Lab 1: Elements of Electronic Systems

Instructor: Xiaoguang "Leo" Liu
lxgliu@ucdavis.edu

Last updated: September 10, 2015

The main objective of this lab is to understand the basic components of a typical electronic system. To this end, we will build a simple electronic system shown in Fig. 1. In this system, we will use a micro-controller and a digital-to-analog converter (DAC) to generate a triangle wave. This triangle wave signal will be low pass filtered and digitized by an analog-to-digital converter (ADC) whose output will be sent to a computer for analysis and display.

Although this system may seem rather useless, it does serve the purpose of introducing the basic concepts and skills in building an embedded system that needs to interface with the analog world. In addition, some of the components in this system will become useful in later labs.

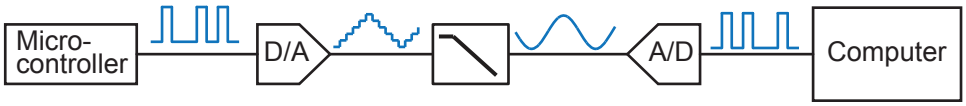


Figure 1: Lab 1 electronic system.

1 Objectives

1. Understand the functionality and characteristics of linear and switching voltage regulators;
2. Learn how to use simple micro-controllers, such as the Arduino Uno and the Teensy 3.1;
3. Learn the functionality and use of DAC and ADC;
4. Learn how to use the serial programming interface to control electronic components;
5. Learn the basic skills of designing and laying out a printed circuit board (PCB);
6. Learn the basic skills PCB assembly involving surface mount (SMD) components.

2 Prelab

2.1 Micro-controller

Micro-controllers are computing systems integrated within a single package. Compared with general purpose computers, micro-controllers find widespread use in applications where low-power consumption, small physical size, low cost, and predictable response time are required.

Traditionally, programming a micro-controller was an art that only well-trained electrical engineers can perform. This all changed with the invention of the Arduino micro-controller platform in 2005 by a group of Italian educators who were frustrated with how difficult it was to learn and use microcontrollers. They decided to develop a hardware and software platform so easy to use that even artists were able to use them. In addition, the Arduino platform is completely open source. Because of this, there has been a large number of library and add-on boards (called "Shields") developed and made available by the community, making it possible to build complex systems without having to reinvent the wheels. The huge popularity of the Arduino platform has motivated the development of numerous similar platfomrs with a wide spectrum of capabilities. These are the reasons that we choose to use Arduino compatible micro-controllers in this class[1]. We hope that even students who are not interested in embedded systems will be able to master how to use them in a relatively short of time.

Although the focus of the EEC 134 class is on high frequency systems, you will find that micro-controllers indispensable whenever you need to control a component electronically.

- To get a quick introduction to the Arduino platform, watch Episodes 1–10 of Jeremy Blums "Tutorial Series for Arduino ". It is advised that you, or your group collectively, go through all of the videos in this list. `https://www.youtube.com/playlist?list=PLA567CE235D39FA84`

2.2 Voltage regulator

In the late 1880s, a heated battle over the best mechanism to transport electricity over long distance broke out between proponents of direct current (primarily Thomas Edison) and alternating current (primarily George Westinghouse and several European companies). History eventually settled on ac current as the preferred method for long distance distribution because of its ability to be easily transformed into high voltages to reduce resistive loss along the wires. So today we all have ac outlets at home and in the lab. However, most if not all the circuits we have studied in our curriculum are powered from dc supplies. Have you ever wondered how dc voltages are generated from an ac supply? The following videos may be instructive. Make sure you watch them carefully.

[1] If you have experience using a microcontroller different than the Arduinos, you should feel free to use that platform to implement this lab; obviously you may want to make sure that you have a consensus within your group.

- How to build an AC-DC power supply:
  `https://www.youtube.com/watch?v=cyhzpFqXwdA`
- Linear voltage regulator:
  `https://www.youtube.com/watch?v=GSzVs7_aW-Y`
- Adjustable linear voltage regulator:
  `https://www.youtube.com/watch?v=IjJWWGPjc-w`
- Switch-mode voltage regulator:
  `https://www.youtube.com/watch?v=CEhBN5_fO5o`

2.3 DAC and ADC

DAC and ADC are the interface between the analog and the digital world. A DAC takes digital input ("1"s and "0"s) and convert them into an analog signal. An ADC does the just the inverse of that, converting an analog signal into a digital one. A simple DAC can be built with only resistors, such as the R-2R ladder shown in Fig. 2-a. A simple ADC is not much more complex using a linear resistor ladder integrated with comparator and decoders (Fig. 2-b)[2]

