Penn Engineering

LAB 3: THEREMIN

ESE5190 Smart Devices

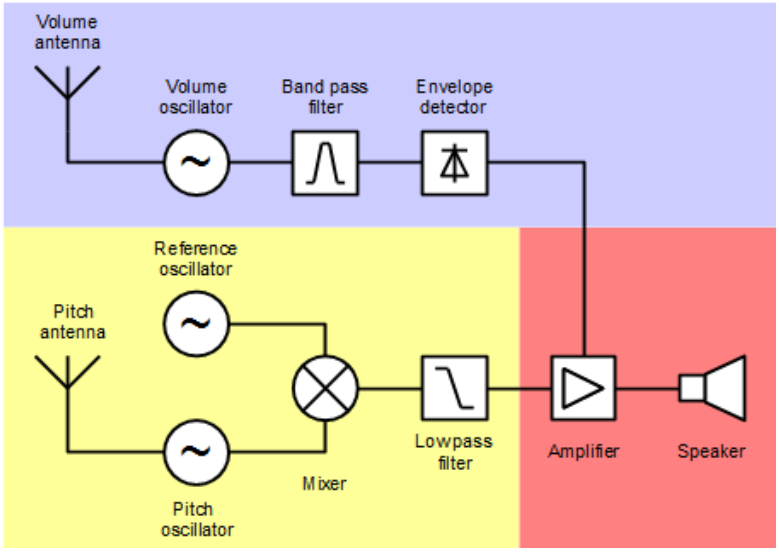
DUE: THURSDAY, OCTOBER 19, 2023 23:59 EDT

Contents

1 Overview

In this lab, you will be creating a musical instrument. It may not be the greatest sounding device but you will get to apply your newly acquired knowledge of microcontroller peripherals! Read through the entire manual before beginning so you have an idea of what the final product will look like.

Behold the Theremin! You can hear a sample of it here A Theremin is an analog instrument - in essence a pair of giant antennas for RF wave oscillators. They are quite large and require a significant amount of power.



We are going to go one step further and build a smaller, portable, digital theremin. Two sensor inputs are used to control the volume and frequency (different tones).

The lab consists of answering a series of questions and submitting a code repository. All the required questions are highlighted in orange. Make a copy of the submission document and record your answers there. Since this lab requires a few video demonstrations, make sure that your links work.

2 Learning Objectives

After this lab, you should understand the following:

1. Timers, PWM, output compare
2. Input capture - pulse width measurements
3. ADC

3 Related Equipment

3.1 Hardware

Elegoo Kit

3.2 Software

3.2.1 AVR Toolchain

It is highly recommended to install Atmel Studio. Refer to the Windows Set-up Guide and OSX Set-up Guide page to set up your programming environment if you haven't already done so. Mac users are welcomed to use the VSCode + PlatformIO set-up. There are guides to set up your environment to program your microcontroller on different platforms.

3.2.2 GitHub Repository

Click on this link to create your repository on Github Classroom. To facilitate easier grading, your repository name should include your pennkey. Please rename your repo to be labx-pennkey.

4 Part A: Printing to Screen

Printing to the screen will be very useful for debugging in this lab. Write your own UART code for printing to screen now that you have learned about the different ways of serial communication. Do not use the library provided in Lab 2. **Yes, your code will be similar** but the goal here is for you to write your own code and understand the set up. You can write it as a library to include in your main file or simply have the functions written directly in the main file.

5 Part B: Waveform Generation

The Arduino Pinout and the ATmega328P datasheet will now become very useful throughout the duration of this entire lab. These are available in Canvas under the 'Datasheets and Application Notes' module.

5.1 Timer Overflow

We will first verify that we can generate a waveform that will translate to an audible tone. There are two buzzers in your kit, an active buzzer and a passive buzzer. The active buzzer has an internal oscillator and when you apply a voltage directly to the leads, a sound will generate. The passive buzzer requires an AC signal (e.g. PWM wave) to generate a sound. If you apply voltage to the leads, there will be no sound produced. Use the **passive buzzer** for this lab as we want to vary the PWM to produce different tones.

Locate the **passive buzzer** and the **NPN Transistor (PN2222)** in your kit and connect it like the circuit below. **The buzzer has polarity so make sure the positive lead is connected to power and the negative lead is connected to the collector lead of the NPN transistor**. See the PN2222 datasheet for the connection of each leg. The schematic symbol for an NPN BJT can be found here.

