

Project Final Report

Automatic Volume Control

4/27/2007

Gedare Bloom, gedare@gwu.edu

Project Abstract

Automatic Volume Control (AVC) is primarily for active use with a television set in order to maintain a specific level of volume. A user selects a comfortable volume level from their multimedia device, and the AVC uses infrared remote capabilities to maintain the comfortable volume (within some threshold). A ZNeo with an infrared transceiver is used, with an omni-directional electret microphone for capturing sounds. A user sets the volume down command by pressing a button on the ZNeo and then pressing the volume down on the remote control. The comfortable volume level is similarly chosen by pressing a button, at which point the microphone is sampled for a small amount of time and the maximum value in that duration is used as a max threshold.

The current implementation only provides maximum volume limiting capability. The interesting work was in figuring out how to capture an infrared signal and then retransmit it. Determining volume for a maximum threshold turned out to be relatively straightforward, by simply attaching a microphone to the analog-to-digital converter of the ZNeo and reading the values directly.

Status

The infrared capabilities work quite well, which I was very pleased about. I tested the ability to capture a variety of remote protocols successfully. One particular protocol that was tested that did not work with my infrared capturing library is the JVC protocol. Capturing the volume level of sound works well, but I did not manage to implement when the sound is too *soft* and adjusting the volume upwards. In reflection, I think that a simple approach that reads in maximum and minimum analog values, and compares the maximum to the threshold and the maximum-minimum to be sure the volume is not completely silent, would provide a good basis for performing more refined volume control. Additionally, adding a mini-amp to the microphone output would probably make it easier to identify the existence of volume that is *soft*.

I did not produce a high level user-interface for my project, but I think such an interface would be simple to create. The hardest part of the project was in creating the infrared capturing capabilities. I provide a documented API for performing infrared capture using the ZNeo, and explain the logic of the algorithms so that it might be adapted to other microcontrollers.

Specification

Platform: Zilog ZNeo development platform.

Platform Capabilities: GPIO (General Purpose I/O), Infrared transceiver (ZHX1810), analog-digital converter (ADC), buttons, timers.

External hardware: CB/PA/Mic Condenser Omnidirectional Microphone Element from RadioShack (part number 270-092), solder-less breadboard, handful of wires and a resistor.

Software Modules: Infrared capture/transmit API, audio processing/decision routine, button manager

Basic Concept

A user should be able to *teach* the device how to issue remote commands for the particular multi-media device that is to be volume-controlled. Additionally, a comfortable level of volume must be selectable by a user. For my project, I accomplish both tasks with the use of buttons as a low level user-interface for indicating to the microcontroller that the user is about to issue a remote command that should be read, or that the microcontroller should record the current maximum volume to use as a threshold value. A more refined solution could be created.

Infrared capture/transmit

Performing infrared capturing or transmitting is a difficult task. An alternative approach would be to provide the ability to select a particular remote protocol, similar to the manner in which universal remotes operate. For recreating this project on the ZNeo, I provide an infrared API that should help solve some of the problems of capturing and transmitting infrared signals. The important pieces of information that are needed in order to transmit an infrared remote control signal are:

- the period of the modulated signal
- the duty cycle of the signal for each period
- the durations of marks and spaces

See the associated infrared documentation for information on setting up the ZNeo for infrared-readiness.

