SCHOOL OF TECHNOLOGY AND PUBLIC MANAGEMENT

ENGINEERING TECHNOLOGY DEPARTMENT

Course ENGT 3260 Microcontrollers

Summer III 2015

Instructor: Dr. Maged Mikhail

Project Report

Submitted By: Nicole Kirch

7/10/2015

Table of Contents

Problem Statement

The goal of this project is to create a Theremin using the Arduino Uno microcontroller. A Theremin is an electronic device that creates sound when the musician waves their hand in front of the circuit. Traditionally the circuitry uses conductive antennas as capacitors to send a voltage to oscillators that create the audio signal. Instead of sticking with only the traditional method of sound creation, I will explore two other options that use the waving of the hand to change the signal frequency and use the Arduino to turn that into an audio signal.

Theory

Microcontrollers are programmable chips that have central processing units (CPU), input and output (I/O) ports, timers, serial communication and memory. In this project I will be using the Arduino Uno. It has analog and digital ports, can power circuits up to 5V (but needs at least a 9V or USB connection to be powered), and uses Universal Serial Bus (USB) for its serial communication connection. The Uno is programmable with the C programming language.

Project Description

In order to convert hand movement into an electronic signal I need sensors. In the first approach I will use a photocell to turn light intensity into voltage. The closer hands are to the photocell, the lower the light reaching the sensor and lower the voltage. This voltage is sent to the Arduino to turn into a certain frequency audio signal. For the second approach I will use a proximity sensor to sense the distance of the hand to the sensor and this information is translated through the circuit with a digital potentiometer setting pitch value and the 555 timer as the oscillator to create the audio signal.

Steps

Light Theremin

Set up the Circuit

The focus of this circuit is to run and read the photocell. The photocell gets its required 5V from the 5V pin on the Arduino and sends its analog reading back to the Arduino. After the Arduino processes it, it sends it to the speaker from a digital write pin (one with pulse width modulation).

Create the code

Initially when I saw the tone() method in the source code, I thought that it was part of the library named Tone. When I imported the library I had many errors relating to commands that I had not created or initialized. Even after trying different methods, the error messages stayed. I went back to the minimalist source code and removed the added library and it worked.

While this code is simple, it is deceptively so, as it interprets the photocell signal and creates the audio signal. It reads the analog light intensity and converts it to a digital signal that through the tone() method is turned into a square wave that is sent to the speaker. The integer that is read is manipulated through a simple algorithm to make sure that it is greater than 200,

because the speaker has a hard time reproducing audio lower than that value in Hertz and human ears have a hard time picking up on those frequencies.

Troubleshooting

When the debugged code was uploaded to the Arduino the pitch produced was in a lower octave (pitch range). The range of notes is fairly limited due to the size and sensitivity of the photocell. The size of the photocell also means that it can have inaccuracies if it doesn't have the perfect lighting set up. Also the sound produced was too fluid, which made it hard to hear the transition between pitches. Initially the algorithm to translate the voltage to pitch was [200 + pitch/2], this produces a frequency no lower than 200 Hz. Changing the divisor to 1 produces a higher octave. The longer the duration of the note causes a smoother transition between notes, but if the duration is shorter the step-like transition between notes actually gives it a more musical quality. This quality is reminiscent of 8-bit video game music.

Proximity Theremin

Set up the Circuit

All of the sound creation in this Theremin is in the circuitry of the 555 timer. What determine the pitch are the values that are fed to it from the ultrasonic proximity sensor via the digital potentiometer. Using a Trigger pin and an Echo pin, the sensor sends out a short high frequency "ping" and when it hits something it bounces back and gets recorded. Trigger and Echo are tied to digital read/write pins on the Arduino. The length of time between sending and receiving the signal correlates to distance based on the speed of sound (343 m/sec). Next this distance is turned into a pitch that the Arduino writes to the timer. The digital potentiometer reads the integer with digital write from the Arduino and controls the timer based on this integer's value. When enabled, the timer becomes an oscillator that feeds an audio signal to the speaker. To know when it is enabled the timer connects to an Arduino digital pin. From this pin a Boolean (HIGH or LOW) is sent via the digital potentiometer.

