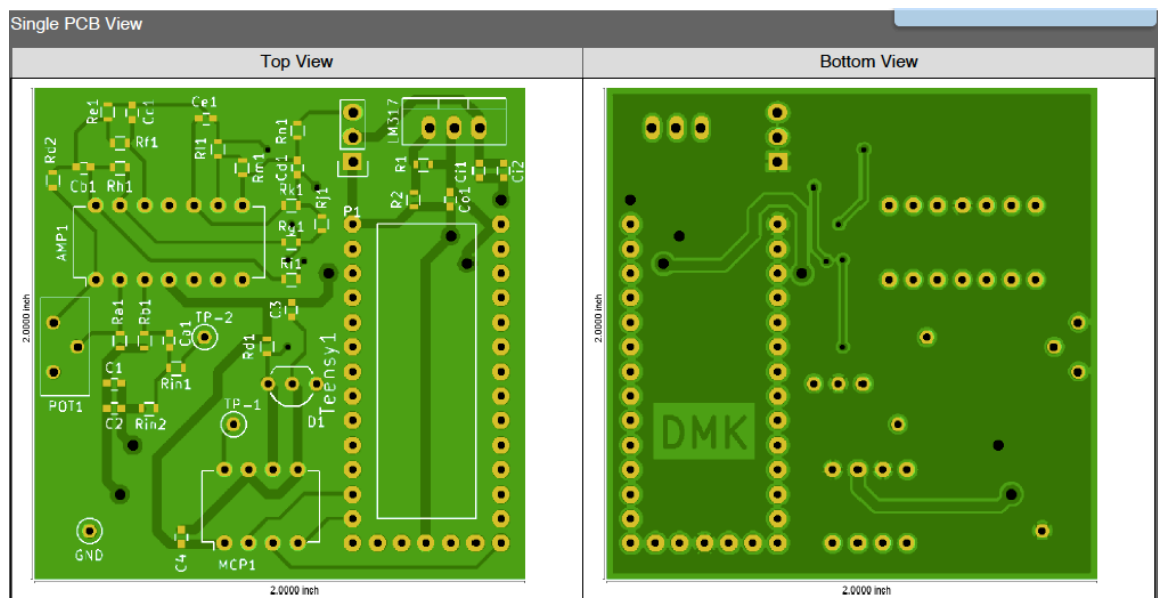
- Click the “Read Netlist” button. The “Netlist” dialog box will appear. Then Click the “Browse netlist files” button, and choose the netlist file generated from Eeschema. And then click the “Read Current Netlist” button.
- Arrangement all the components’ footprints.
- Now follow the guidance of the rat's nest lines and connect the components together using copper traces. To draw a trace, Click the \Add tracks and vias" button , click on the component pin that you want the trace to start from, move the cursor to the pin that you want the trace to end, and double click to complete the trace.
- The below figure is the final PCB layout of our group baseband without a copper fill to easier to see.



- Finally, fill the PCB with copper to increase the isolation between components and provide easy access to ground. Follow step by step in the PCB tutorial.

4) PCB Fabrication

- After we satisfied with the PCB layout, we need to generate the artwork files for the PCB manufacturer to produce your PCB. The most common artwork file format is the Gerber format.
- To generate the Gerber files in Pcbnew, go to “Files→Plot”.
- Click “Plot” to generate the Gerber files for the artwork and click “Generate Drill File” to generate the drill file (.drl) for the vias.
- Combine all these Gerber files into a .zip file.
- Then upload to Bay Area Circuits website to get the DFM Report.
- Make sure no red marks on the DFM Report then we could send all the Gerber files to Bay Area Circuit to manufacture.
- The following figure shows our group PCB Board views (Top & Bottom):



5) Testing

After the PCB boards come back, solder (SMD first and then through holes) and test the Baseband.

a. Function Generator

- Open the Arduino IDE, create a new sketch, and input the code below:

```
/*  
Triangle wave and sync pulse generator to control a (0-5V input range) VCO for FMCW radar.
```