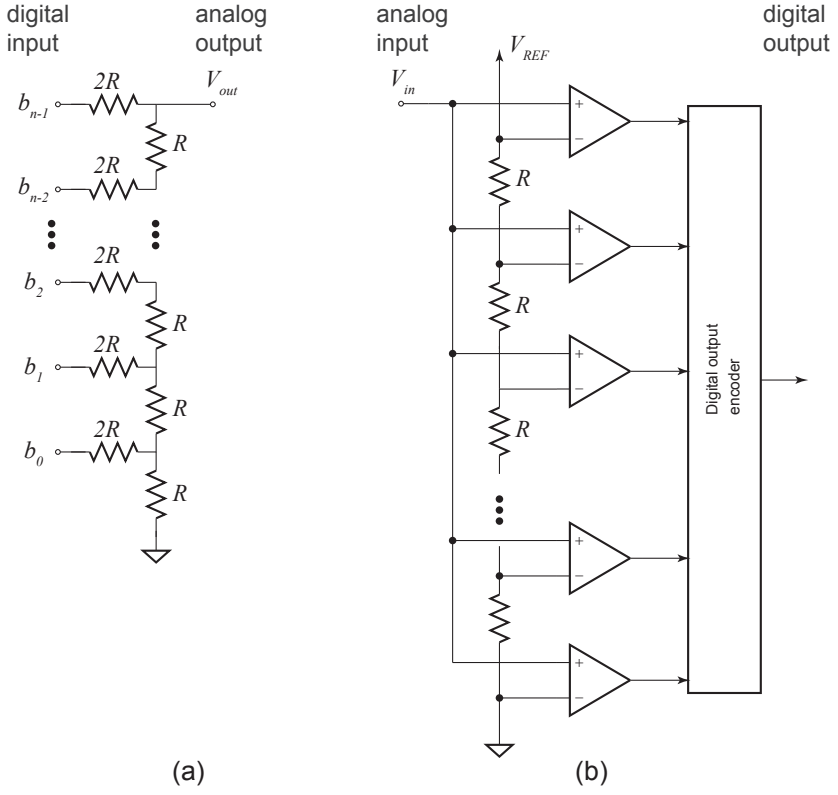


Figure 2: (a) R-2R ladder DAC. (b) ADC using a linear string of resistors.

To learn more about DACs and ADCs, watch the following video and read the following documents.

[2]A variety of other architectures exist for both DACs and ADCs with respective advantages and disadvantages.

- humanHardDrive, "Electronics 201 – Analog/Digital Conversion," `https://www.youtube.com/watch?v=cjmcAE1L6OQ`
- Bill McCulley, "Bridging the divide: Part I - DAC Introduction[3]," National Semicondcutor, 2010
- Chris Pearson, "High Speed, Digital to Analog Converters Basics," Texas Instruments, 2012.
- Mike Kester, "DAC Interface Fundamentals," Analog Devices, 2008.
- Chris Pearson, "High-Speed, Analog-to-Digital Converters Basics," Texas Instruments, 2011.

2.4 Serial interface

For micro-controllers that don't have built-in DACs and/or ADCs, an external DAC/ADC IC needs to be used when conversion between analog and digital is required. Even micr-controllers with on-chip DAC/ADC may run into situations where more than conversions channels are needed. When using an external device, a digital interface for transmitting and receiving data is needed between the micro-controller and the device.

Digital interface generally comes in two flavors: *parallel* and *serial*. In a parallel interface, each physical pin corresponds to one digital bit. For example, an 8-bit word would need 8 physical pins for transmission/reception. In a serial interface, the digital bits are transmitted/received sequentially in time and theoretically only two pins are needed (one for signal and one for ground). Depending on whether a clock is needed to time the transmission/reception of digital bits, serial interface can be further categorized into *synchronous* and *asynchronous*. For synchronous transmission, an additional pin for the clock signal would be needed.

Obviously, at the same physical bit rate[4], parallel interfaces is much faster than a serial one. But the serial interface requires far less pins and therefore a smaller chip footprint and lower PCB routing complexity, both of which translates to lower cost!. Therefore, for medium to high rate (kp/s to a few Gp/s) data transmission, serial links are more popular.

In this class, we use synchronous serial interface, in particular the Serial Peripheral Interface (SPI) protocol, as the interface between micro-controllers and external devices, such as DACs, ADCs, and some other electronically controllable RF components.

To learn more about the SPI interface, read the following documents:

- Mike Grusin, "Serial Peripheral Interface (SPI)," Sparkfun. `https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi`

[3]Following the link will lead you to EEC134's repository on Github. If you want the actual PDF file, you can clone the entire repository to your harddrive. The file is under the foler /labs/lab1/references/.

[4]how often the voltage changes from logic 1 to 0 (or the other way) on a pin.

- F. Leens, "Introduction to IC and SPI protocols," Byte Paradigm, 2009. `http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/`

2.5 Printed circuit board design

A major goal of this lab is to help you get familiar with printed circuit board (PCB) design techniques. There are numerous PCB design software packages available on the market. In this quarter, you will start with a simple tool called KiCad. Once you are familiar with the basic concepts, it is easier to transition to a more sophisticated tool, such as the Cadence Allegro which is made available to us by a donation from Cadence.

KiCad is an open-source electronic design software with very good PCB layout capabilities. KiCAD is becoming more popular in the hobbyist electronics community because it doesn't have any restrictions on the number of layers or boards sizes, nor does it lock you down to a particular PCB manufacturer.

To learn how to use KiCad and design PCBs in general, please go through the following materials.

