Therepong: A Two-Player Pong Game Controlled with Capacitive Displacement Sensors

Sudvarg, Marion Bailey, Ricker

December 12, 2012

Abstract

Here we describe a novel and entertaining way of using capacitive displacement sensors. Using similar methods to the creation of a theremin, we set up two sensor plates. Instead of using these to create and modify sound, however, we use them to control the position of two paddles in a version of the popular computer game "Pong". We add further controls to the game with a selector switch, push-button switch, and two potentiometer knobs.

Introduction

Earlier in our class, we learned how to make a Theremin using capacitive displacement sensing techniques. In essence, a metal plate forms half a capacitor, with a person's hand forming the other side. Moving the hand over the plate varies the capacitance of this system. This change in position varies the sound output by the Theremin. Instead of using this method to modify sound, we can instead use this method to directly measure distance from the hand to the plate.

See Appendix I, Figure 1 for a circuit diagram of such a capacitive displacement sensor. The first part of the circuit is a relaxation oscillator, which varies in frequency with the variable capacitor, according to the following formula:

$$f = \frac{1}{2ln(3) * R1 * C1}$$

Varying frequency also allows for voltage output variation with the second part of the circuit, an RC filter, so we get a voltage-varied signal that can be read by a microprocessor.

Experimental Method

In our initial design, we used two such capacitive displacement sensors to plot position in two axes. When the sensors were placed apart from each other, they both worked satisfactorily. However, when together perpendicular to each other, we got unsatisfactory results. They did not operate independently. Since the object they were sensing (a and or arm) was not insulated from itself in

two dimensions, charge was able to flow around the object, which caused movement in one direction to show as movement in both axes.

We did some research to find a solution to this problem. We found a paper from 1994 about a new method of two-dimensional position analysis.[1] This suggested using a two-dimensional grid of capacitors. However, upon further study, we found this method works only at a relatively small scale (past 1mm, error increases drastically), and so for accurate measurement at our scale we would need a huge number of very small capacitor plates, which was unfeasible. Indeed, even in a single dimension, capacitive displacement sensors are meant to work precisely at small distances. One company selling sensors advertises high range sensors that are active to 10mm.[2]

So, with these limitations, we decided to have our two sensor plates placed apart from each other. But plotting independent positions has been done before, so we needed a novel use for our system. We decided to have each of our plates control the position of a paddle in a computer game of ping pong.

The output from our two sensors go to two inputs of an Arduino Uno microcontroller. We also have three more controls. One potentiometer controls game color (paddles, ball, and score text). Another controls ball speed. A push-button switch and selector switch in parallel force the game to reset (the selector switch can hold the game in start position, or when it is switched to run the game, the pushbutton can quickly reset the game). This setup is diagrammed in Figure 2 of Appendix I. Note that Sensor 1 controls the top paddle and Sensor 2 controls the bottom paddle.

The following table gives resistor and capacitor values for the two sensors, based on the labels in Figure 1 of Appendix I.

Sensor	1	2
R1	.971MΩ	.998MΩ
R2	9.89kΩ	9.89kΩ
R3	9.86kΩ	9.86kΩ
R4	9.85kΩ	9.89kΩ
C2	2.10μF	2.09μF

Full code for the Arduino microcontroller is provided as Appendix II. Processing code controlling the ping pong game is provided as Appendix III.

Results

Getting the ping pong paddles to move all the way across the screen required tuning of the signal analyzer built into our Processing code.

We took measurements of the output signals of the two sensors with a hand at various distances from them:

Sensor 1	Frequency (kHz)	Voltage (V)
Hand Away	34.5	2.20
Hand Touching Plate	3.45	1.43
Hand Hovering 1-2 cm	21.5	1.75
Sensor 2		
Hand Away	25.9	1.77
Hand Touching Plate	3.45	1.20
Hand Hovering 1-2 cm	19	1.49

The analog inputs of our microcontroller map a voltage value from 0-5V to a 10 bit value (0 to 1023). So in our processing code, we needed to map the lowest microcontroller value to 0, and the highest microcontroller value to 200 (the width of the game screen). To find these values, we use the following formula, where A is the analog output value and V is the measured voltage.

$$A = \frac{1023 * V}{5}$$

Using this formula, we get the following table of values, noting that the lowest values are where the hand touches the plate and the highest values are where the hand is entirely away from the plate.