Audio processing

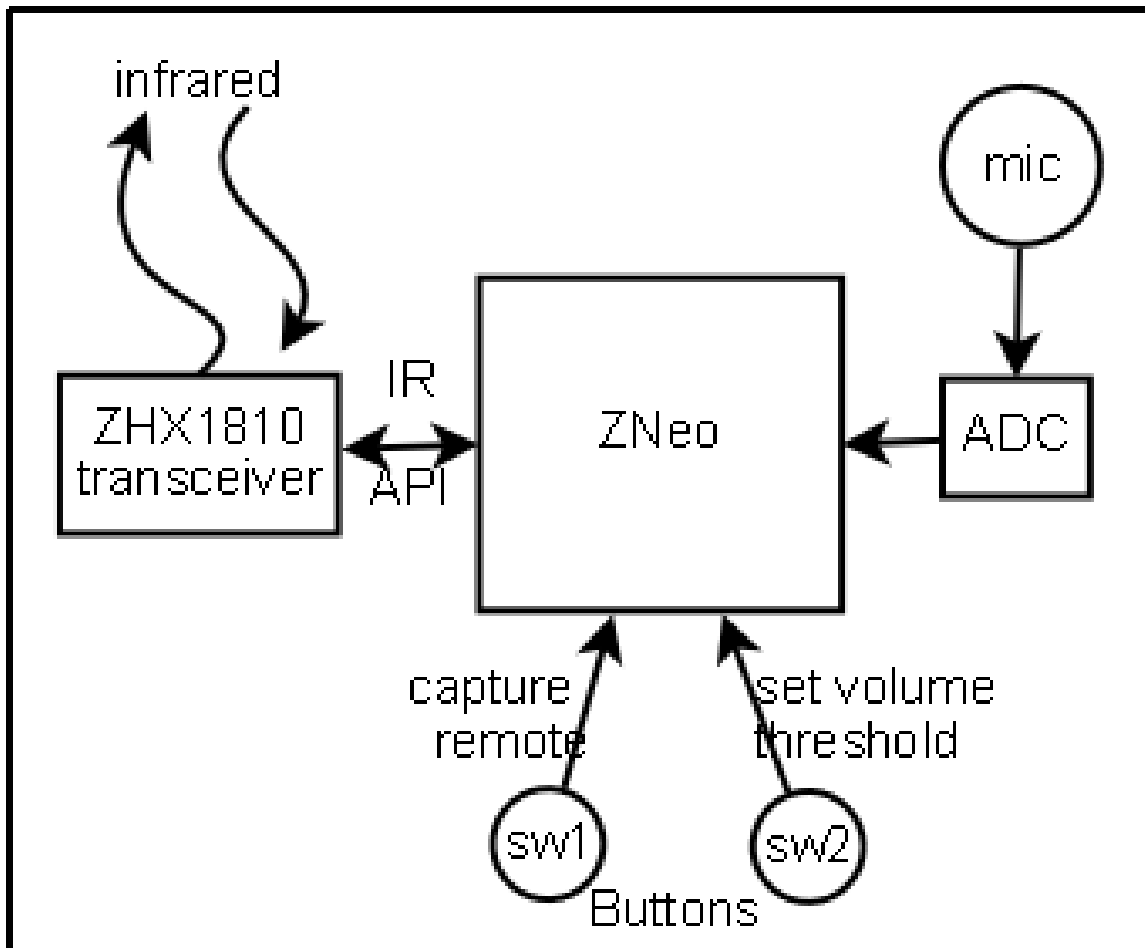
Audio processing involves attaching the microphone to the ADC and reading analog values across a span of time, using the maximum value in that time to decide if the volume should be reduced. This is the main portion of the application that performs volume control, and is actually the least refined part of my project. A *better* decision algorithm could be constructed however I have found that the simple solution that was implemented seems to work well in most situations. See the section with ‘Construction’ for information on how to set up the microphone.

The user selects the comfortable volume level by pressing a button on the ZNeo when the volume of the speaker next to the microphone is generating noise similar to what is desired for the user. The button interrupt service routine reads the analog signal

from the microphone for an arbitrary amount of time (approximately 100,000 analog conversions) and uses the maximum of those readings as the threshold. There is no rationale for the number of readings performed, but too few readings may miss the maximum analog value (which corresponds to the highest amplitude, which is the greatest volume of a sound signal).