- Video tutorials from Contextual Electronics: `https://www.youtube.com/playlist?list=PLy2022BX6Esr6yxwDzhqYZyuuenJE2s5B`
- Video tutorials form Yoonseo Kang: `http://vimeo.com/user9565582`
- A simple KiCad example:

  `http://exploreembedded.com/wiki/A_simple_example_for_beginners:_LED_Breakout`

Pre-lab Assignment 1.1
Due: Sep. 25th, 2015

Please answer the following questions:

a) What are the advantages and disadvantages of using switch mode voltage regulator vs a linear voltage regulator?

b) For the circuit of Fig. 3, with sing an input voltage of 9V and R1=510 Ω, what's the value of R2 such that the output voltage is 5 V? What is the efficiency of the regulator in this case? "LM317" refers to the Texas Instruments LM317 voltage regulator IC. You should be able to find its datasheet online.

c) According to the datasheet you found above, what is the typical drop out voltage for the TI LM317? If an input voltage of 12 V is used, what range of output voltage can be considered regulated?

d) What is the maximum efficiency of the TI LM2694 switch mode voltage regulator for an output voltage of 5 V? Under what conditions is this efficiency achieved?

e) What do *LSB* and *MSB* mean in a DAC? For a 12-bit DAC with an output reference voltage V_{ref} of 2 V, how much voltage does an LSB correspond to? What about a 24-bit DAC instead?

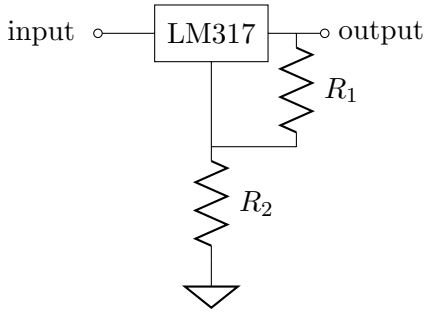


Figure 3: LM317 linear regulator circuit.

Pre-lab Assignment 1.2
Due: Oct. 2th, 2015

Please answer the following questions:

a) What does *SFDR* mean for a DAC? What is the typical SFDR of the Analog Devices (ADI) AD9788 DAC?

b) What is the *image* signal for a DAC output? If a DAC is operating at a clock rate of 200 Msps and the output fundamental signal is a 50 MHz sine wave, what are the frequencies of the first three images above the fundamental?

c) What does *SNR* mean for a DAC? What is the typical SNR for the Linear Technology (Linear) LTC2641 DAC?

d) What does *THD* mean for an ADC? What is the typical THD for the TI ADS5400 ADC ?

e) What does *SINAD* mean for an ADC? What is the typical SINAD for the Linear LTM9008-14 ADC?

f) What does *ENOB* mean for an ADC? How is it calculated? What is the ENOB for the Maxim Integrated (Maxim) MAX11270 ADC?

Pre-lab Assignment 1.3
Due: Oct. 9th, 2015

Please answer the following questions:

a) What is the highest speed 8-bit ADC you can find? What is its power consumption? What would be the power consumption of an 8-bit ADC with half of this speed?

b) What is the frequency domain representation of the following triangle wave signal?

$$x(t) = \sum_{k=-\infty}^{\infty} \left\{ \left(\frac{2t}{T} - \frac{kT}{2} \right) \Pi \left[\frac{2(t-k)}{T} \right] + \left(1 - \frac{2t}{T} \right) \Pi \left[\frac{2(t-k-\frac{1}{2})}{T} \right] \right\},$$

where $\Pi(t)$ is the rectangular pulse function

$$\Pi(t) = \begin{cases} 1 & 0 \le t \le 1; \\ 0 & \text{elsewhere.} \end{cases}$$

If we pass the signal through an ideal low-pass filter that keeps only the first three harmonics (including the fundamental, the 2nd harmonic,

and the 3rd harmonic), what would the filter output signal look like in the time domain?

c) Design an analog low-pass filter that meets the following specifications:

   (a) In-band gain: 10 dB;

   (b) 3-dB cut-off frequency: 20 kHz;

   (c) Attenuation at 100 kHz: 30 dB.

   You may find online filter design tools, such as the TI WEBENCH Filter Designer[5] and the ADI Filter Wizard, to be useful. Verify the performance of your filter by simulation. You may use a SPICE simulator, such as LTSpice, or a high frequency circuit simulator such as Agilent/Keysight Analog Design Systems (ADS). Both software are available on lab computers.

[5] A short introduction video is available at `https://www.youtube.com/watch?v=bdtLbtfTV8A`

3 Equipment & Supplies

- breadboard
- jumper wires
- Arduino UNO development board
- Teensy 3.1 development board (optional)
- MCP4921 digital-to-analog converter (DAC)
- misc resistors
- misc capacitors
- 8x AA rechargeable batteries
- battery pack

4 Procedures

4.1 Power supply/voltage regulator

We will use a USB battery pack to power up our system. Batteries provide the cleanest (i.e. very little noise) type of supply but with one major disadvantage, their voltage keeps dropping due to increased internal resistance under discharge. A voltage regulator is therefore often used to provide a relatively stable dc voltage supply.

