Final Project Report
E3390 Electronic Circuits Design Lab
The Digital Harpsichord

Cesar Aguilar
Andrew Sabatino

Submitted in partial fulfillment of the requirements for the
Bachelor of Science Degree

Dec 16, 2006

Department of Electrical Engineering
Columbia University

Table of Contents

1. Executive Summary
2. Block Diagram, Design Targets, and Specifications
3. Individual Block Descriptions: Schematics
4. Bill of Materials
5. Health, Safety, & Environmental Issues
6. Final Gantt Chart
7. Criticism of This Course

Appendices –

1. Code
2. Sample Input Song: “What Child is This?”
3. Sample Processed Song

1. Executive Summary

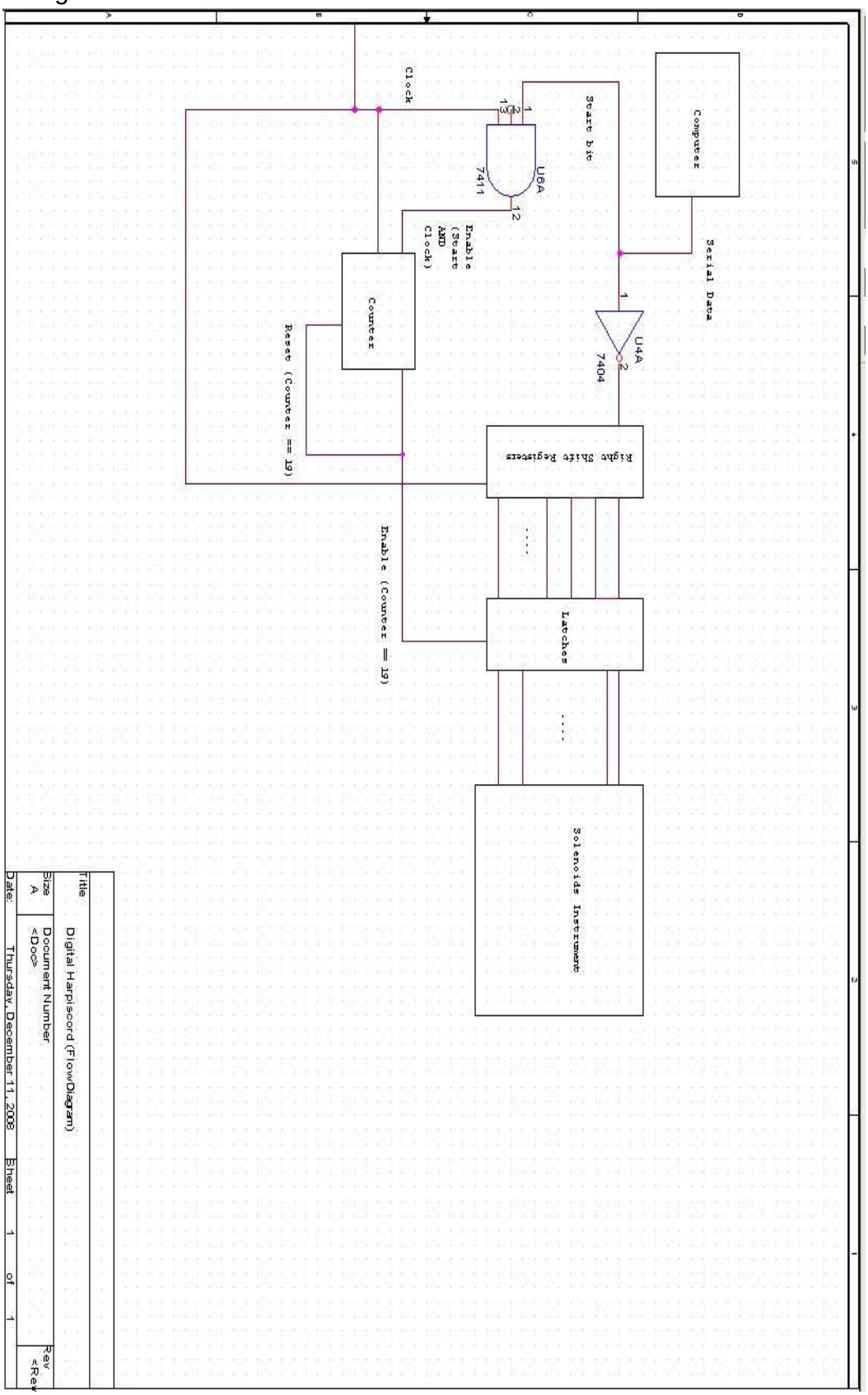
The design presented is for the construction of a unique, harpsichord-like instrument controlled by computer input software and the requisite control circuitry. In the tradition of keyed harpsichord, the design uses harpsichord jacks to pluck strings tensioned between tuning pegs and a bridge. The harpsichord jacks have a sprung tongue which holds a plectrum. The plectrum rests in a position below the string and plucks the string, producing a sound when forced upward. When the plectrum is restored to its initial position, the tongue hinges backward past the string, preventing a second plucking sound when the jack is reset. The strings used are steel-wound silk and nylon, ball end classical guitar strings.