Create the Code

Because the audio signal is created by the circuit, the code focuses on translating the information gathered by the proximity sensor. The SPI library is imported to interface with the digital potentiometer and the New Ping library is imported to create a sonar (ultrasonic) ping that the proximity sensor turns the time delay into an integer of distance in centimeters. Once the distance is read, it is evaluated on whether or not it is in range to give a correct value for pitch control. If it is outside the range determined by the code, then the timer is disabled. Otherwise the timer is enabled and ready for oscillation with the given integer (translated via ((distance - 10) * 3) + 1 algorithm).

Troubleshooting

This Theremin worked from the first upload without any extra tweaking of the code. It has a limited range that it senses for pitch changing, and the code has made the circuit not function outside of this range. Another issue is that the amplitude of the audio signal is low.

To change the amplitude I replaced the 10k ohm resistor going from the 5V pin to pin 7 on the timer to be 1k ohm which will create a smaller voltage drop across the resistor and allow

more voltage to reach the audio signal. This only slightly changed the pitch of the audio signal and did not change the range sensitivity or the amplitude.

Materials Used

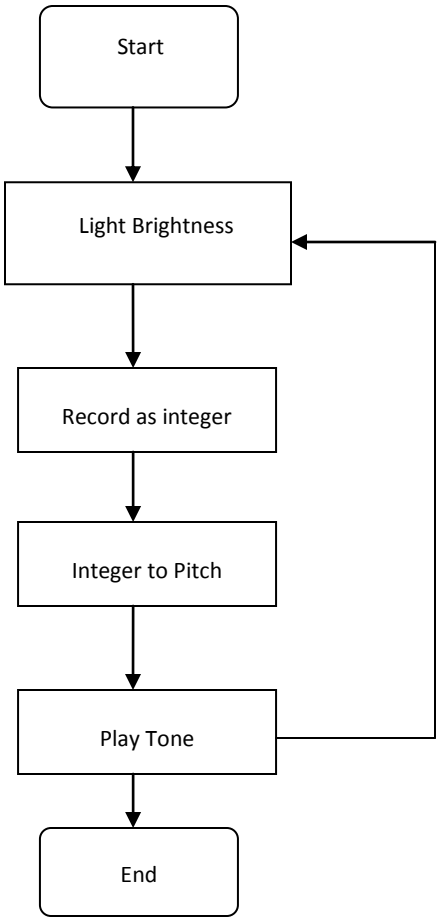
Light Theremin
1k ohm resistor
Photocell
Arduino Uno
8 ohm Speaker
9V (optional)

Proximity Theremin
10k ohm resistor * 2
1k ohm resistor (for troubleshooting)
100k ohm resistor *1
10 micro farad capacitor * 2
4.7 n farad capacitor
555 Timer
MCP4131 Digital Potentiometer
HC-SR04 Ultrasonic Range Sensor Proximity Sensor
Arduino Uno
8 ohm Speaker
9V (optional)