Measured Value	V	A
Sensor 1 Lowest Value	1.43	293
Sensor 1 Highest Value	2.20	450
Sensor 2 Lowest Value	1.20	246
Sensor 2 Highest Value	1.77	362

So, in theory, we want to map the top paddle position from 246 to 362 and the bottom paddle position from 293 to 450. These values provide good starting points, but actual experimentation with the game made us adjust these values to a range of 250 to 450 for the top paddle and 250-370 for the bottom paddle. These are very close to the expected values except for the lowest value of the bottom paddle.

Discussion and Conclusions

This project led us to interesting conclusions about capacitive measuring techniques. Clearly, capacitive measurement can have a wide range of uses, but typically not a wide range of distances. Using very small sensors, we could have created a two-axis measuring system, but this would have been beyond the scope of the time and materials we had to work with.

But using two capacitive measuring devices, much like Theremins, to control a video game is an approach that is novel and fun. We had to do some work tuning our program to place the paddles in the correct ranges, but the ranges ended up being surprisingly close to what we expected. Perhaps, in the future, we can study ways to make the outputs more consistent (instead of seeing jiggling in the paddle positions).

Acknowledgments

I would like to thank Ricker Bailey for his help with this project; indeed, for working with me the entire year. He is very skilled with circuits, and his knowledge has helped me plenty of times. His contributions to this project were immense.

I would also like to thank James Buckley for teaching Physics 321, Electronics Lab. It has been one of the most enjoyable classes I have taken at Wash U, and one of the very few that I felt true sadness at its ending. The TA's, Jeff Pobst and Nathan Todd, were huge assets as well.

References

[1]Bonse, M.H.W, Zhu, F., & Spronck, J.W. "A new two-dimensional capacitive position transducer." *Sensors and Actuators* A, 41-42 (1994). p 29-32

[2]Lion Precision. Products Page. Accessed 29 Nov 2012. http://www.lionprecision.com/products.html