In our first design, we will build an adjustable voltage regulator that can be used to provide the supply voltage for several parts in our system. The regulator is based on the LM317, a linear voltage regulator IC (Fig. 4).

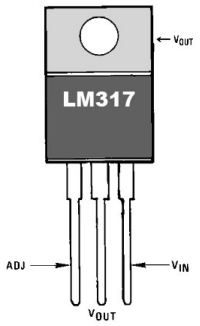


Figure 4: Pin out of the LM317 IC.

A linear regulator provides very stable and clean voltage output but it can only provide regulated output voltage lower than the primary voltage source. Another disadvantage of a linear regulator is its low efficiency when the difference between the input and output voltages is large. Because the same amount of current flows through the regulator and the load, the power being dissipated is roughly

$$P_d = I_{load} \times (V_{in} - V_{out})\,. \tag{1}$$

When the load current is high, this power dissipation can be significant. It is therefore important to provide good heat sinking to the regulator IC. In fact, many regulator ICs, such as the LM317, have large exposed metal pad with low thermal resistance to facilitate mounting a heat sink. However, it is very important to note that the metal pad on some regulator ICs, such as the LM317, is connected electrically to the output of the IC. If the heat sink may potentially come in contact with any other part of the circuit (including the enclosure, which often is tied to ground), proper isolation is needed between the IC and the heat sink.

1. Using Fig. 5 as a reference, build the regulator circuit on the breadboard. You should be able to find the LM317 pinout diagram from its datasheet, which you find from online. *As a general suggestion, it is important to keep your circuit well laid out and keep the connecting wires as short as*

possible. Cut the wires to the right length if necessary. Fig. 6 shows an example breadboard layout for the LM317 regulator circuit in this lab.

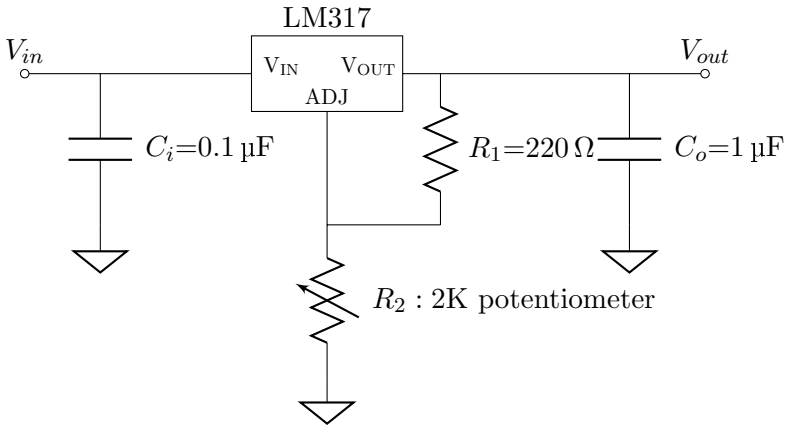


Figure 5: Schematic of the voltage regulator circuit using LM317.

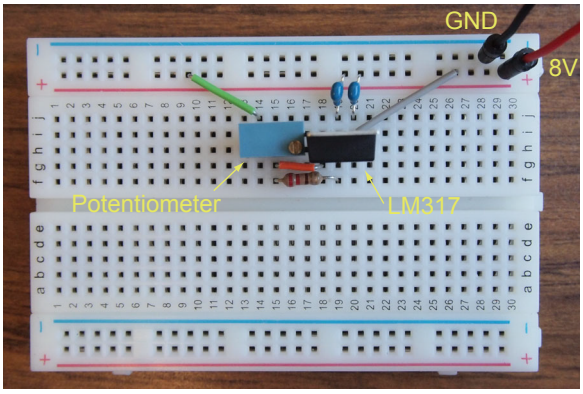


Figure 6: An example breadboard layout of the circuit in Fig. 5.

2. For testing the circuit, we will first use the lab bench power supply as our input power. Set the power supply voltage to 8 V and connect it to the input of your voltage regulator circuit. Adjust the R_2 potentiometer until the output voltage is 5 V. You can use the lab multimeter to measure the output voltage.

3. Line regulation:

   (a) Adjust the input voltage from 8 V to 5 V at 0.25 V intervals, and then from 5 V to 2 V at 0.5 V intervals, record the regulator output voltage using the multimeter;

   (b) Plot your results. In what input voltage range does the LM317 provide good line regulation?

   (c) What is the dropout voltage at various input voltages? Does it agree with the datasheet?

4.2 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. In this circuit, we will use the TI LT1009 reference IC with 2.5 V output voltage and less than ± 0.2% initial accuracy (no calibration and adjustment). The LT1009 IC can be used as a two-terminal voltage reference although it does provide an optional 3rd terminal for ±5% adjustment.

1. Build circuit shown in Fig. 7 on your breadboard.

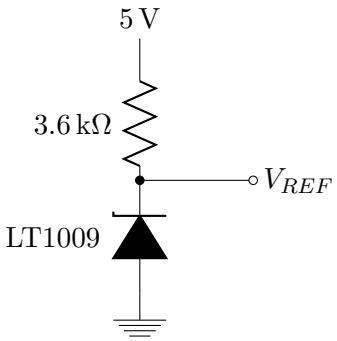


Figure 7: Schematic of the LT1009 precision voltage reference circuit.