Each jack is attached to a push-type solenoid which converts an applied voltage to the mechanical push necessary to drive the harpsichord jack. There are thirteen solenoids and thirteen strings, allowing for a range of one full chromatic octave with one additional note. The strings are tuned such that the lowest and highest notes are a C natural.

The thirteen solenoids are controlled by digital circuitry that translates parallel output serial data from a PC into 13 bits of parallel data which are refreshed at a rate of 192 Hz. The serial data on the PC is generated by unique software which takes text file input that specifies a song. The text input determines the duration of each note and the required pitch. The 192Hz refresh rate allows for the creation of notes as fast as 64^{th} notes in simple or complex meters in a tempo of 60 bpm.

2. Block Diagram, Design Targets, and Specifications

Block Diagram



Specifications and Design Targets:

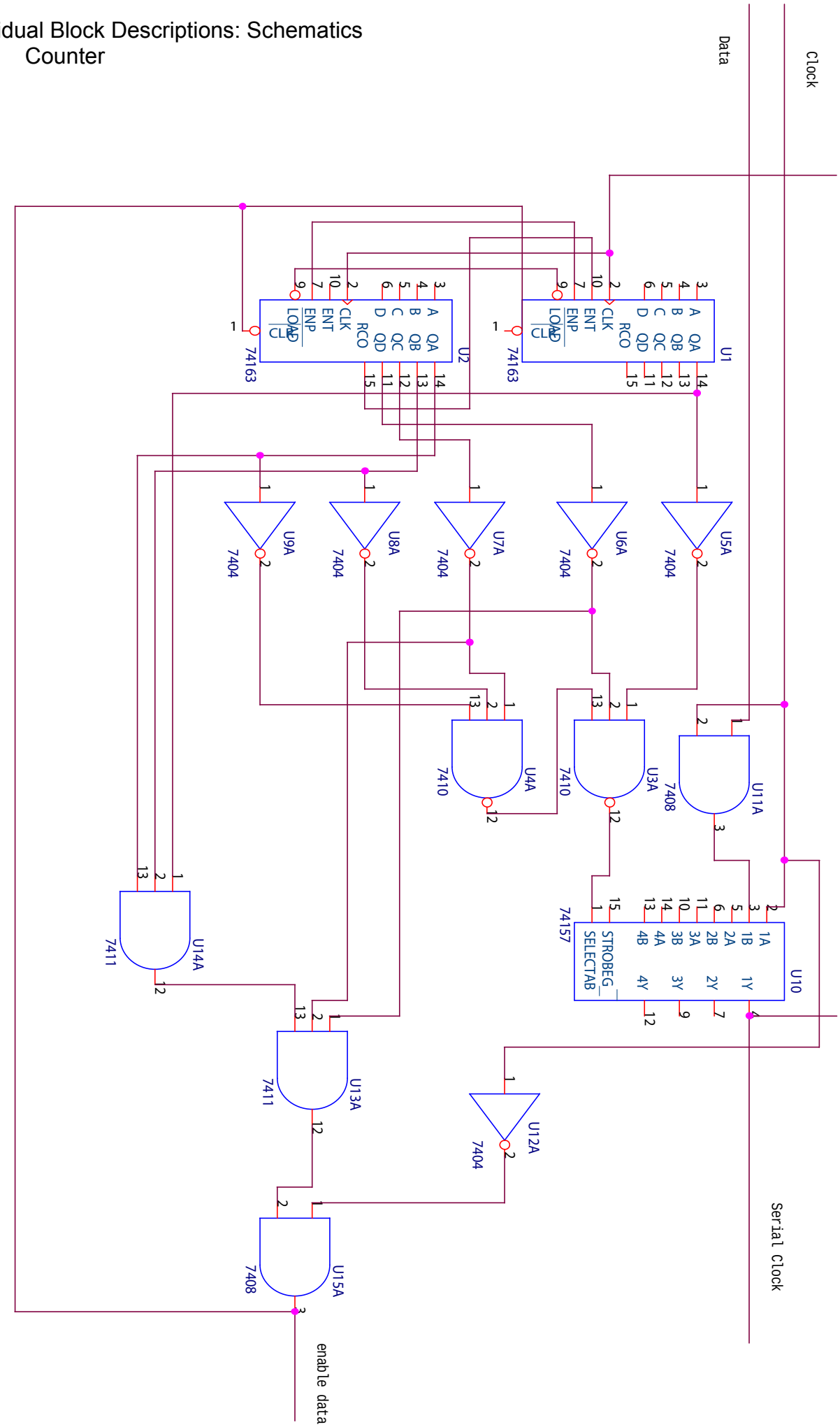
Inputs:
18V, 2 A
5V, 800 mA
192 Hz square wave, 5 V p-p, 2.5 V offset
Note that the current drawn by each solenoid is approximately 480 mA. Each solenoid draws about 9 W of power. A two ampere current is necessary to deliver the 36 W necessary to operate 4 solenoids at once in a 4-note chord. If it were necessary to play all 13 notes at once, a 113 W power supply would be required.

Each of the 13 strings is tensioned between 15-20 lbs, as per their specifications. As a result, the upper deck of the harpsichord experiences a 192-260 lb force with a tendency to warp the plywood inward. As a result, 3, 1/8" aluminum L-brackets are attached lengthwise to the underside of the upper deck to counter the force of the tensioned strings, much like the steel brackets present in the metal harp of a piano.