Appendix I Figures

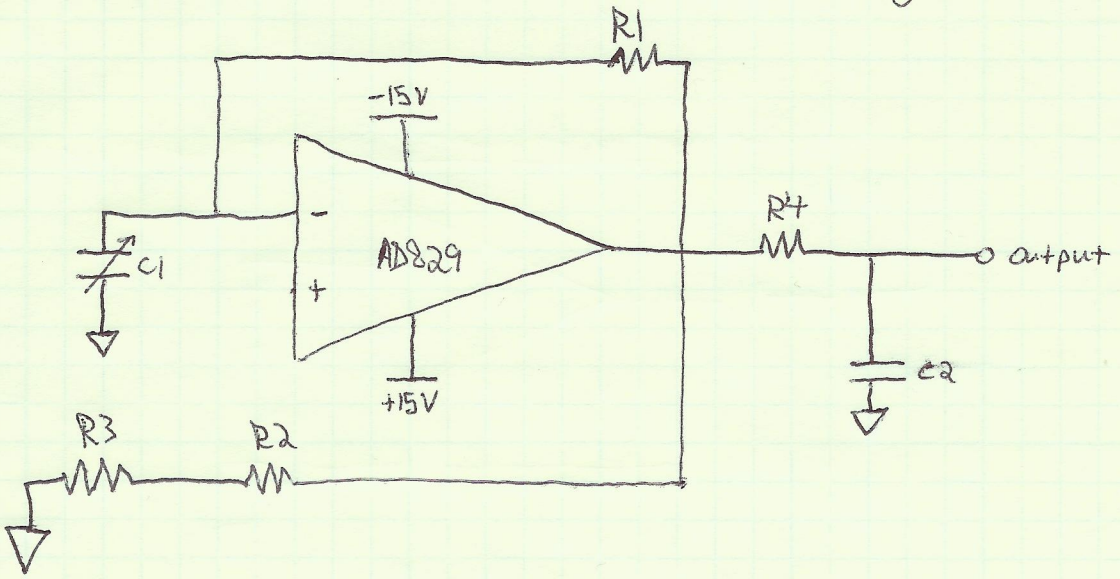

Figure 1
Capacitive Displacement Sensor Diagram

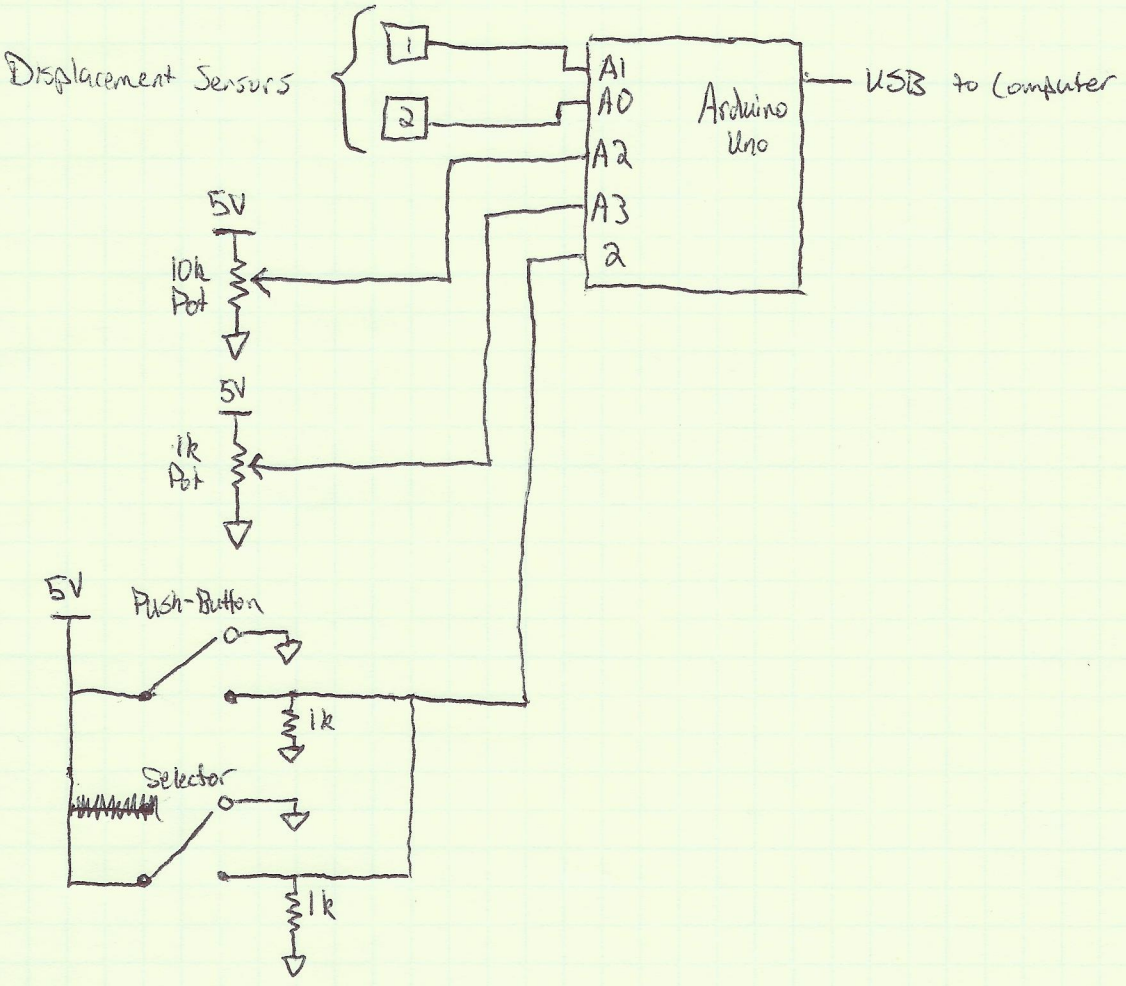

Figure 2
Hardware Setup

Appendix II Arduino Code

```
//This allows the Arduino to read five inputs from our circuit and print
//their values to a serial line input to the computer. A Processing sketch
//then uses this information to control a ping pong game.

#include <math.h> //Include library of necessary math functions

//Define Arduino pins
int sensorPinY = A0; //A0 receives the output value from our second Theremin
                     //setup
int sensorPinX = A1; //A1 receives the output value from our first Theremin
                     //setup
int colorPin = A2; //A2 receives a voltage value from the potentiometer
                   //controlling color
int saturPin = A3; //A3 receives a voltage value from the potentiometer
                   //controlling speed. Note that initially this
                  //potentiometer controlled color saturation but
                   //when it became useful to make it control speed, we
                   //neglected to change variable name
int drawPin = 2; //Digital input 2 receives the output value from our two
                 //switches that reset the game

//Initialize string for the variable that passes information about whether or
//not to reset the game based on readout from digital input 2
String isDraw = "";

//Setup serial connection at 9600 baud
void setup()
{
  Serial.begin(9600);
}

void loop(){
  if (digitalRead(drawPin) == HIGH) {
    isDraw = String("1");
  }
  else {
    if (digitalRead(drawPin) == LOW) {
      isDraw = String("0");
    }
  }

  //Initialize positions of each paddle. As stated in the Processing sketch,
  //we initially used each Theremin setup to control position on two axes.
  //When we changed to two paddles we did not change variable names
  long xcoord = analogRead(sensorPinX);
  long ycoord = analogRead(sensorPinY);

  //Print input values as a string of comma separated values
  Serial.println(xcoord + String(",") + ycoord + String(",") + isDraw +
String(",") + analogRead(colorPin) + String(",") + analogRead(saturPin));

  //Delay loop so game runs in 100ms (0.1s) iterations
  delay(100);
}
```

