Theremin+

From wiki-class
Jump to: navigation, search

Created by: Nicholas Harris, Caleb Stephens, Justin Matthews

Contents

[hide]

Objective

The overall goal of our project is to create a theremin, an electronic musical instrument

that creates sound based on the user's body position, which generates waveforms from the Arduino Uno. This will be accomplished by using a series of buttons that will make up the selection mechanism for the different waveforms, a microcontroller to make the selection for the output, and ultrasonic sensors that will serve as the detection for the user's body position.

In addition to what a "normal" theremin is capable of doing, we will also add buttons that modulate the signal to produce different sounds (much as a synthesizer is capable of doing). With this, the user will be able to select between several different sounds to be produced from the system.

Project Proposal

Link to File

Project Progress

Week 1

(10/13 - 10/19)
☑ Get approval for parts

Week 2

(10/20 - 10/26)
☑ Obtain parts
☑ Begin program development

This was the initialization of our prototype.
We had some initial challenges with the hardware, which we overcame by switching to an Arduino Uno.
Spent the majority of our time in this phase testing the capabilities of the ultrasonic sensors.
Confirmed that we would be able to take samples from both sensors at the same time at a high enough frequency to feed into our calculations.

Week 3

(10/27 - 11/2)
☑ Start prototyping
☑ Calibrate Sensors to determine output :: Ultrasonic sensor testing

Here we wired up the ultrasonic sensors to the same Arduino and programmed a readable output for our testing purposes.
Looked into how all of our components would be able to fit into our housing for the prototype.
Began looking at various methods of input for the modulation buttons.

Week 4

(11/3 - 11/9)
☑ Work on housing for theremin
☑ Continue coding progress :: Note definition testing

Furthered progress on the housing, and determined how the dowel rod would be attached.
More questions arose about how we would be able to sample at a high enough frequency to produce sound with minimal delay (lead to the note definition lookup table).

Week 5

(11/10 - 11/16)
☑ Work on housing for theremin
☑ Continue coding progress :: DAC testing

We began testing the sine waves being pushed through the speaker, by implementing a DAC to our design.
The challenge here was getting the sine wave to not be distorted with the added load from the speaker.

Week 6

(11/17 - 11/23)
☑ Finish main coding operations (have a functioning sinwave)
☑ Start implementing modulations :: Button Testing

We settled on using a 16 button keypad for the modulation implementation.
In order to produce the right sampling rate from the ultrasonic sensors, we decided to utilize a second Arduino Uno to sample/control modulations,
while having our original Arduino calculate waveforms and push the sine waves to the speaker.
Began sodering the components

Week 7

(11/24 - 11/30) [Thanksgiving week]
☑ Debug
☑ Work on speakers :: Waveforms

Calculated the waveforms for the modulations.
Debugged the speakers and got audio output.

Week 8

(12/1 - 12/5) [Demo week]
☐ Get demo ready

Finished sodering.
Combined the Arduinos into the final product.

Problems Encountered

Clock frequency

Problem:

- The Raspberry Pi did not have a sample rate that would work with our project without significant delay.
- Since our development is just a prototype, it is not expected to run perfectly, but the Raspberry Pi will not work with the ultrasonic sensors to create a musical instrument.

Solution:

- Swapped out the Raspberry Pi for an Arduino Uno. Higher clock speed, thus, a higher sampling rate. Have not encountered a problem with this.

Output volume

Problem:

- Speakers with the low current input from the Arduino, produces a tone at a very low volume

Solution:

- Attach a buffer with current supplied by a battery pack.

Sine wave implementation

Problem:

- Using the DAC with the Arduino Uno was not producing the desired sine wave Old sine wave

Solution:

- Used analog discovery to test where the problem was and fixed the connections. Now producing a decent sine wave that works with our speaker.

Final Design

The user adjusts the distance their hands are from the device. The ultrasonic sensors take the distances of the user's hands and passes it to the first Arduino. This arduino translates the input into a signal that will tell the other arduino several things: the note, the octave, and the modulation. The modulation is passed through 3 digital signals, and the note/octave is passed through a DAC to be turned into an analog signal.

The second arduino takes these inputs, selects the correct modulation formula, and calculates the sampled waveform. This waveform gets passed to a second DAC, through a buffer, and into the speaker.