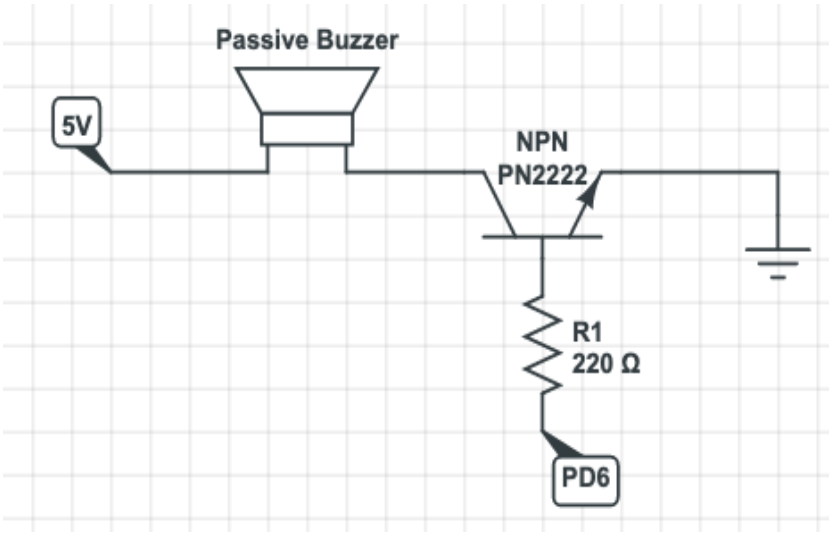


Figure 1 Circuit diagram for buzzer

Have Timer0 toggle its output when Timer0 overflows. Use interrupts. An audible sound should be emitting from your buzzer.

- What frequency is being generated here? Is it what you expected? Show your work.

5.2 Normal Mode

Generate a 440Hz square wave using Timer0 in **Normal mode**. Hint: PD6 should be *toggled* in and interrupt when OCR0A reaches a certain count. You may need to update OCR0A each time. If you don't want to hear the annoying sound of the buzzer, just hook up the output pins to your oscilloscope instead.

- Did you have to prescale the system clock and/or timer clock? If so, by how much?
- What number should OCR0A be in this case?
- Attach a screenshot of your code snippet or copy and paste the snippet into a box in the submission document. It should be quite short.

5.3 CTC Mode

Generate a 440Hz square wave using Timer0 in **CTC mode**. You should not have to use any interrupts.

- What number should OCR0A be in this case?
- Attach a screenshot of your code snippet or copy and paste the snippet into a box in the submission document. It should be quite short.

5.4 PWM Mode

Generate a 440Hz square wave using Timer0 in **Phase Correct PWM mode**. You should not have to use any interrupts.

- What number should OCR0A be in this case? Specify which Phase Correct mode you use - namely, what is the TOP value. (Refer to Table 14-8 in the datasheet).
- Attach a screenshot of your code snippet or copy and paste the snippet into a box in the submission document. It should be quite short.

6 Part C: Measuring Distance

We will be using the HC-SR04 ultrasonic sensor to measure distance. An ultrasonic sensor measures the echo of a sequence of high frequency sound waves to detect distant objects - similar to a bat using echolocation.

- Reading through the datasheet, what is the length or duration of the pulse that needs to be supplied in order to start the ranging?
- What are the **Trig** and **Echo** pins used for?

Keep in mind that you don't want to send another trigger signal until after you have detected the full return pulse width from the echo pin. You have the option of using the same pin, the input capture pin, to send out a trigger and then read in the echo. In this scenario, you will have to make sure that you toggle between making the port an input pin and an output pin. This prevents a second trigger signal from being sent before the echo has been received. If you use two separate pins for Trig and Echo, then you'll need to make sure that the trigger doesn't send a signal until the echo pulse has been received.

Connect the pins of the ultrasonic sensor accordingly into your circuit.

The oscilloscope will prove to be very useful here. In addition to verifying that you are indeed sending out the output pulse, you can check the Echo pin to see if the returning pulse width varies as expected (move your hand or an object back and forth in front of the sensor).

Write a program that uses the ultrasonic to print out measurements (in ticks or in cm, up to you) as you move your hand or an object back and forth in front for the sensor.