Appendix III Processing Code

```
//Initialize Variables

import processing.serial.*; //Import serial port libraries for Processing
Serial myPort; //Initialize variable myPort of type Serial

//Initialize float values for ball speed in two axes
float xSpeed;
float ySpeed;

//Initialize integer values for ball position in two axes
int ballX;
int ballY;

float xRat; //Creates a value for a ratio of xSpeed to ySpeed

//A function to reset the ball after a player scores
void ballInit() {
  //Speed in both axes is randomly assigned to be positive or negative so the
  //ball starts moving in a random direction
  ySpeed = round(random(0,1));
  if (ySpeed==0) { ySpeed = -1; }
  xSpeed = round(random(0,1));
  if (xSpeed==0) { xSpeed = -1; }
  xRat = random(0,1)*2;

  //Defines initial ball position
  ballX = 100;
  ballY = 300;

  //Defines initial scores
  int player1Score = 0;
  int player2Score = 0;
}

void setup () {
  //Set window size and drawing mode
  size(200, 600);
  ellipseMode(CENTER);
  rectMode(CENTER);
  colorMode(HSB,360,100,100);

  //Lists available serial ports. This is commented out for actual use, but
  //when first running the game, we may need this to change the port we are
  //reading
  //println(Serial.list());
  //Typically the first serial port will be the Arduino, so we open
  //Serial.list()[0].
  //Change to Serial.list()[i] where i is the Arduino port if necessary
  myPort = new Serial(this, Serial.list()[0], 9600);

  //Buffer serial input to newline, then run serialEvent
  myPort.bufferUntil('\n');
```

```
  //Set initial background to white and initialize ball
  background(0,0,100);
  ballInit();
}

void draw () {
//Everything happens in the serialEvent()
}


void serialEvent (Serial myPort) {
  // Get a single line as and ASCII string from serial input
  String inString = myPort.readStringUntil('\n');

  if (inString != null) {
    //Trim off any whitespace and split CSV
    inString = trim(inString);
    String[] inStringSplit = split(inString,',');

    //Convert first 2 values to integers and map the typical range of values
    //for each Theremin output to the screen width. Note that we use xcoord
    //and ycoord for the x-axis position of the two different paddles. When
    //we initially worked with having each Theremin plate control a different
    //axis, we defined these variables as xcoord and ycoord. When we switched
    //to control of two paddles moving on the same axis, we kept the same
    //variable names
    float xcoord = float(inStringSplit[0]);
    xcoord = map(xcoord, 250, 450, 0, 200);
    float ycoord = float(inStringSplit[1]);
    ycoord = map(ycoord, 250, 370, 0, 200);
    //Convert third value to an integer. If this is a 1 the game is reset
    //(code for this farther down)
    float tracer = float(inStringSplit[2]);
    //We convert the fourth value to a float value and use this to determine
    //game color
    float hueval = float(inStringSplit[3]);
    hueval = map(hueval, 0, 1023, 0, 360);
    //We convert the fifth value to a float value and map it to ball speed,
    //from 2 to 20
    float speedval = float(inStringSplit[4]);
    speedval = map(speedval, 0, 1023, 2, 20);

    //Brightness is set at 100
    float brightval = 100;

    //Speeds are now set as speedval (set by serial input) with initial
    //direction maintained. xRat is used to scale xSpeed
    //to a ratio of ySpeed so our ball is not always traveling at a 45 degree
    //angle
    ySpeed = int(speedval) * ySpeed/abs(ySpeed);
    xSpeed = int(speedval) * xSpeed/abs(xSpeed) * xRat;

    //If the ball strikes the left or right edge of the screen, it bounces
    //off
    if (ballX<=(0+abs(xSpeed)/2)) {xSpeed = -xSpeed;}
    if (ballX>=(200-abs(xSpeed)/2) ) {xSpeed = -xSpeed;}
```

```
    //If the ball strikes either paddle, it bounces off. We have a 2 pixel
    //buffer on each side of the paddles
    if ( ballY<=(500+abs(ySpeed)/2) && ballY>(500-abs(ySpeed)/2) &&
(abs(ballX - ycoord) <= 27) ) {ySpeed = -ySpeed;}
    if ( ballY<=(100+abs(ySpeed)/2) && ballY>(100-abs(ySpeed)/2) &&
(abs(ballX - xcoord) <= 27) ) {ySpeed = -ySpeed;}

    //If the ball goes past the top or bottom edge of the screen, a player
    //scores and the ball is reset
    if ( ballY>=0 && ballY<abs(ySpeed) ) { player1Score++; ballInit(); }
    if ( ballY<=600 && ballY>abs(600-ySpeed) ) { player2Score++; ballInit();
}

    //Clear drawing to allow for new positions to be drawn
    background(0,0,100);

    //Code to reset the game
    if(tracer == 1) {
      player1Score = 0;
      player2Score = 0;
      ballInit();
    }

      //Draw game on canvas

      //Colors are set based on hueval
      fill(hueval,100,brightval);
      stroke(hueval,100,brightval);

      //Draw paddles based on input from Theremin plates. Paddles get thicker
      //with higher ball speed
      rect(xcoord,100,50,abs(ySpeed));
      rect(ycoord,500,50,abs(ySpeed));

      ellipse(ballX,ballY,6,6); //Draw the ball

      //Draw the scores
      String p1 = "Player 1 Score: " + player2Score;
      String p2 = "Player 2 Score: " + player1Score;
      text(p1,5,20);
      text(p2,5,580);

      //Increment ball position based on speed
      ballX += xSpeed;
      ballY += ySpeed;

  }
}
```