User-Controlled Harmonizer

Justin Xiao

April 2017

1 Introduction

When performing music live as a singer, it is often desired to introduce harmonies to your singing. However, if one is a solo performer, this could prove difficult. Now, if one were able to control the harmonies flexibly, they would do away with the need of backup vocalists and thus achieve their artistic vision with full flexibility.

To do so, I propose the construction of a harmoniser. This device would receive two signals - the vocal audio from a microphone, and a set of pressed keys on a keyboard. At the output would be the original audio signal, with pitch-shifted versions of it mixed in as determined by the keys pressed. For example, if I were to sing a C into the microphone while pressing the E key on the keyboard, the output of the speaker would be my voice singing both a C and E, in real time.

As a singer and musician, I think the creation of such a device could greatly expand my musical creativity. As of yet, few such device exists, including one created in the MIT Media Lab by Ben Bloomberg and used by musician Jacob Collier. After seeing Jacob Collier in concert, using the harmoniser, I was inspired to make this project come to life myself.

2 Hardware Description

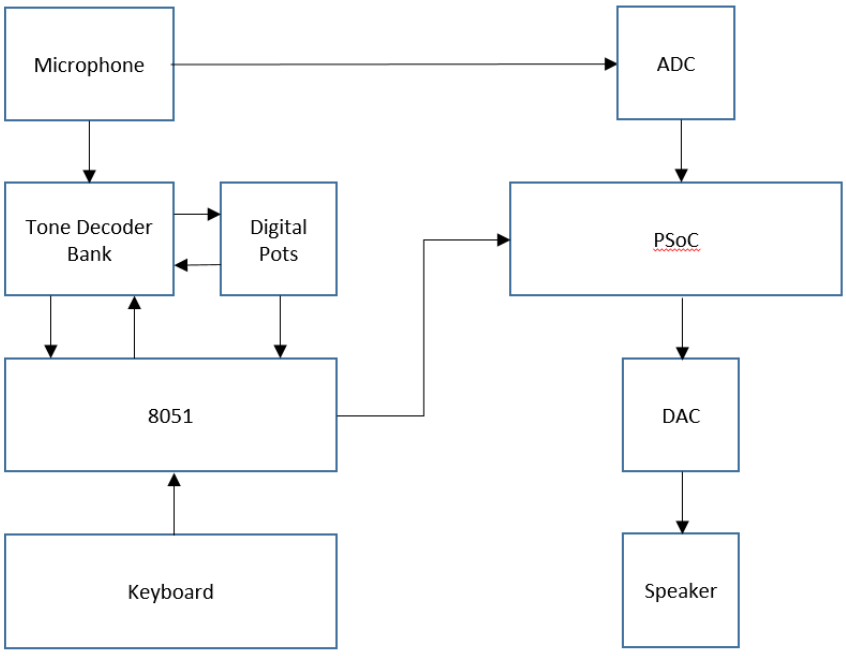

Figure 1: Hardware design functional diagram

The hardware design is as follows. A vocal signal from a microphone is fed through a bandpass filter and then through a bank of LM567 tone decoders. These chips function by outputting low when a tone of some frequency is detected within some bandwidth, set by external components. By sweeping some digital potentiometers across some resistances, we can quickly check what frequency is input in the microphone by determining the resistance at which we measure an output from the tone decoders. This reduces computational complexity involved with digitally determining the frequency, freeing the microcontrollers to perform more complex calculations.

While the 8051 scans the tone decoders for input frequency, it also scans a keyboard (row of buttons) to check which notes are being pressed. This can be done by connecting individual buttons to to the ports of an 8255 chip and checking which bits are high.

In parallel, the microphone signal is fed through an ADC0820 to convert the signal to digital, such that the PSoC can read it. The 8051 communicates serially with the PSoC, sending it data on frequency and notes pressed. After executing a pitch shifting algorithm in the PSoC, a digital signal comprising the original audio and its pitch shifted variants are output through the DAC connected to a speaker.