Implementation

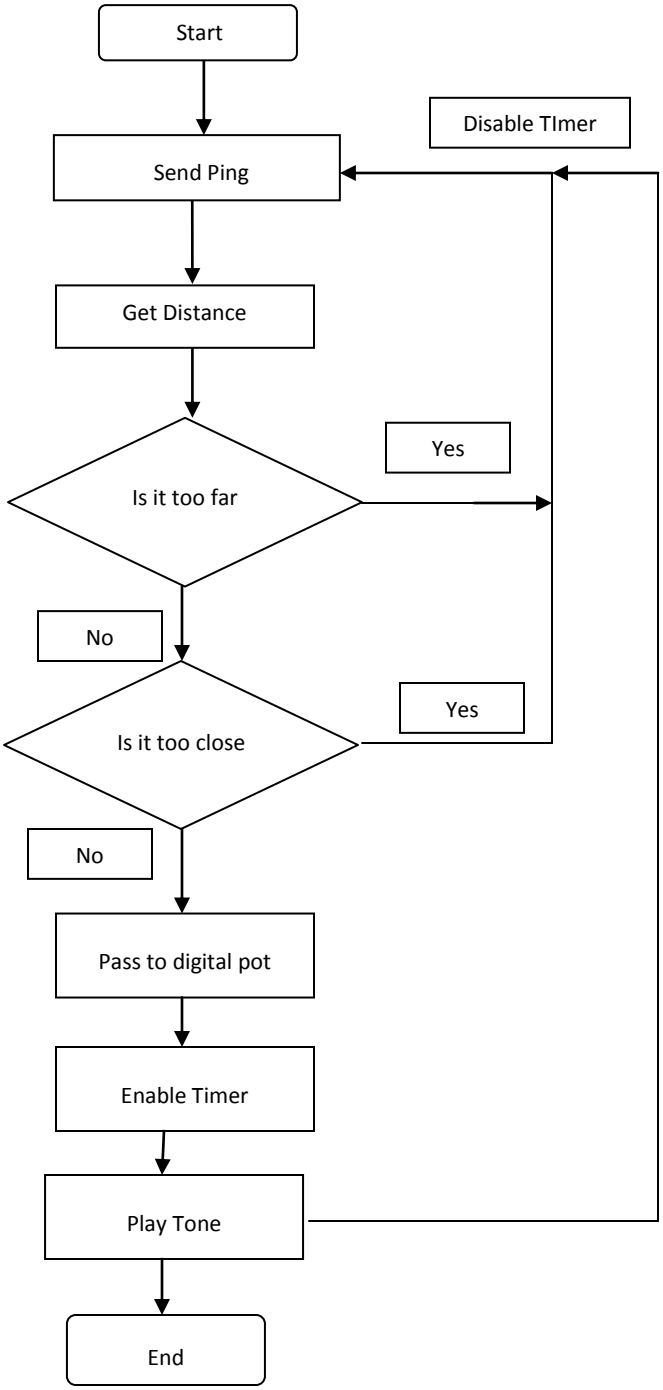
Simple Theremin

The circuit of the light sensing Theremin only requires one resistor and the photocell in addition to the Arduino, speaker and power source.



Proximity Theremin

The Timer requires a certain set-up in order to act like an oscillator which requires 3 capacitors and 2 resistors. The digital potentiometer requires a resistor to connect to the timer. All integrated circuits used connect to Vcc, Ground and digital pins (some pulse width modulator enabled) from the Arduino.



Code

Simple Theremin

```
//initializes the digital write pin for the speaker
   int speakerPin = 11;
   //initializes the analog read pin for the photocell
   int photocellPin = 0;
   //duration of notes if there is too much bleed
   int duration = 500;
   int reading = 0;
   int pitch = 0;
        void setup()
   {
     pinMode(speakerPin, OUTPUT);
   }
       void loop()
   {
     //reads the intensity of the light and generates an integer
   reading = analogRead(photocellPin);
   //sets the value of the pitch based on the photocell integer
   pitch = 200 + reading/1;
   //generates the tone from the pitch value and sends it to the speaker
   tone(speakerPin,pitch,duration);
  delay(duration);
      }
```

