<u>EECS 129B Winter 2008 Final Project Report</u>

Team Members

Last Name	**First Name**
Rubow	Erik
Birkel	Anton

Player Harmonica

1. Abstract

The goal of the Player Harmonica project was to build a device that could play the harmonica autonomously and could be reprogrammed to play arbitrary songs. This was a challenging endeavor for two computer engineers because of the amount of mechanical engineering involved. Basically, we had to come up with a way to blow or suck air through a selected hole in the harmonica, and control this electronically with a programmable microcontroller. We chose to avoid any techniques involving the moving of the harmonica or any air tubes back and forth, anticipating that quick and accurate movements with such a system would be difficult. Instead we employed a technique in which a system of overlaid masks controlled by solenoids, each in an on or off state, would determine the path and direction of air. The song to be played is stored in a small EEPROM internal to the microcontroller, so that it is non-volatile and can be updated over a serial link without reloading the firmware.

2. Project overview (High level)

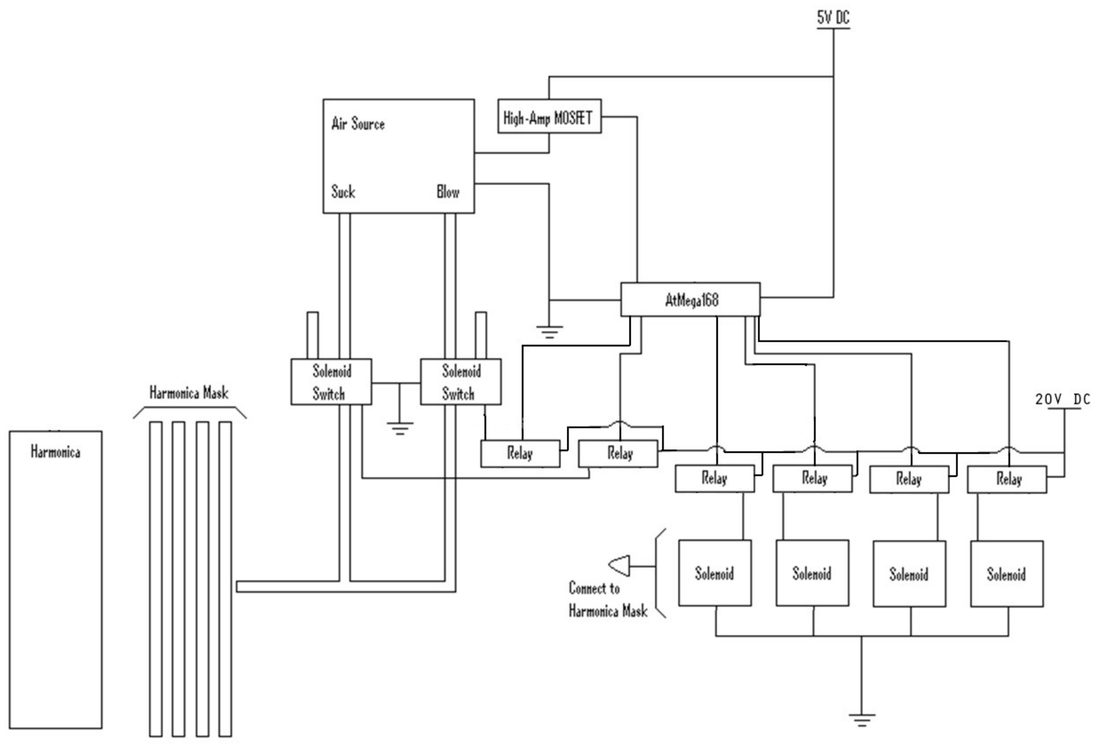

Figure 1 - High-Level Diagram

3. Project components

1) Mechanical Engineering
   - -Find an air supply with suitable blow and suck strengths
   - -Design and build the mask system that selects the path of air
   - -Design and build the mask system that selects the direction of air
   - -Choose the proper dimensions so that everything fits together nicely

2) Electrical Engineering
   - -Provide stable voltage sources with high current and wattage output
   - -Choose relays with a small coil voltage and high load voltage and current
   - -Design circuit to provide variable voltage source with very high current capacity for the air pump

3) Software Engineering
   - -Encode musical notes as bytes and design song file formats (binary and ascii)
   - -Design a simple file transfer protocol for writing to the EEPROM over serial link

-Make all functions non-blocking to enable quick response to user input
-Create an interface for manual control of harmonica operation
-Implement client application to run on a computer allowing a user to create songs as text files, convert them to binary, upload them, initiate and stop playback, and control the harmonica manually.