2. Verify that the output voltage of the circuit is 2.5 V.

4.3 Function generator

There are numerous ways to build a function generator. You can use a 555 timer, a ring of inverters, dedicated oscillator ICs, or a direct digital synthesis (DDS) circuit which uses high-speed digital circuits to implement fast waveform generators. It does so by outputting values from a look-up table (LUT), which stores the desired waveform in discretized values, and converting them to an analog signal by a DAC.

In this lab, we will try to emulate a DDS using a micro-controller and a DAC IC. We are interested in generating a triangle wave which will be useful in later labs. Since DAC outputs are discrete in nature, we are really just approximating the triangle wave with a stair-case waveform (Fig. 8). Obviously, the higher the resolution of the DAC, the better this approximate is.

For this class, we will use the *Teensy 3.1* micro-controller. The Teensy 3.1 is developed by Paul J. Stoffregen of PJRC.com. The programming of Teensy 3.1 is done in the same IDE as the Arduino platform[6] and most Arduino programs will run without any problem on the Teensy. Compared to most Arduinos, however, the Teensy 3.1 has a much faster micro-controller (72 MHz, overclockable to 144 MHz) with floating point support and is therefore a much better platform than Arduinos when some number crunching is needed.

[6] with an add-on called Teensyduino, which has already been installed on the lab computer

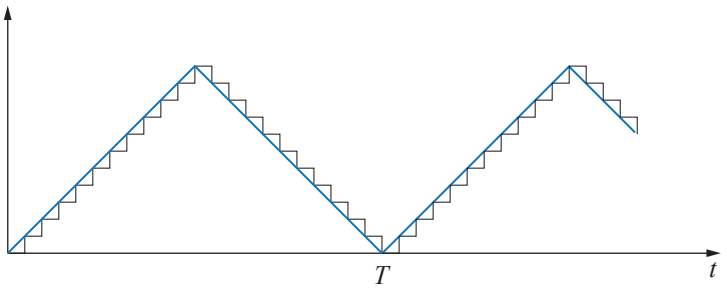


Figure 8: Triangle wave and its approximation by a stair-case waveform which more closely resembles the output of a DAC.

The DAC IC we are going to use is the Microchip MCP4921 12-bit DAC. Fig. 9a shows the pin out of the IC. The MCP uses SPI interface for digital input. The output voltage is related to the digital code input by

$$V_{OUT} = \frac{V_{REF} \times D_n}{2^{12}} \times G,$$

where V_{REF} is the reference voltage, D_n is the digital code, and G is a gain setting. G can be set by one of the register bits and can take value of either 1 or 2.

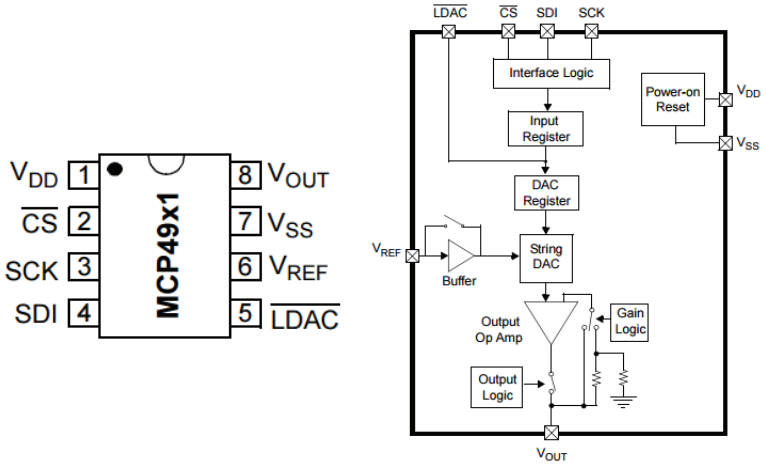


Figure 9: (a) Pinout diagram and (b) Block diagram of the MCP4921 DAC.

1. Using Fig. 10 as a reference, connect the Teensy to the MCP4921 DAC. Connect the MCP4921's VDD pin to the output of the LM317 regulator (5 V). Connect the MCP4921's VREF pin to the LT1009 reference's output (2.5 V). Connect both AVSS and $\overline{\text{LDAC}}$ of the MCP4921 to ground. A 0.1 µF and a 1 µF capacitor could be used at VREF to minimize noise at the reference. It is very important to keep your circuit (ICs, resistors, capacitors, wires, etc) organized. Use as short of a wire as possible between two connection points. Refer to Fig. **??** for an example layout.