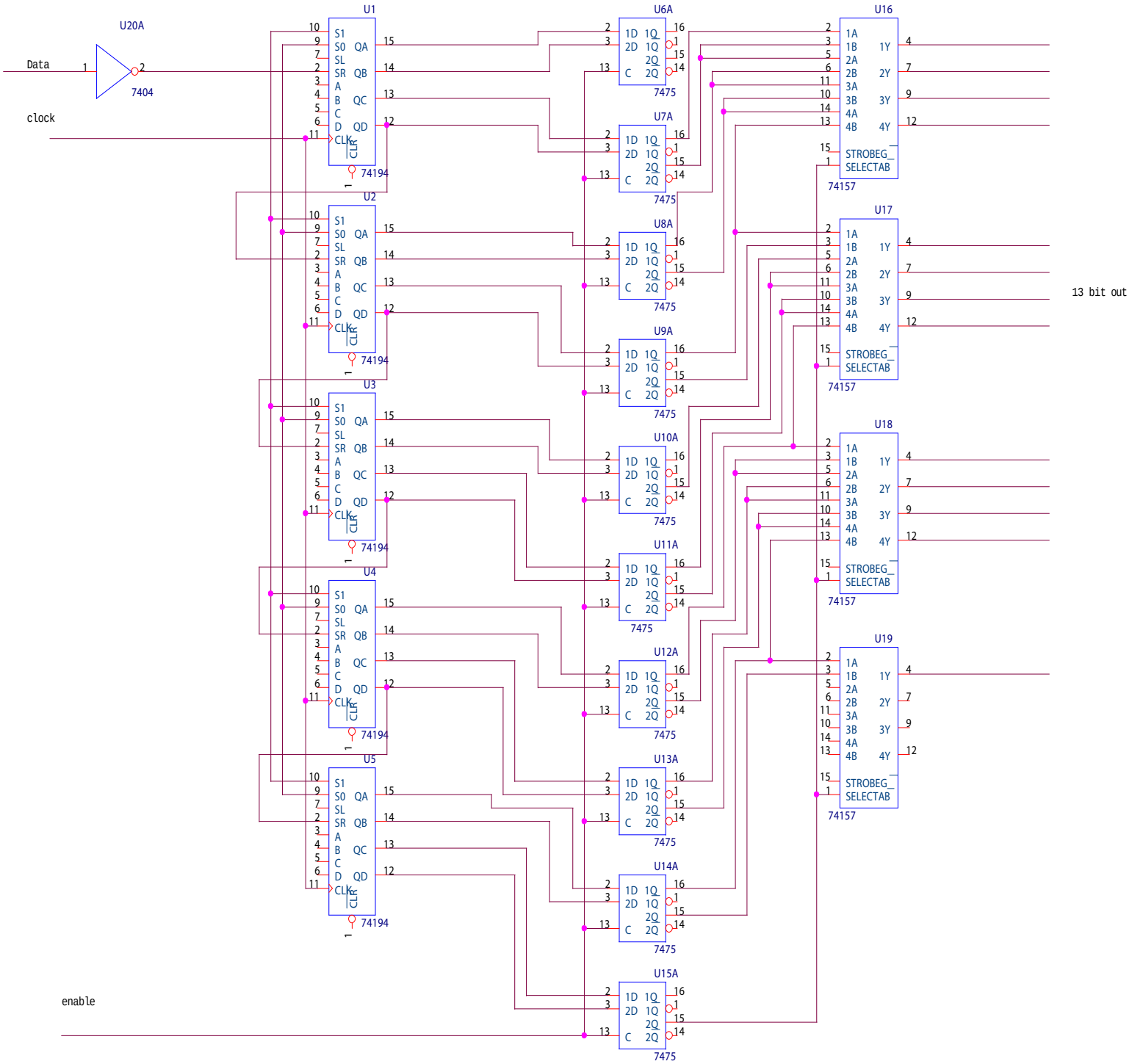
The upper and lower decks of the piano are made with ½", 4 ply, birch plywood. The inter-deck spacers are made with solid cedar. All fastening screws are stainless steel and brass. The side panels, offering partial enclosure, are 1/8" birch plywood.

