Lab #0: Hello World plus speaker for tone generation

ECE-167/L: Sensing and Sensor Technology

University of California Santa Cruz

Purpose

This lab is intended to familiarize you, the student with the basic hardware and microcontroller that you will be using for the rest of the quarter. The micro is an uno32/uc32 with our own custom bootloader (ds30) on it. Follow the instructions below to complete the lab. Keep in mind that the reasoning behind these instructions should be understood, not just the instructions themselves.[1]

Open MPLAB-X and Create a New Project

Follow the instructions outlined in CMPE167_MPLABXNewProjectInstructions.pdf. Be sure to understand the difference between absolute and relative paths when making the project, or you will risk deleting your project when running the code updater.

Setup the Uno32 Microcontroller

First, the ds30 loader .hex file should be burned on to the pic32. This requires a pre-built project and use of the picKit3. In lab section there will also be a special pickit3 that will allow the TA to flash the bootloader.

1 Hello World!

After setting up a new project in MPLAB-X, and after burning on the ds30 bootloader on to the micro, write a program that can print "Hello World!" via serial communication from the microcontroller to the ds30 terminal.

[1] This will get more important as the class progresses, but you need to understand the lab and its purpose, not just plow through.

2 Hello A/D (Reading an analog signal)

Print the A/D reading from the potentiometer on the microcontroller shield using the AD library. Make sure to have absolute file paths when including the AD library. One AD pin in particular corresponds to the onboard potentiometer. You can check the datasheet to figure out which one it is.

3 Tone out speaker hard coded

Generate a tone with the speaker using the tone generator library. You will have to connect a PWM pin from the microcontroller (specifically a PWM pin which this library controls) to the audio amplifier PCB. MAKE SURE TO USE A JUMPER WIRE - NOT A REGULAR WIRE. If you don't understand the reason for this distinction, you should figure it out soon! There should be some frequencies already defined in the header file (ones which sound good), but you are welcome to define more frequencies in your main file (frequencies that might not sound so good).

4 Tone adjusted via POT

Now use the potentiometer A/D reading to change the tone of the speaker. Keep in mind that changing the tone means changing the frequency of the PWM signal. You will notice that changing the frequency with the on-board potentiometer sound really scratchy and bad. Why? Implement some software filtering to eliminate the scratchy sound. Explain in you lab report what methods you used. You will want to try some different ideas, but be sure to write about what worked and what did not in your report.[2]

5 Basic filtering of output (single pole RC to smooth tone generation)

By now you may have noticed that the sound from the speaker is not very "smooth." To fix this, you will need to use a low-pass filter to smooth the signal coming from the micro to into the audio amplifier. How does the filter change the sound of the speaker? What is the cutoff frequency for the low-pass filter, and why? How does the low-pass change the square wave?

6 Make some music

Build an instrument to play music with your now harmonious assembly of microcontroller, low-pass filter, audio amp, and speaker. The requirements for the instrument are below. Map four tones to the four buttons on the I/O Shield. There are tones provided in the header file but feel free to use your own. Read the POT (with appropriate filtering) and use the value as an offset to the tones provided. This means that while the difference between tones will remain same the average frequency changes. Experiment with the offset scaling until you find one that makes sense. While one of the buttons is held down, that augmented tone will be

[2] **Hint:** The potentiometer is probably not linear. You might want to find a polynomial fit, and code it as a function of the potentiometer reading to really linearize it. This can be done using many techniques which we will go over in detail down the road.

played through the speaker. Releasing the button will stop the tone. If more than one button is pressed we do not expect a tone to play but feel free to make better behavior if this does occur.

7 Check-Off and Lab Report

Demonstrate your fully functioning instrument to the TA's. Do not forget to submit your commit ID including your completed code to the google form. Congratulations! By this point you have completed the first lab! Now you must write your lab report.

In the lab report, ensure that a student who has already taken the class and is fairly well versed in embedded software would be able to recreate your solution. Include details and algorithms, pseudo-code, flow-charts, and any "bumps and road-hazards" that a future student might watch out for. Please be clear and concise, use complete sentences, and maintain a dispassionate passive voice in your writing.[3]

[3] English teachers hate the passive voice, but it is used as the primary narration in scientific writing as you want the reader to know what was done, but not be concerned with who did it. In this sense, proper scientific writing is *egoless*. If passive voice is too burdensome for a particular sentence, use the "we performed" or other such construct instead.