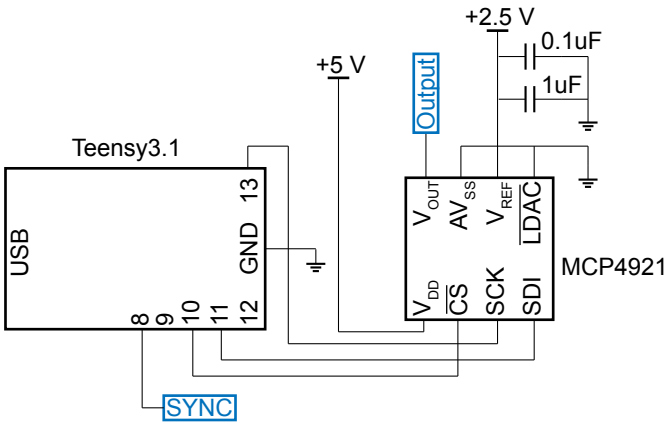


Figure 10: Schematic of the connection between Arduino UNO and the MCP4921.

2. Once the circuit is built, open the Arduino IDE, create a new sketch, and input the code "triangle_teensy_mcp4921.ino" . **Make sure you read through and understand the code.** You may also want to read the datesheet of MCP4921 to understand its SPI control protocol.

3. Connect the Teensy with the computer using a USB cable, which is used to power up and program the Teensy.

4. Compile and upload the code to the Teensy. Use the following settings under the "Tools" menu:

   (a) Board: "Teensy 3.1"

   (b) USB Type: "Serial"

   (c) CPU Speed: "144 MHz Optimized (Overclocked)"

   (d) Port: Select the COM port that the Teensy is connected to.

5. In your report, include a screen capture of both the triangle (at VOUT) and sync (at SYNC) output signals on the oscilloscope. Record their amplitudes and periods.

6. Modify your code to set the triangle wave amplitude to 2.5 V. Hint: change the gain setting of the MCP4921.

7. How can you modify the code to change the amplitude and period of the output waveform? What is the fastest triangle wave you could generate? What do you think is the limitation to going even faster?

8. Modify the code to generate a sinusoidal wave. What is the highest frequency sinusoidal wave you can generate? The following link may give you some hint. `http://interface.khm.de/index.php/lab/experiments/arduino-dds-sinewave-generator/`

9. The Teensy actually has a built-in DAC. Try to implement the triangle wave generator using the built-in DAC. Refer to `https://www.pjrc.com/teensy/teensy31.html` for some hint. What is the fastest triangle wave you can generate?

10. Use code "triangle_teensy_audio.ino" to generate a 4 kHz sine wave using Teensy's built in DAC. The code makes use of the Teensy Audio library. You should dig into the source code, in particular the "synth_waveform.cpp" file, of the audio library to understand how it works.

4.4 Active low-pass filter

In this part of the lab, we will implement an active low pass filter (LPF) with an adjustable gain stage. In the final radar system, the LPF+Gain stage will be responsible for filtering out spurious signals that may disturb the received baseband radar signals. We will use the Texas Instrument OPA4228 quad Op-Amp for both the gain stage and the active LPF. A pinout diagram of OPA4228 is shown in Fig. 6.10.4. Fig. 11 shows the schematic of the circuit. Fig. **??** shows an example of the actual circuit on a breadboard.

The filter has a cut-off frequency of 15 kHz.

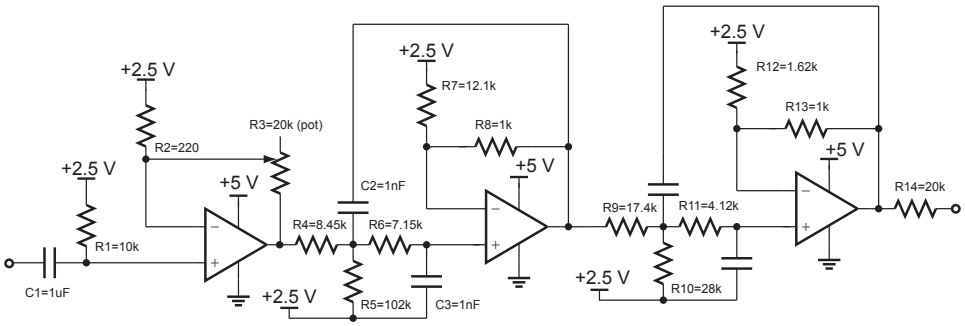


Figure 11: Schematic of the active low-pass filter.

4.4.1 Gain stage

1. Identify the gain stage in Fig. 11. What is the expression of its gain? Notice the +2.5 V bias does not affect this result, rather it simply shifts the bias of the overall amplifier since a single 5 V supply (battery) is used vs a dual-rail (both positive and negative) supply.

2. Build the gain stage on your breadboard.

   (a) Power the op-amp chip by using 5 V for VDD and ground for VSS. You may use the LM317 regulator or the USB battery pack to provide the 5 V. It's a good idea to disconnect and turn off the battery pack when measurement is not being made.

(b) You can use the 2.5 V LT1009 reference as the bias for the input and output.

(c) The gain is controlled using a 20 kΩ potentiometer (POT).

3. To test the circuit, generate a 1 kHz 100 mVpp sine wave signal using the lab signal generator and connect the signal to the input of the gain stage circuit. Measure the output of the circuit using the lab oscilloscope. What is the maximum peak-to-peak voltage at the output before clipping occurs? What is the gain at this point?