- What is the largest distance (in cm) that you observed printed out in the terminal?
- What is the smallest distance (in cm) that you observed printed out in the terminal?

7 Part D: Generating Different Tones

In the music world, an octave is a double of frequency. E.g. Frequencies that range from 440Hz to 880Hz is an octave. The octave gets its name from the fact that there are 8 notes in a major scale. 440Hz to 880Hz correspond to the musical notes: A, B, C#, D, F#, G#, A.

For our digital theremin, we are going to the C6 to C7 octave. Higher frequencies correspond to higher pitched notes/sound and lower frequencies correspond to lower pitched notes.

Use Timer0 in PWM mode to generate the frequencies and observe your output on the oscilloscope.

- Fill in the table below with the correct OCR0A values that will yield the required frequency. You will have to choose a prescaler that will allow for the entire range to be generated with just one timer. Rounding errors will be expected.

Note	C6	D6	E6	F6	G6	A6	B6	C7
Freq (Hz)	1046	1174	1318	1397	1568	1760	1975	2093
OCR0A								

7.1 Continuous Frequency

Now, we are going to connect what we've done with the ultrasonic sensor with the frequencies generated above. Given the OCR0A values above, derive a linear formula that takes in a measurement value and outputs the corresponding OCR0A value.

In other words, the closest distance read by the ultrasonic sensor will correspond to 2093Hz and the furthest distance read by ultrasonic sensor will correspond to 1046Hz. I.e. Given a sensor reading of say 535, what frequency should be outputted? (No need to answer this question, it's just a rhetorical question meant to provide clarification.)

- Write your linear formula here. It should look something like: FREQ (or OCR0A) = SENSOR_VALUE * SOME_RATIO + SOME_NUMBER

Write a program that toggles the buzzer at frequencies that relate to the distance of an object in front of the ultrasonic sensor. You may use any waveform generation mode, however the PWM modes will prove to be the most useful. Closer distances correspond to higher frequencies while further distances correspond to lower frequencies.

- Take a video of the buzzer changing frequency as your hand moves back and forth in front of the ultrasonic sensor.

7.2 Discrete Frequency

What if instead of a continuous range, we want a discrete range instead? Divide the range of the ultrasonic sensor readings into **8 discrete ranges** so that as your hand moves further (or closer) away from the ping sensor, the sound emitting from the buzzer changes in discrete steps.

Write a program that toggles the buzzer at frequencies that relate to the distance of an object in front of the ultrasonic sensor. Closer distances correspond to higher frequencies while further distances correspond to lower frequencies.

- Take a video of the buzzer changing frequency as your hand moves back and forth in front of the ultrasonic sensor.

You will want to add a button to your circuit so that when the button is pressed, the user can toggle between discrete mode and continuous mode.

8 Part E: Adjusting Volume

A voltage divider can be used to adjust the volume of our instrument. A photoresistor varies its resistance according to how much light it sensors. The brighter the light, the lower the resistance. Connect the ADC output of the circuit below to **PA0**.

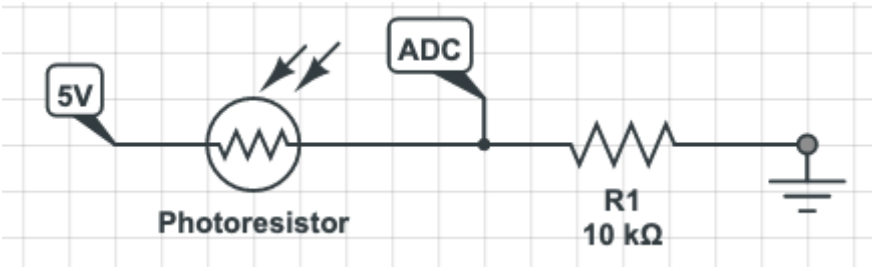


Figure 2 Photoresistor circuit

Set up ADC so that it will read in the voltages at PA0. Print the voltages to the screen and vary the amount of light on the photoresistor to verify that your ADC value is changing.

To maximize the range of the readings, try shining a bright light (e.g. flashlight on a cellphone) on the photoresistor, get a reading, and then use your finger or a cloth/cover and cover the photoresistor completely to get a reading when there is no light detected.

- What is the maximum and minimum ADC values read?