4. Design description

The tubes carrying air (for blowing and sucking) are directly attached to the first stage of masks, which select the direction of air into an intermediate air chamber. One solenoid controls each tube. If one solenoid is on, air will be blown into or sucked out of the air chamber. If neither solenoid is on, no air will pass through the air chamber. The second stage of masks consists of 4 sheets of Plexiglas with holes arranged in a binary fashion in them (see Figure 2). The sheets are moved up and down by solenoids in order to select a specific hole on the harmonica for air to pass through.

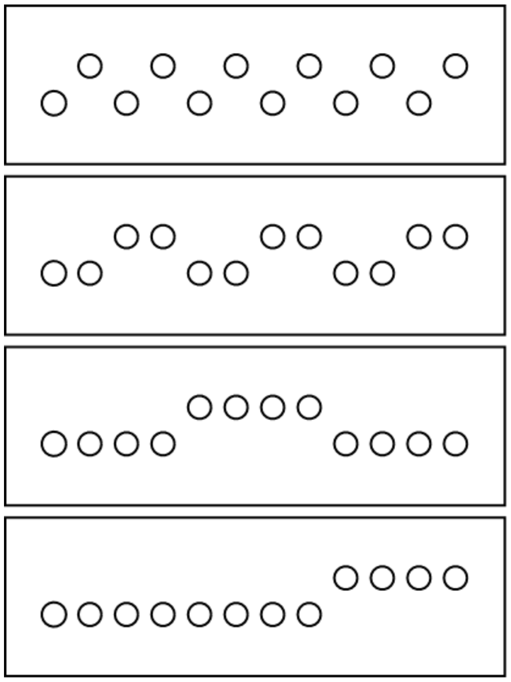

Figure 2 – Overlaid Masks Concept

Each solenoid is controlled through a relay switch because the ATMega168 is unable to provide large enough currents. A MOSFET with a high current rating is used to control the air source. By varying the voltage between 1.5V and 6.0V DC we can control the speed of the air created by the fan, and by extension the volume of the harmonica. See Figure 3 for an electrical schematic.