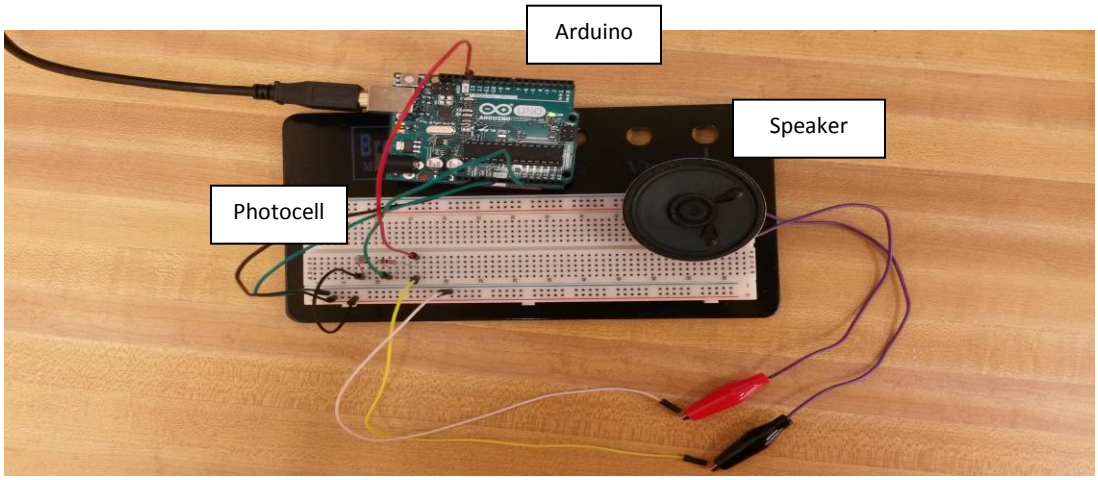
Proximity Theremin

```
//Add these libraries to the code
//SPI to control the digital potentiometer
#include <SPI.h>
//create a ping to find distance through the ultrasonic sensor
#include <NewPing.h>
// set pin 10 as the slave select for the digital pot
const int slaveSelectPin = 10;
// set pin 12 as the reset pin for the 555 timer
//this sets distance range
const int timerEnablePin = 12;
// Arduino pin tied to trigger pin on the ultrasonic sensor
#define TRIGGER_PIN  3
// Arduino pin tied to echo pin on the ultrasonic sensor
#define ECHO_PIN     2
// Maximum distance we want to ping for (in centimeters). Check device datasheet.
#define MAX_DISTANCE 4000
// NewPing setup of pins and maximum distance.
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
int level = 0;
int distance = 0;
void setup()
{
   // set the slaveSelectPin as an output:
   pinMode (slaveSelectPin, OUTPUT);
   // set the 555 timer enable pin to output
   pinMode (timerEnablePin, OUTPUT);
```

```
   // disable the 555 timer
   digitalWrite(timerEnablePin, LOW);
   // initialize SPI:
   SPI.begin();
   // Open serial monitor at 115200 baud to see ping results.
   Serial.begin(115200);
}
void loop()
{
   // Send ping, get ping time in microseconds (uS).
   unsigned int uS = sonar.ping();
   // Convert ping time to distance
   distance = uS / US_ROUNDTRIP_CM;
   // Print distance to serial console
   Serial.print("Ping: ");
   Serial.print(uS / US_ROUNDTRIP_CM);
   Serial.println("cm");
   if (distance < 11)
   {
      // too close - disable the 555 timer - no sound
      digitalWrite(timerEnablePin, LOW);
   }
   else if (distance > 52)
   {
      // too far - disable the 555 timer - no sound
      digitalWrite(timerEnablePin, LOW);
   }
   else
   {
      // set pot level to a value proportional to distance
      //pitch algorithm
      level = int ((distance - 10) * 3) + 1;
      digitalPotWrite(level);
      // enable the 555 timer
      //play the pitch
      digitalWrite(timerEnablePin, HIGH);
   }
   //delay between pings in milliseconds
   delay(100);
}
// SPI transation to set the value of the digital pot
//this sets pitch value
int digitalPotWrite(int value)
{
  //take the SS pin low to select the chip:
  digitalWrite(slaveSelectPin, LOW);
  //send in the address and value via SPI:
  SPI.transfer(0);
  SPI.transfer(value);
  // take the SS pin high to de-select the chip:
  digitalWrite(slaveSelectPin, HIGH);
}
```