To control the amplitude of the buzzer, the duty cycle of the PWM wave will be modulated. A smaller duty cycle will correspond to lower volumes and a larger duty cycle will correspond to higher volumes. We will cap the duty cycle at 50% to keep the suffering of the ears to a minimum.

Divide the range you observe from the photoresistor into 10 discrete segments. The 10 segments will correspond to 10 duty cycle segments.

- Fill in the table below with your ADC ranges.

ADC Ranges	Duty Cycle
E.g. 0 - 100	5%
	10%
	15%

	20%
	25%
	30%
	35%
	40%
	45%
	50%

Write a program that takes in the ADC values of the phototransistor and depending on which discrete segment the value falls in, prints the duty cycle to the terminal. E.g. "ADC: 10, Duty Cycle: 5%"

9 Part F: Putting it All Together

Finally, we are going to put all the parts together. One of the PWM modes in Timer 0 may come in handy here.

Your final program and circuit must fulfill the following requirements:

Mode Selection: The user can switch between continuous mode and discrete mode by pressing a button. You can either write your program so that the user must keep the button pressed to be in a different mode or the user just presses and then releases the button to switch to a different mode. Hint: Pin change interrupts might be useful here and you may need to debounce. If you prefer to not to use a button, you can disconnect and connect a wire attached to 5V.

Continuous Mode: Frequency emitted from buzzer changes in a *continuous* manner. Frequency changes relative to the distance detected by the ultrasonic sensor.

Discrete Mode: Frequency emitted from buzzer changes in a *discrete* manner. Frequency changes relative to the distance detected by the ultrasonic sensor.

Volume Control: The volume emitted by the buzzer corresponds to how much light the photoresistor receives. You can choose to vary the volume in a discrete manner according to the ranges in question 18 or in a continuous manner. Mention which version you choose to do in your lab write up.

You do not have to use the pins/ports that we have been using thus far in the lab. Feel free to rewire the circuit and use other peripherals that you feel would be more relevant - e.g. maybe Timer2 could come in handy.

- Draw your final circuit in Circuit Lab and attach an image.
- Take a video of your final product working, clearly showing the four requirements above and attach the link.

Save your file as `main.c`.

10 Part G: Extra Credit

Note that since this is extra credit, the teaching team cannot provide detailed help. At most, we will only give suggestions or hints.

- Why is there a resistor required at the base of the transistor in Figures 1?
- Why is a BJT used instead of a MOSFET in Figure 1 and can a MOSFET be used instead?
- What additional feature would you want to add to this "instrument"? Outline how you would implement it.
- Generate a 440Hz sine wave. Take a video or image of the result, include the image or link to the video, attach a diagram of the circuit, and upload the code into your repo with the name sinewave_ec.c

11 Submission Requirements

Click on this link to create your repository on Github Classroom, if you haven't already. Name your repo to be `lab3-pennkey`. (e.g. If my pennkey is 'dracaena, then my repo name will be `lab3-dracaena`.)

Make a copy of the submission document and record your answers there. Name your document "`lab3_pennkey.pdf`". (e.g. If my pennkey is 'dmalfoy', the title of my submission will be "lab3-dmalfoy.pdf") **You will lose points if we cannot open your links on the first try so make sure your video links work.** You can try opening the links in an incognito tab.

The following files are required to be in your repository:

Any code written for this lab, separated into a .c file for each part

main.c

Any additional dependencies (optional)

lab3_pennkey.pdf

README including any additional information that would help us grade your lab report (optional)

Submit your pdf on gradescope. Access it via canvas for easier login. If required, the access code is MK2GZ2. Make sure you are using your seas/upenn account.

12 Grading Rubric

Only the following files will be graded:

- main.c
- lab3_pennkey.pdf

If your solution(s) used any additional files or dependencies, you MUST specify this in your README.

Submission Requirements	05 pts
Lab Write Up	
Questions 1 - 12	36 pts (3pts x 12 Questions)
Question 13	16 pts (2pts x 8 Cells)
Questions 14 - 17	12 pts (3 pts x 4 Questions)
Question 18	20 pts (2pts x 10 Cells)
Question 19	05 pts
Question 20	40 pts
main.c	40 pts
Part G: Extra Credit	
Questions 21 - 22	04 pts (2pts x 2 Questions)
Questions 23 - 24	06 pts (3pts x 2 Questions)
	174 pts (excludes extra credit)

[1] Theremin Block diagram