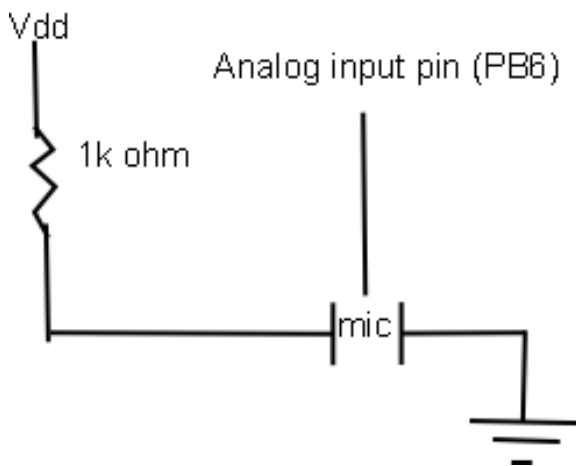
Implementation & Construction

Hardware block diagram



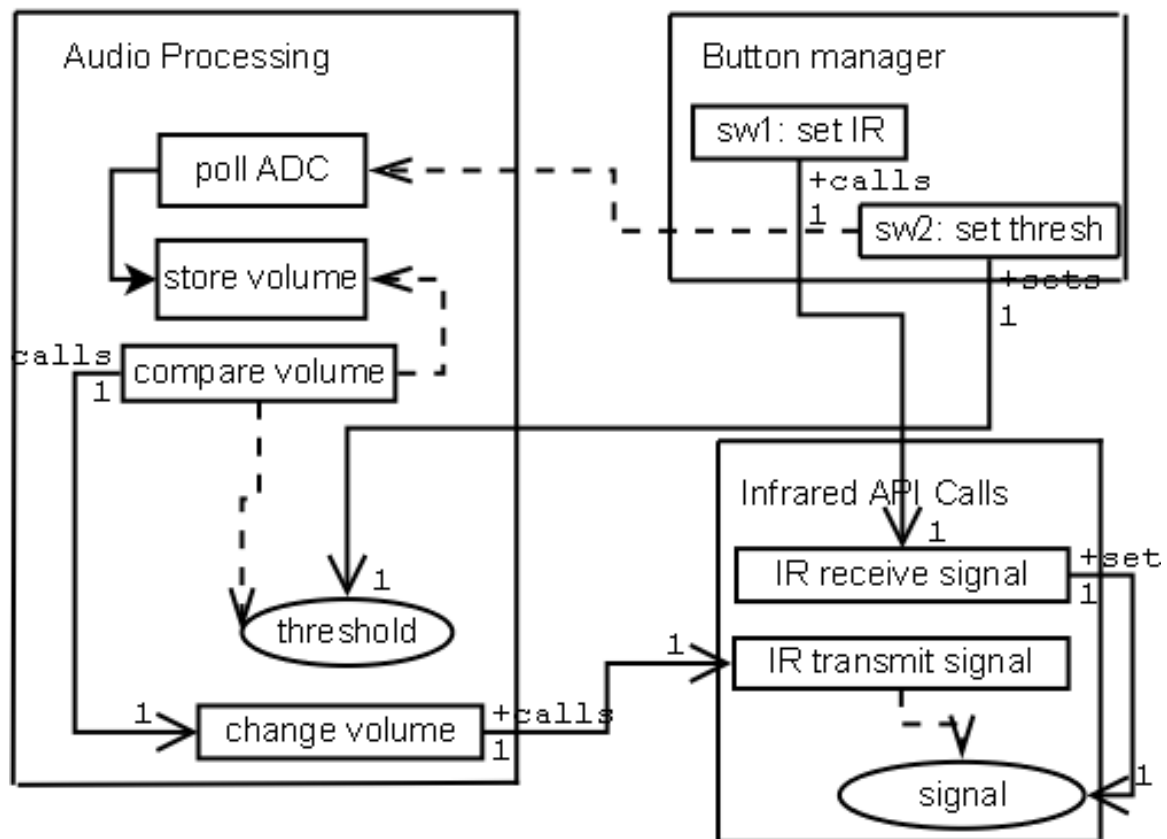
The microphone should be attached to an input of the ZNeo's analog-to-digital converter. See the following schematic for more details.