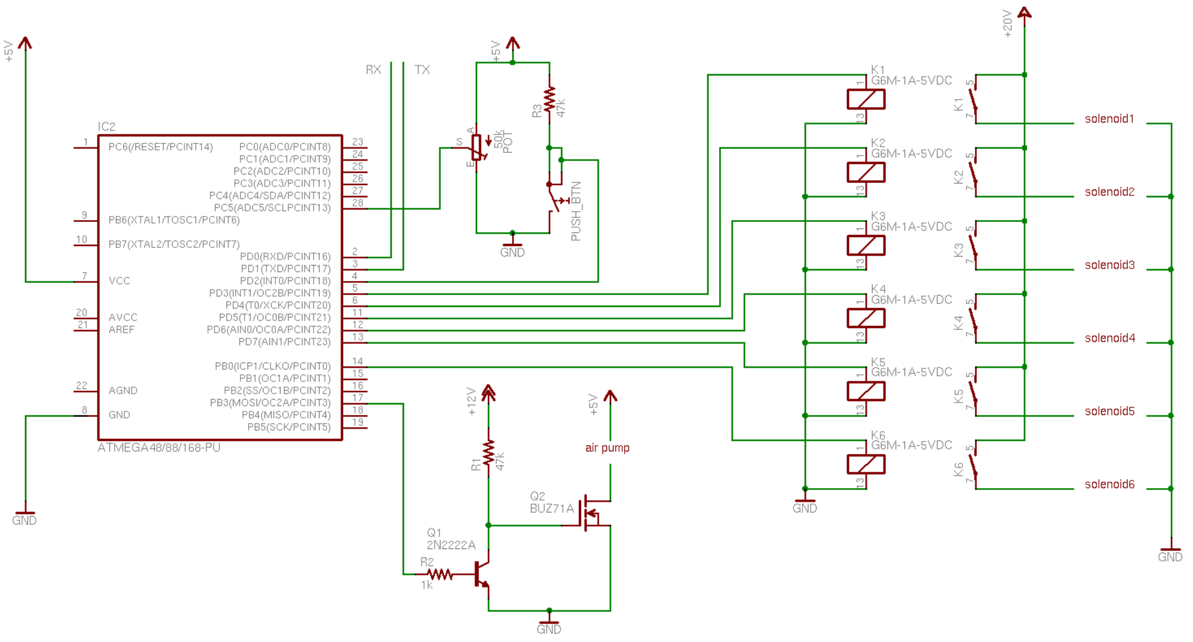

Figure 3 – Electrical Design

The user is provided with a push button, a potentiometer, and a serial interface for interaction with the device. The microcontroller checks every 10ms and processes the input immediately. Song playback is done in a non-blocking fashion so that the user interface is still available while a song is playing. The software was essentially designed as a state machine. A button push in wait mode puts the microcontroller into play mode, and vice versa. The potentiometer controls the air supply using a PWM output and the previously described MOSFET circuit. The primary function of the serial interface is to write to and read from the microcontroller's EEPROM. However, it is also used to manually control the harmonica and to start and stop song playback. The protocol is expandable. A special command byte puts the microcontroller into command mode, after which a particular command is sent to put it into read mode, write mode, playback mode, or some other mode. If a command byte is found in a data stream, it is preceded by an escape byte. An escape byte in a data stream is also preceded by an escape byte. See Figure 4 for a representation of the microcontroller's operation. The binary song format is as follows: The first byte is the number of notes. The second byte is the tempo in beats per minute. The rest of the file consists of a pair of bytes for each note. The first is the encoded note value. The second is the note length, with a quantum of one third of a sixteenth note. The ascii file format is similar: The first line is the number of notes (in decimal). The second line is the tempo (in decimal). Each remaining line corresponds to a note and contains a two-character note (C4, E6, etc.) and the duration (in decimal, same quantum) separated by a space.

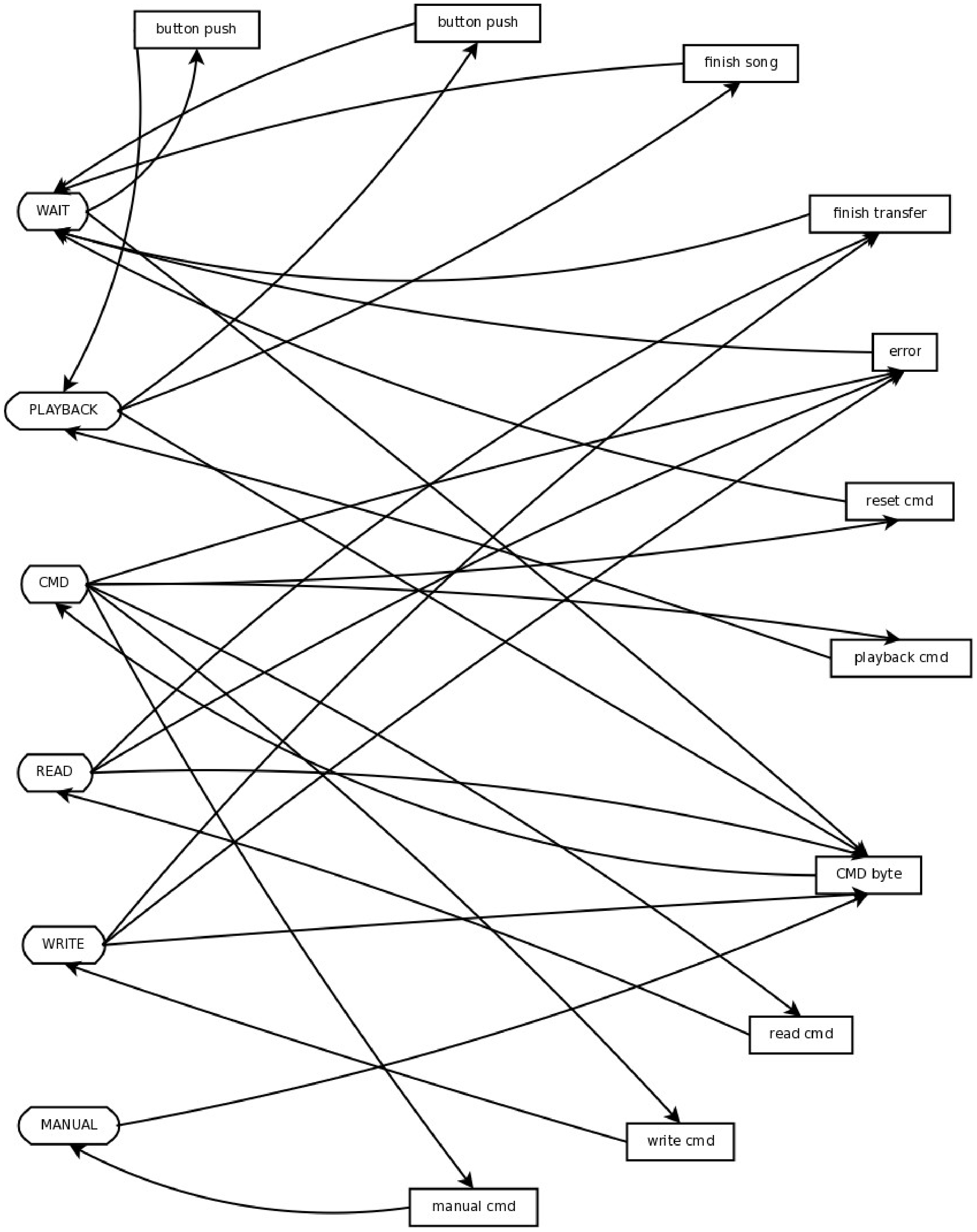
button push
button push
finish song
finish transfer
error
reset cmd
playback cmd
CMD byte
read cmd
write cmd
manual cmd
WAIT
PLAYBACK
CMD
READ
WRITE
MANUAL