3. Individual Block Descriptions: Schematics

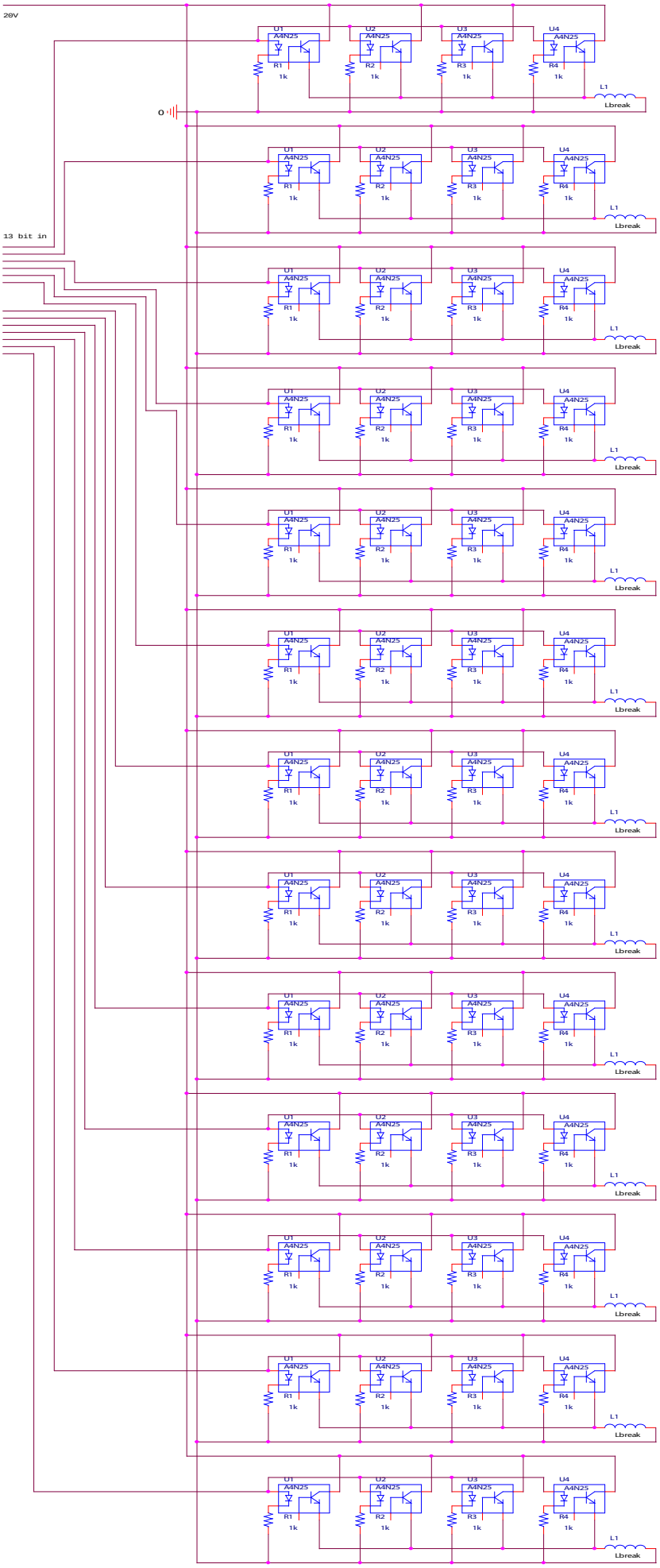
I. Counter



II. 13-Bit Select



III. Optoisolators and solenoids



4. Bill of Materials

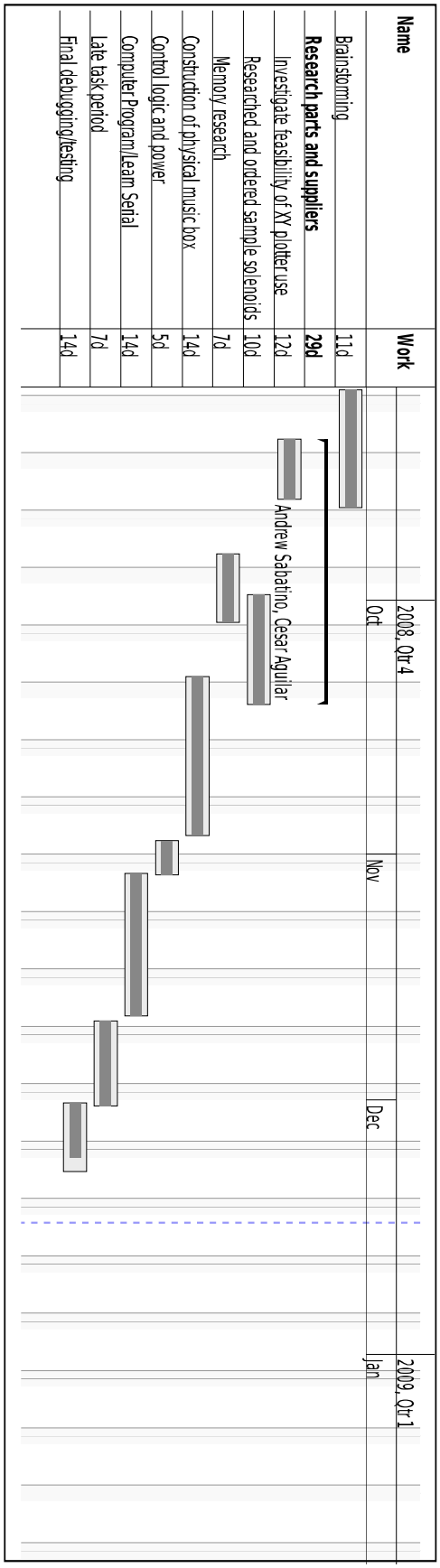
Part	Manufacturer	#	Cost
Solenoid A0008A	Pontiac Coil	13	$ 100.00
Optoisolators PS2502-4-ND	Darlington	13	$ 25.16
5.2mm birch plywood		1	$ 6.89
12mm birch plywood		2	$ 21.42
3/32 cotter pin		4	$ 2.85
brass wood finishing screws, 3 bags		3	$ 9.52
Martin M160 ball end nylon 28-42	Martin	3	$ 14.07
Classical guitar tuners, nickle plated	Ping	3	$ 29.97
Jack damper felt, 1 yard	Zuckermann	1	$ 9.00
Delrin jacks with tongues	Zuckermann	13	$ 38.35
1/8" aluminum l-brackets		3	$ 27.81
self adhesive red felt		2	$ 3.18
2" c-clamps	Husky	6	$ 13.56
Additional electronic parts, boards, wires	various		$ 15.00
Total Cost			$ 316.78

5. Health, Safety, and Environmental Issues

a. Product Dangers
   The product poses minimal threats to the safety of the user. Care should be taken when working with any electrical currents, and the device should not be operated in any environment containing excess moisture. The product contains moving parts, and care should be taken to avoid contact with these parts when in operation, as they employ a significant amount of force. In addition, care should be taken that all inputs are applied to the proper terminals, as a number of components can be destroyed if an error in application is made. Note that the solenoids will heat when operated. Operation should be discontinued immediately if excessive heat is evident in any part of the device.
b. Health Hazards
   The product exclusively uses parts that are ROHS compliant, and, as a result, poses no notable health hazards.
c. Environmental Hazards
   i. The product is entirely self-contained, and is not a significant source of electromagnetic interference.
   ii. All parts are operated well within their recommended power range, though care should be taken to apply the inputs to the correct terminals. In addition, the current usage of the device should be continually monitored, and should never exceed 2A at

the 18V supply or 1A at the 5V supply. This device should not be operated in any environment containing excess moisture.

6. Final Gantt Chart



7. Criticism of the Course

Overall, we found the relatively free form of the course useful, as we set our own schedule and adhered to it. We were able to divide our labor very effectively. As is perhaps to be expected, we spent the first few weeks exploring various design possibilities that we did not end up employing, named the use of an XY plotter to create a player violin. The required weekly progress reports were useful in keeping us active. One additional thing that may have been useful is a midterm-like review about one week before the final presentations, just to bring to light any design and software issues slightly earlier, though an internal review could have achieved the same result and we could have perhaps corrected our small timing error prior to final presentation.