Sensing Arduino

The "first" Arduino is what does most of the calculation and "work" for our project. It constantly reads data from the ultrasonic sensors in an alternating manner with both sensors ultimately being read twice per second. It uses the distances that it obtains from the sensors to create a value between 0 and 63, which will go on to represent what frequency we will create.
The other data that this Ardino must report is which button was last pressed. In order to do this, we created a voltage dividing circuit that connects the buttons to an input pin to read which button is currently selected. In order to maintain the integrity of the selection, we used a simple loop to wait for a steady state of voltage (so as to ignore the voltage "bounce").

Continuously while the Arduino takes input, it sets the "note" by setting a voltage proportional to the note's value on the DAC (i.e. 0 V for note 0, up to 5 V for note 63). It also sets a digital set of pins to high or low to binary encode which button was pushed (000 for button 1, through 111 for button 8). Both of these values will later be read by the Synthesizing Arduino.

Synthesysing Arduino

This Arduinoo is what actually makes the sound for the speaker. It takes the data fed to it from the sensing Arduino via an analogue pin and three digital pins. It then turns this data into the note frequency and "instrument" respectively. It then goes to a lookup table and selects the "instrument" and then generates the waveform that it is given from the table at the DAC according to the frequency that was determined. The output of the DAC then goes to a MOSFET emitter follower powered by a 12 V battery array in order to supply enough power to actually drive the speaker.

Other considerations

Although the project works properly, there are several thing that still could use improvement.
One of the largest things that could use improvement is how we are actually creating the waveforms. We are currently using what ammounts to 5-bit representations of the sound waves we are trying to mimic. The reason we chose to do this rather than use a higher resolution was that we were uncertain how stable the output signals would be if we attempted to use the higher resolutions as our controler is rather limited in its capibilities. If we had more time, we could have attempted to use higher resolutions emulations of the sound waves, which would make them more distinct and cleaner.

Another major factor that could likely be improved is how we are passing the signal between our Arduinos. Our final prototype has our sensing Arduino use a DAC to set a voltage representing the note to be played by our synthesizing Arduino. Because our two Arduinos do not have a common ground reference, the voltage set high by one could be misinterpreted by the other resulting in some of the errors in output. The main reason that we elected to transfer data in this method rather than something like PWM was that our synthesizing

Arduino could not take the time to wait for a signal that could have at least 36 distinct values as it had to constantly produce the waveforms.

The last point of optimization I would like to bring up concerns the ultrasonic sensors. They work most of the time, and were capible of a fairly precise measurement when we were originally testing them. Unfortunately when we started actually using the speaker, it caused excess vibrations that we did not really account for. As such, occasionally, the ultrasonic sensors report errorous data. While some of this can be mitigated by ignoring distance values in excess of ~70 cm, values within range cannot be ignored. One might suggest that we could simply force the ultrasonic sensors to report the same value multiple times in a row in order to be considered valid, but this slows down the note transition process when we are attempting to play the Thermin, and as such we elected to deal with the "squeaks" that our device made, much as people tollerate the "squeaks" of middle and high school woodwinds.

Demo

YouTube

Team responsibilities

Nick: Software development, prototype testing, debugging

Caleb: Hardware/Casing, Prototype design

Justin: Prototype documentation, testing, and debugging

Expense Report

Caleb:

Purchase 1

Wood support for inside the prototype box	17.88
Subtotal	**17.88**
Tax	**1.48**
Total (USD)	**19.36**

Purchase 2

T Nut for the dowel rod	1.18
2 - Hex nuts for securing dowel @1.20 ea	2.40
2 - Treaded rod @ 2.42 ea	4.84
Copper wire	5.48
Velcro for securing internal components	2.93
Subtotal	**16.83**
Tax	**1.39**
Total (USD)	**18.22**

Purchase 3

Screws	4.98
Foam insulation	1.53
Subtotal	**6.51**
Tax	**0.54**
Total (USD)	**7.05**

Nick:

Purchase1

Speaker	1.95

Shipping	7.99
Subtotal	**9.94**
Tax	**0.82**
Total (USD)	**10.76**

Retrieved from "https://wiki-class.cse.tamu.edu/index.php?title=Theremin%2B&oldid=21469"