Figure 4 – Microcontroller Software State Machine

5. System test plan

Test Number	Description of Set-up	Input or Stimulus	Expected Behavior
1	Only air pump, microcontroller, and control circuit (no solenoids or mask assembly).	Microcontroller provides PWM air pump control signal, low duty cycle to high duty cycle to low duty cycle.	Air pump is powered on, air stream strength varies with control signal, air pump is powered off.
2	Only soleniods, physical mask assembly, microcontroller, and control circuits (no air).	Microcontroller provides all binary combinations of solenoid control signals.	Each solenoid lifts its mask when its control signal is high, for all combinations. The desired hole-path is in alignment in each case.
3	Air pump, mask assembly, solenoids, microcontroller, and control circuits (full assembly).	Microcontroller provides high duty cycle PWM signal for the air pump and solenoid control signals so as to play each “blow” note.	Each “blow” note is heard.
4	Full assembly.	Microcontroller provides high duty-cycle PWM signal to air pump and solenoid control signals so as to play each “suck” note.	Each “suck” note is heard.

5	Full assembly.	Microcontroller provides high duty-cycle PWM signal to air pump and solenoid control signals so as to play a C scale spanning the range of the harmonica. Increase the speed until a limit is reached.	Each note is heard. The maximum transition speed is reasonable.
6	Full assembly plus usb connection to a computer. Microcontroller is pre-programmed with one song that is played when a user pushes a button.	User pushes button, waits for song to finish. Computer loads new song onto the microcontroller without reloading the firmware or restarting. User pushes button again, waits for song to finish, powers off the entire assembly (including the microcontroller), powers it back on and pushes the button again.	Pre-programmed song is played after the first button-press. New song is played after the second button-press and after the third button-press.

6. Project timeline

Chose Harmonica Project – 11/9/07

Ordered Harmonica - 11/22

Redesign includes masks and solenoids – 11/23/07

Ordered a variety of solenoids for testing plus mosfet and relays – 12/1/07

Tested electrical and strength properties of solenoids – 12/7/07

Initial dimensions specification – 1/4/08

Ordered small air pump – 1/11/08

Small air pump insufficient, bought mattress inflater pump – 1/18/08

First attempt to cut plexiglas using score-and-crack method – 1/25/08

Successfully designed and tested air pump control circuit - 1/26/08

Cut some acrylic pieces with band saw – 1/29/08

Finished bulk of cutting acrylic using radial arm saw – 2/1/08

Finished hand-cutting acrylic, filed edges – 2/8/08

Drilled holes in acrylic, glued mask boxes – 2/15/08

Received high-current power supply - 2/16/08

Bought wood, build solenoid holders and air chamber - 2/22/08

Completed Construction - 3/12/08

Completed Software - 3/12/08

7. Division of work

Erik:

- Design mask assembly
- Cut mask parts
- Assemble masks and solenoid mounts
- Write microcontroller code

Anton:

- Design solenoid mount
- Drill mask parts
- Assemble masks and solenoid mounts
- Write computer interface code

8. Cost

Part Number/Name	Unit Cost	Number of Units	Total Cost
Harmonica	59.00	1	59.00
Arduino	34.95	1	34.95
Air pump	11.99	1	11.99
Tubing	10.99	1	10.99
Power Mosfet	0.65	1	0.65
PNP	0.15	1	0.15
Resistors	0.05	3	0.15

Relays	4.50	6	27.00
Power Suppy	31.39	1	31.39
Solenoids	3.95	6	23.70
Plexiglas	6.29	1	6.29
Plywood	3.52	1	3.52
Dowel Rod	1.70	1	1.99
Krazy Glue	0.99	1	0.99

Total cost of project: **$212.76**

9. Problem encountered and comments

One of the largest problems encountered during the project was fabrication. Because Plexiglas was selected as the construction material we were unable to easily cut, drill, and assemble our structure. We had to find fine-tooth bladed saws as well as a diamond tipped drill bit that would not chip or crack the Plexiglas. While researching techniques and strategies to effectively cut and drill the Plexiglas finally lead to the correct tools, we could only find access to hand tools. Unfortunately the hand tools were not only slow and imprecise, but were also prone to snapping the Plexiglas, resulting in many scrapped pieces.

It was also difficult to figure out how to properly control our air supply and solenoids from the Atmega168 microcontroller because the devices are high-voltage and high-current, while the microcontroller only produces 5v at a very low current. While the solution was simply a matter of purchasing relays, we initially had insufficient understanding of relays and their operating characteristics. As such our first relay purchase would not work correctly. After more research and understanding we were able to buy the relays that would work properly in our circuit.