The MPC4921 DAC is used to generate a triangle wave with a period of 40ms. PWM of the Arduino UNO is used to simultaneously generate the sync pulse, used for signal processing.

```

*/

#include <SPI.h> // Include the SPI library

word outputValue = 4; // A word is a 16-bit number
int incr = 1;

const int slaveSelectPin = 10; //set the slave select (chip select) pin number
const int SYNC = 8; //set the SYNC output pin number

void setup()
{
  // Set pins for output
  pinMode(SYNC, OUTPUT); // SYNC pin
  digitalWrite(SYNC, LOW); // Sync pulse low
  pinMode(slaveSelectPin, OUTPUT); // Slave-select (SS) pin
  SPI.begin(); // Activate the SPI bus
  SPI.beginTransaction(SPISettings(16000000, MSBFIRST, SPI_MODE0)); // Set up the SPI
  transaction; this is not very elegant as there is never a close transaction action.
}

void loop()
{
  analogWrite(A14, outputValue);
  if (outputValue == 4040 || outputValue == 0)
  {
    incr = -incr;
    digitalWrite(SYNC, !digitalRead(SYNC));
  }

  outputValue = outputValue + incr;

  byte HighByte = highByte(outputValue); // Take the upper byte
  HighByte = 0b00001111 & HighByte; // Shift in the four upper bits (12 bit total)
  HighByte = 0b01010000 | HighByte; // Keep the Gain at 1 and the Shutdown(active low)
  pin off
  byte LowByte = lowByte(outputValue); // Shift in the 8 lower bits

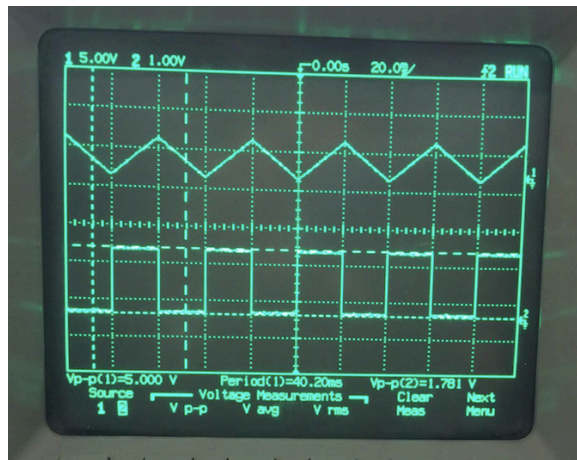
  digitalWrite(slaveSelectPin, LOW);
  SPI.transfer(HighByte); // Send the upper byte
  SPI.transfer(LowByte); // Send the lower byte
  digitalWrite(slaveSelectPin, HIGH); // Turn off the SPI transmission
}

```

→ This code will implement a triangle wave with peak to peak voltage at 5V and a period of 40ms.

- Connect the Teensy with the computer using a USB cable.
- Compile and upload the code to the Teensy.
- Record both the triangle (at VOUT) and sync (at SYNC – pin#8 of Teensy 3.1) output signals on the oscilloscope. Record their amplitudes and periods.

- If we get a triangle wave of 5V peak to peak and a period of 40ms and a square wave at SYNC as the following figure, our function generator worked fine.