Microphone Schematic



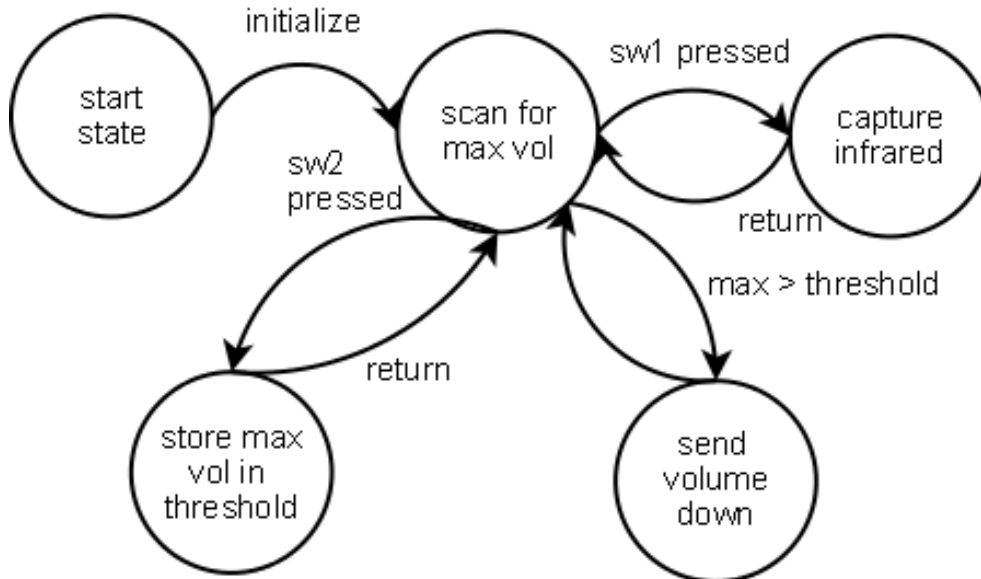
Software Diagrams

Software Block Diagram



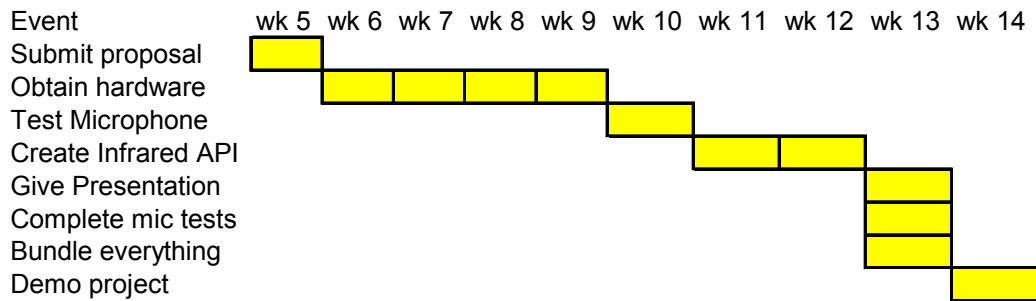
Here we can see the interactions of the various necessary components in the AVC system. 'threshold' and 'signal' are shared variables. Dashed lines indicate dependence.

State Diagram



Here we see the state diagram for the various phases of the decision algorithm that implements volume control. The ‘scan for max vol’ state is the default action that is taken whenever no other event is occurring. For more details on how ‘capture infrared’ and ‘send volume down’ operate, see the appendix information on infrared capture and transmission.

Milestones



Retrospective

Design decisions

Figuring out how to perform the infrared capture and transmit was an important and time-consuming process. However, by focusing on the infrared implementation first, I was able to put the most energy into refining that portion which led to a fairly straightforward integration of infrared software and microphone hardware for the final project. This was very influential because it made my project have a broader appeal than if the infrared signals needed to be hard-coded to be operational.

The microphone element had some troublesome decisions surrounding it. For one thing, most of the projects that I found using these types of microphones involve using an

operational amplifier in the circuitry. When I tried to use the op-amp that comes packaged with the ZNeo, the gain swamped my input and I was having trouble being sure the microphone was working. However, without the op-amp gain, I was able to notice that volume change is noticeable in the hundredth's position of the voltage reading from the ADC (which corresponds to only one or two bit differences). So the microphone just barely gives enough information to provide volume control without use of an operational amplifier. This was good news, because it simplified the hardware and reduced the overall complexity of bundling the infrared API and the microphone tests.

I decided not to perform noise-filtering, because initial tests showed that by locating the microphone close to the speaker was enough to overcome most ambient noises. Looking back, I was most concerned with how to capture the volume levels and perform decisions based on volume, but that part of the project turned out to be drastically simpler than the infrared transmission.

What I learned

I learned a lot about how infrared remote signals operate. I used the knowledge to implement the infrared capture/transmit API and give a presentation on infrared remote signals. Dealing with wiring the microphone made me have to become more familiar with design of simple circuits. Also, in order to figure out how to make the ZNeo perform direct infrared signal transmission and reception, I had to really become familiar with reading schematics and processing datasheet information. Everything that we did during the class was useful and reinforced some of the techniques that we were taught as 'good practice'.

What I would do differently

I'm not sure that I would do much different. I would probably make certain that my hardware choices are going to work (e.g. that the microphone will allow me to make decisions) before the deadline is looming. However, everything seemed to work out well.

I might not be so excited to perform infrared capturing and re-transmitting, knowing how much trouble and effort it took to get it working well.

Attachments

Program

Infrared API

ZNeo Product Description

Infrared API documentation / summary paper