4. Adjust the potentiometer to have an output voltage swing of 3 Vpp for the same input as in the above step. Tabulate and plot the gain from 100 Hz to 1 MHz while keeping the pot fixed. Find the 3 dB cutoff frequency or the 3 dB bandwidth at this gain. Use at least 20 data points. What is the gain-bandwidth (GBW) product of the gain stage?

4.4.2 Active LPF

1. Identify the active LPF portion of the circuit in Fig. 11. The overall filter consists of 2 low pass filters. What is the order of the overall filter and how is it determined from the schematic?

2. Build the LPF circuit on your breadboard.

3. To measure just the filter response, disconnect R4 and R14 from the circuit (Fig. 11). Input a 3 Vpp signal in place of R4 and measure the output in place of R14. Tabulate and plot the frequency response from 100 Hz to 50 kHz. What is the cutoff frequency of the filter?

4. How would one increase or decrease the cutoff frequency?

4.4.3 Gain Stage + Filter

1. Reconnect R4 (R14 is still disconnected) and measure the overall response of the amplifier. Input a 100 mVpp sine wave to the IF input.

2. Adjust the pot for a 3 Vpp output at 1 kHz. Measure the frequency response from 100 Hz to 50 kHz and record the result. What is the cutoff frequency? Compare the results to previous measurements. Plot all of the result for the report.

4.5 Acquiring and analyzing analog signals

Acquiring and analyzing analog signals are common tasks in any embedded system that needs to interact with the real world. It is particularly so for high frequency wireless systems because they are usually used for transmitting and receiving information.

An analog-to-digital converter is the main interface device for acquiring analog signals. However, designing a high quality sampling system with high speed ADCs involves some serious engineering at many different levels and is

well beyond what we can cover in a quarter; in fact the topic can probably become a lab course of its own.

In this lab, we will cut some corners and use sampling sub-systems that have already been implemented for us. The first and mandatory option is to use the computer sound card. High quality sound cards are actually extremely precise (up to 32-bit) sampling systems working at audio sampling rate (up to 192 ksps). Even low-end ones will usually give 16-bit resolution.

So in Experiment 4.5.1, we will use the computer sound card to sample the filtered signal from the function generator we built in previous experiments. A program written in the Python programming language will be used to collect the data and perform a frequency analysis.

4.5.1 Using the computer sound card

1. Set up the system according to Fig. 12.

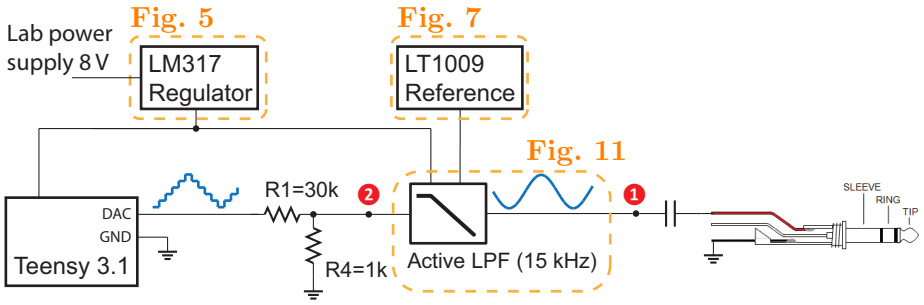


Figure 12: Acquiring signal through computer sound card.

2. Program the Teensy with "triangle_teensy_audio".
3. Verify that the output of the LPF does not exceed 3 Vpp.
4. Connect the output of the LPF to the red wire of the audio cable. Connect the ground reference of the LPF to the black wire of the audio cable.
5. Plug the audio cable into the microphone port of the computer.
6. On the computer, run the python program "realtime_audio_analysis.py". The program is well documented and you should try to understand it thoroughly. Record the data.
   (a) If you are using the lab computer, you can run the program from the Spyder IDE that comes with the Anaconda Python distribution. You could also run the program from the command line by "python realtime_audio_analysis.py".
   (b) If you want to run the program on your computer, you should have Python 2.7 installed. The simplest way to do this is to install the Anaconda 2.7 distribution. You will also need to install the *PyAudio* and *PySerial* packages.

(c) To record the data, you can either modify the Python code to save the raw data, or do a screen capture. Obviously, the former gives you more accurate result.

7. Now use the audio cable to acquire the signal after the gain stage (position 2 in Fig. 12). Record the data and compare it with that from the last step. Explain the difference, if there is any.

4.5.2 (Optional) Using the Teensy's built-in ADC

A second option is to use the Teensy 3.1's built-in ADC module. Fortunate for us, the Teensy 3.1 comes with libraries for directly getting samples from the ADC. In this experiment, we will also be performing signal analysis on the Teensy directly. To display the result, we will add on a simple 128×160 LCD display.