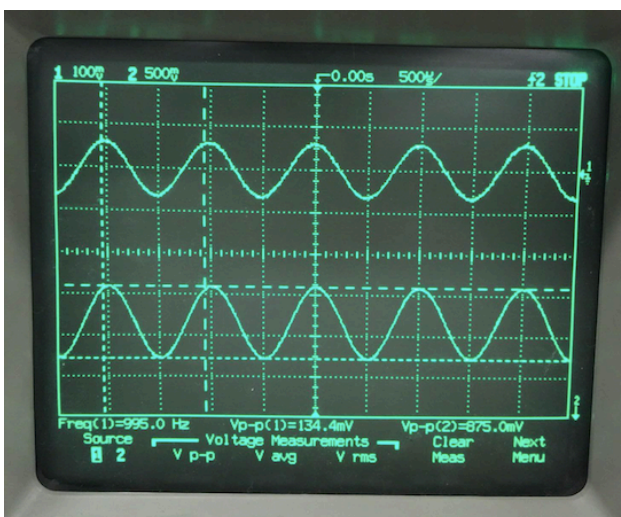


- If not, we could change the value of “int incr = 1” to get the right 40ms period and change the bits of ‘0’ or ‘1’ in the 2 line below to get a peak-peak voltage higher than 3V:

```
HighByte = 0b00001111 & HighByte; // Shift in the four upper bits (12 bit total)
HighByte = 0b01010000 | HighByte; // Keep the Gain at 1 and the Shutdown(active low)
```

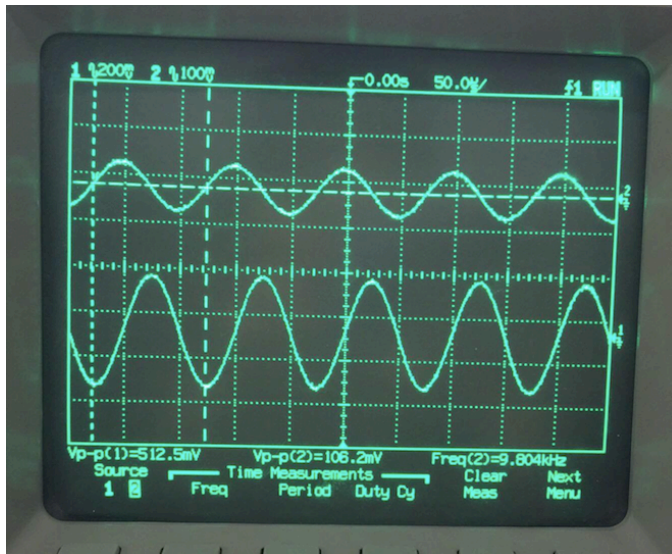
b. Gain Stage + Active LPF

- Use a function generator in the lab room to input to the Gain Stage + Active LPF a sine wave with $V_{p-p} = 100\text{mV}$ and Frequency = 1kHz.
- Record both the input and output on the oscilloscope.



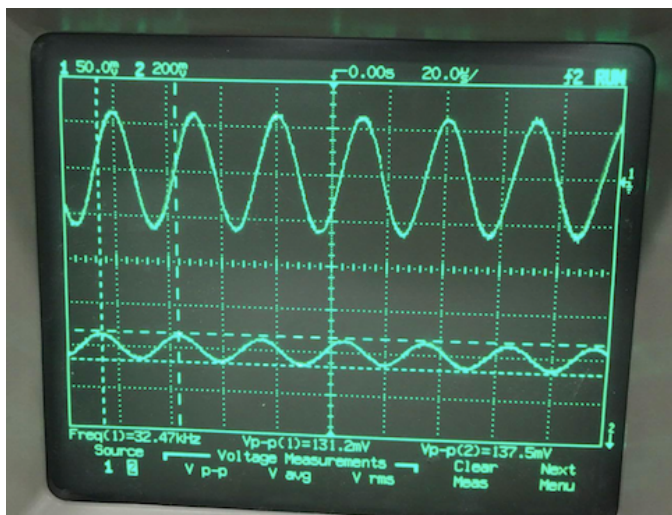
$f = 1 \text{ kHz} \rightarrow \text{Gain} = 7$

- Increase the frequency to 10 kHz:



$f = 10 \text{ kHz} \rightarrow \text{Gain} = 5$

- Finally, increase the frequency to test at what frequency the gain would be equal 1
- The following figure is recorded that our group Baseband would have a gain of 1 at frequency of 32.47 kHz.



→ If after testing, we could get the signals and gains like above figure, we could conclude that the Baseband PCB is functional.