3 Software Description

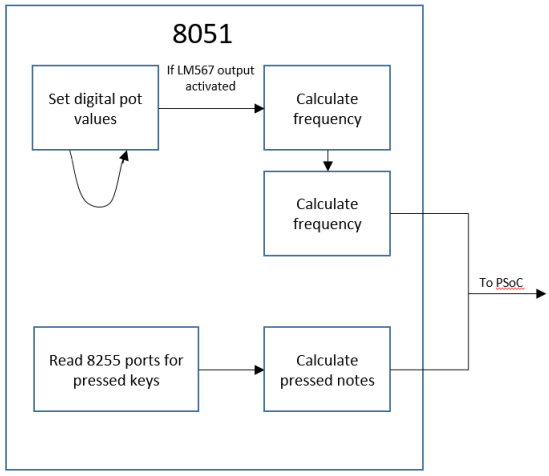

Figure 2: Software design functional diagram for 8051

On the software side, the 8051 runs relatively simple functions. Using interrupts, the 8051 can periodically sweep the digital pots and check the outputs of the LM567, mapping these outputs to frequencies and hence to notes. Simultaneously, the 8051 scans a row of buttons to determine which ones are being pressed, and maps those to desired notes. These are then serially sent to the PSoC.

The PSoC, being a much more powerful controller, does most of the digital signal processing for pitch shifting. A block diagram of its logic flow is shown ing Fig. 3. Sampling the microphone audio, the streaming bits are stored in a buffer array of some fixed length. Following the PSOLA pitch shift algorithm in order to preserve formants such that the shifted voice doesn't sound like a chipmuk, the PSoC first uses our detected frequency from the LM567s to mark the audio signal at period places, particularly at local maxima. We then apply Hanning windows to these markers to retrieve a segment of the signal. To shift pitch, we move these windowed signals closer for a higher pitch and further for lower pitch and add them. Finally, to preserve the time, we resample and add or subtract windows.

4 Project Scope

My goal for a reasonable and competent project is to achieve a pitch shift controllable by a keyboard.

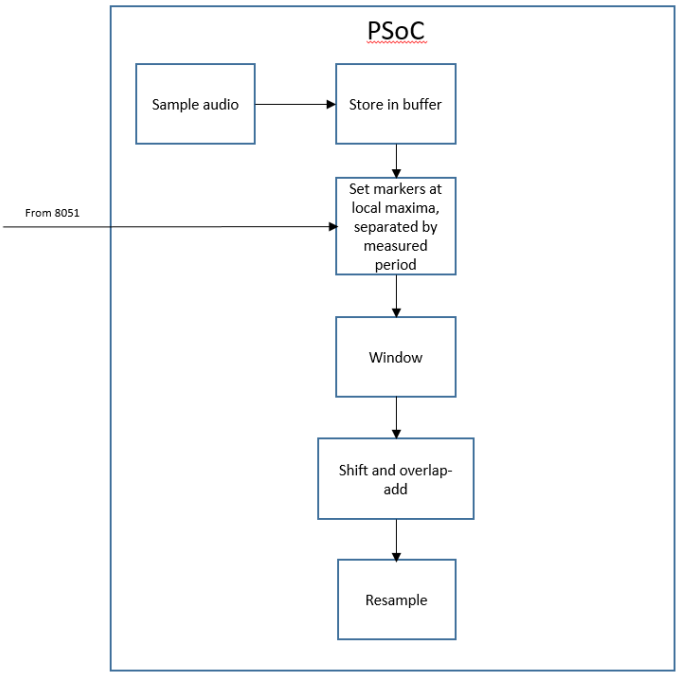

Figure 3: Software design functional diagram for PSoC

My goal for an excellent project would be to control pitch shift with a keyboard while preserving formants and allowing for polyphony

As a high risk goal, I would like to introduce large polyphony (e.g. up to 12 added voices) to add to the original voice, controllable by MIDI keyboard.

5 Special Components

Some special components which I will need are as follows:

8255 - Extra input/output ports

Button array - I will need to construct a button array that can be used as a keyboard

LM567 - tone decoder

Digital potentiometer

ADC0820 - Analog/digital converter to allow audio signals to be read by microcontrollers

AD558 - Digital/analog converter to convert microcontroller output to audio

Microphone - record input audio

6 Timetable

A planned timetable is as follows:

During the week starting 4/17, I will:

- Implement the PSOLA algorithm in Python
- Begin porting implementation to C and in real-time

During the week starting 4/24, I will:

- Finish a basic version of PSOLA for PSoC
- Begin adding peripherals (Digital pots, tone decoders, amplifiers)

During the week starting 5/1, I will:

- Refine PSoC PSOLA
- Finish adding peripherals (buttons), write 8051 code to independently control peripherals

During the week starting 5/8 I will:
- Refine PSoC PSOLA as needed
- Refine hardware peripherals
- Write software to control all peripherals simultaneously on 8051
- Implement communication between PSoC and 8051

During the week starting 5/8:
- Complete the system's logic
- Refine the harmonizer software on PSoC

During the week starting 5/15, I will:
- Debug and refine, finish any loose ends
- Add reach features (polyphony, MIDI input, etc.)