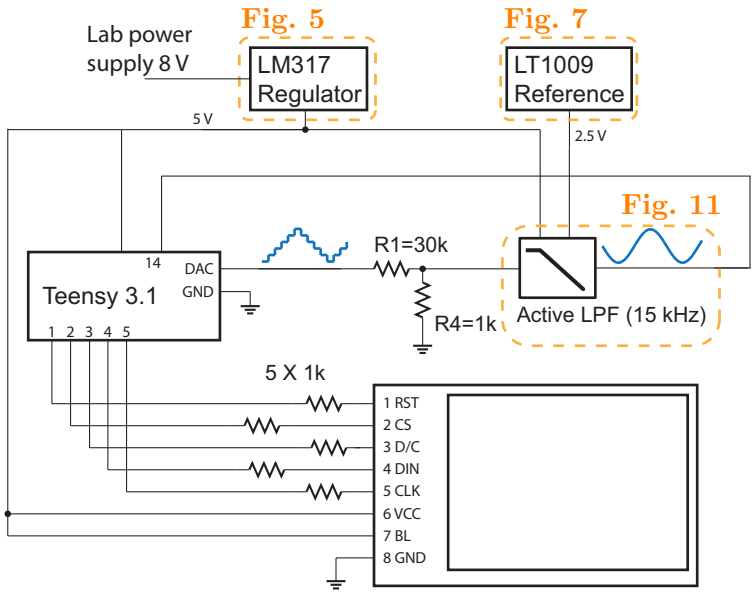
1. Set up the system according to Fig. 13a.
2. Program the Teensy with "sa_teensy.ino". The code generates a 4 kHz triangle wave through the built-in DAC using the Teensy audio library, acquires analog signal input through pin A15, performs a fast Fourier Transform (FFT) to get the signal's spectrum, and displays it on the LCD. In essence, we've built a very simple signal generator and a spectrum analyzer for audio frequencies. The code is well documented and you should try to understand it thoroughly.
3. Observe the display on the LCD screen and compare it with the result of Experiment 4.5.1.
4. Obviously this spectrum analyzer is rather primitive. There aren't any magnitude or frequency labels, nor can you place markers or cursors for accurate reading. Try to implement more features. Send in a pull request through Github if you feel that your code is worth sharing with future students.

4.6 PCB Design

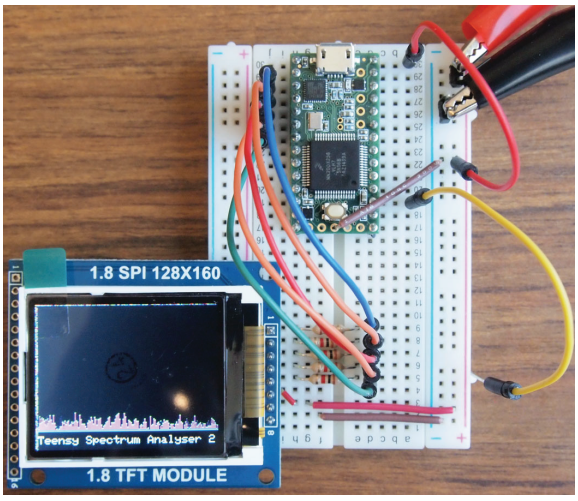
Design a PCB incorporating the circuits of Fig. 5, 7, 10, and 11. We will add a resistive attenuator circuit just so that the output does not easily saturate. Fig. 14 shows a high level block diagram of the PCB.

Follow the guidelines below when designing your PCB.

1. We will use Bay Are Circuits as our PCB vendor. The design constraints can be found on their website.
2. Use 0603 (imperial) surface mount (SMD) components for all the resistors and capacitors. The potentiometer will remain the same through hole package as you used in the lab.
3. For mounting the ICs, we will use pin-headers that you can get from the TA. This shouldn't change the way you design the IC footprint.



(a)



(b)

Figure 13: Acquiring, processing, and displaying the signal all within Teensy. (a) Schematic. (b) An example breadboard layout.

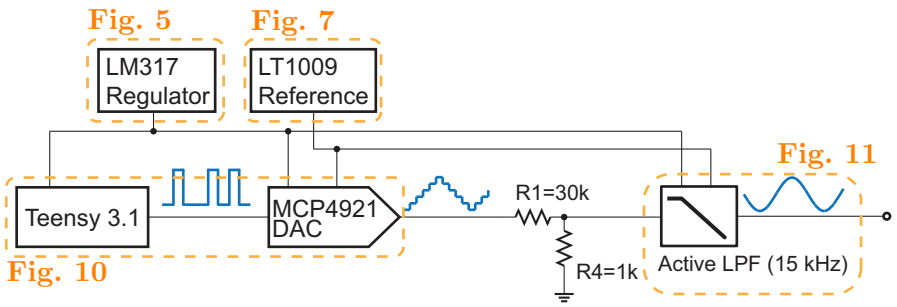


Figure 14: Block diagram of the PCB design.

4. The following items are due by Oct. 23rd, 2015.

   - A PCB design report to the TA. The design report should include the following:
     - Bill of materials.
     - Screen capture of the schematic.
     - Screen capture of the PCB layout.
     - Bay Area Circuit design for manufacturing (DFM) report from showing no errors.
   - A PCB review report to the TA. The review report should be completed by a different team than yours. You will be responsible for finding this review team. The review report should follow the general guideline of the PCB Review Report Template.

5. After the PCB comes back, you will need to assemble the circuit and test it. The tests would be similar to what you have done in Measurement 4.1–4.5. Test report for this PCB is due by Nov. 13th, 2015.