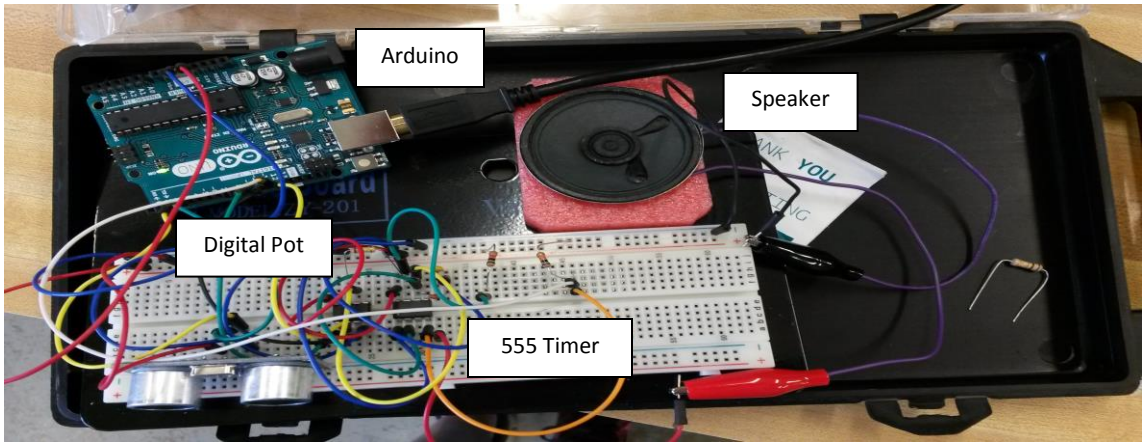
Results

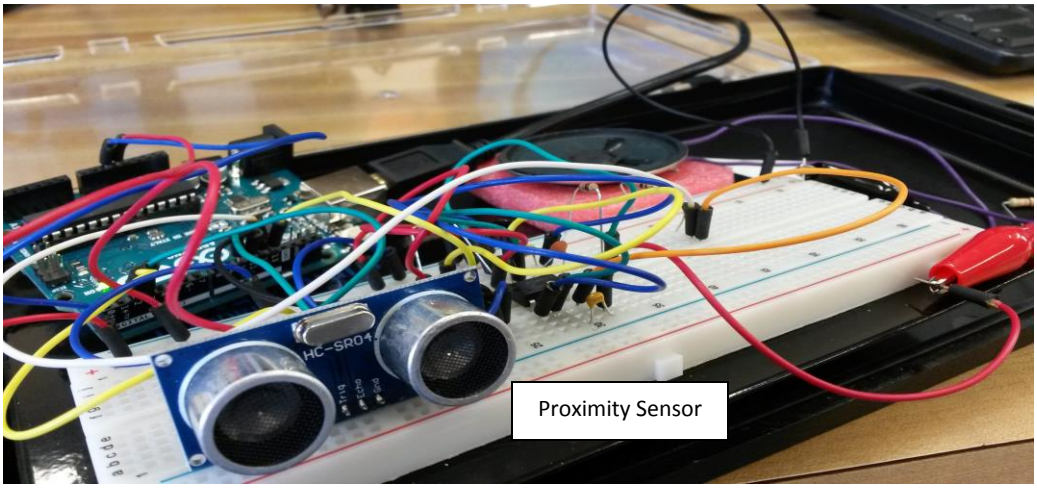
Simple Theremin



This set-up worked similarly to what I expected. Changing the intensity of the light to the photocell changed the pitch and the pitch range was limited. The location of this limited bandwidth was also able to change by changing the code. If I were to go more in depth with this project then I would add a potentiometer to change the frequencies that are in range in real time to couple with the photocell manipulation. The feature that I did not expect was the duration of the pitch which could control how the pitch changes, either abrupt or smooth.

Proximity Theremin





I expected this to be the better Theremin, with more distance range and better tone quality. But with not knowing too much of how the Ultrasonic Range Finder worked, it ended up that this was the weaker procedure. The sound produced from this set-up had a lower amplitude tone and a very small range to pick up pitch in comparison to the light Theremin. The pitch algorithm is controlled through the code, but is determined by the manufacturer specifications of the sensor, which is something I did not do enough research with. A larger surface area object works better with this sensor than a hand, so I used a piece of paper instead of my hand to wave in front of the sensor. As the object reaches the edge of the sensing range, the sound starts to sound like pings instead of a continuous tone. This is the timer being enabled and disabled.

References

1. Monk, Simon. "Lesson 10 Pseudo Theramin." *adafruit.com*, published, last accessed https://learn.adafruit.com/adafruit-arduino-lesson-10-making-sounds/arduino-code

2. Thompson, John. "Skill Builder: Advanced Arduino Sound Synthesis." *Make:*, last accessed July 13, 2015 http://makezine.com/projects/make-35/advanced-arduino-sound-synthesis/

3. Jackson, Jamie. "Arduino Theremin." *blog.jacobean.net*, published January 5, 2013. last accessed July 13, 2015 http://blog.jacobean.net/?p=766

4. psychephylax . "NewPing Library." *Arduino.cc*, last modified November 4, 2013. Last accessed July 13, 2015 http://playground.arduino.cc/Code/NewPing

5. Glinsky, Albert. "Theremin: Ether Music and Espionage." Board of Trustees of the University of Illinois, 2000