Appendix 1: Code

```
#include "com.h"
#include <stdio.h>

static HANDLE _hCom;


BOOL openComPort (const char* port, const char* baudrate)
{
  char buildStr[50];
  DCB dcb;
  COMMTIMEOUTS timeouts = {0};


  _hCom = CreateFile(port,
                     GENERIC_READ | GENERIC_WRITE,
                     0,
                     0,
                     OPEN_EXISTING,
                     0,
                     0);

  if(_hCom == INVALID_HANDLE_VALUE)
  {
    _hCom = NULL;
    return FALSE;
  }

  /* set timeouts */
  timeouts.ReadTotalTimeoutConstant    = 100;
  timeouts.ReadTotalTimeoutMultiplier  = 0;
  timeouts.WriteTotalTimeoutMultiplier = 0;
  timeouts.WriteTotalTimeoutConstant   = 0;
  if(SetCommTimeouts(_hCom, &timeouts) == FALSE)
    return FALSE;

  dcb.DCBlength = sizeof(DCB);
  if(GetCommState(_hCom, &dcb) == FALSE)
    return FALSE;

  /* Simplified way of setting up the COM port using strings: */
  buildStr[0] = '\0';
  strcat(buildStr, "baud=");
  strcat(buildStr, baudrate);
  strcat(buildStr," parity=N data=8 stop=1");
```

```
  /* (A more effective way is to setup the members of the DCB struct manually,
     then you don't need BuildCommDCB) */

  BuildCommDCB(buildStr, &dcb);
  return SetCommState(_hCom, &dcb);
}

void closeComPort(void)
{

#ifndef _COM_H
#define _COM_H

#include <windows.h>
#include <string.h>

BOOL  openComPort  (const char* port, const char* baudrate);

void  closeComPort (void);

DWORD sendData     (const char* data, DWORD size);

DWORD receiveData  (char* data, DWORD size);

#define HEX__(n) 0x##n##LU

#define B8__(x) ((x&0x0000000FLU)?1:0)      \
              +((x&0x000000F0LU)?2:0)      \
              +((x&0x00000F00LU)?4:0)      \
              +((x&0x0000F000LU)?8:0)      \
              +((x&0x000F0000LU)?16:0)     \
              +((x&0x00F00000LU)?32:0)     \
              +((x&0x0F000000LU)?64:0)     \
              +((x&0xF0000000LU)?128:0)

#define B8(d) ((unsigned char)B8__(HEX__(d)))

#define B16(dmsb,dlsb) (((unsigned short)B8(dmsb)<<8)     \
                      + B8(dlsb))

#define B32(dmsb,db2,db3,dlsb) (((unsigned long)B8(dmsb)<<24)      \
                              + ((unsigned long)B8(db2)<<16) \
                              + ((unsigned long)B8(db3)<<8)    \
                              + B8(dlsb))
#endif /* _COMPORT_H */
```

Appendix 2: Sample Input Song: “What Child is This?”

```
160:E:6
0:G:3 64:A:6 96:B:4 144:C2:12 160:B:6
0:A:3 64:F#:6 96:D:4 144:E:12 160:F#:6
0:G:3 64:E:6 96:E:4 144:D:12 160:E:6
0:F#:2 96:B:3 160:E:6
0:G:3 64:A:6 96:B:4 144:C2:12 160:B:6
0:A:3 64:F#:6 96:D:4 144:E:12 160:F#:6
0:G:4 48:F#:12 64:E:6 96:D#:4 144:C1:12 160:D#:6
0:E:2 96:E:3
```

Appendix 3: Sample Processed Song

```
#include <stdio.h>
#include "com.h"

int main(void){
	unsigned char x[2];
	int i=0;
	openComPort("COM1","900");
	x[1] = B8(00000000);
	x[0] = B8(00000000);
	sendData(x,2);
	printf("%d\n",i++);
	x[1] = B8(00000000);
	x[0] = B8(00000000);
	sendData(x,2);
	printf("%d\n",i++);
	x[1] = B8(00000000);
	x[0] = B8(00000000);
	sendData(x,2);
	printf("%d\n",i++);
	x[1] = B8(00000000);
	x[0] = B8(00000000);
	sendData(x,2);
	printf("%d\n",i++);
	x[1] = B8(00000000);
	x[0] = B8(00000000);
	sendData(x,2);
	printf("%d\n",i++);
	x[1] = B8(00000000);
	x[0] = B8(00000000);
```