Whac-a-Prof Arcade Game

Submitted 5/10/2013 by:
Group #23
Kelly Banta
Tess Bloom
Caley Follmer
Clinton Knackstedt

Table of Contents

Design Summary

Our project was to design and build a “Whac-a-Prof” that consists of 5 solenoids, each holding the head of a professor that was made on a 3D printer. We 3D modeled these heads which will be familiar to most mechanical engineering students. The professors are Dr. Dave, Dr. Williams, Dr. Vermeulen, Dr. Stansloski, and Joe, our mechatronics TA. The system will randomly select heads to be raised for a limited time, and wait for them to be hit. If hit they would give a point to the player. The game is started by inserting a quarter into the coin slot (see Figure 1,7) and then pressing the start button(see Figure 6), after a few seconds the round will start. An LCD display (see Figure 4,5) indicates the cost of the game by default, if the device has a credit to play it will say that it is ready to run, and when the start button is pressed then the display says that it is running. When the button is pushed, the speaker (see Figure 1) plays a countdown song and once the round starts it plays game time music; at the end of the round a foghorn track is played. As the solenoids randomly go up and down the Arduino waits for a head to be hit and will increment the display (our scoreboard). At the end of a round the display and the audio board is reset and the LCD screen will return to either the price or ready screen depending on if another credit has been received. Team 23 is proud to have created a project that we feel is fun and entertaining for all as well as very unique.

Relevant Figures

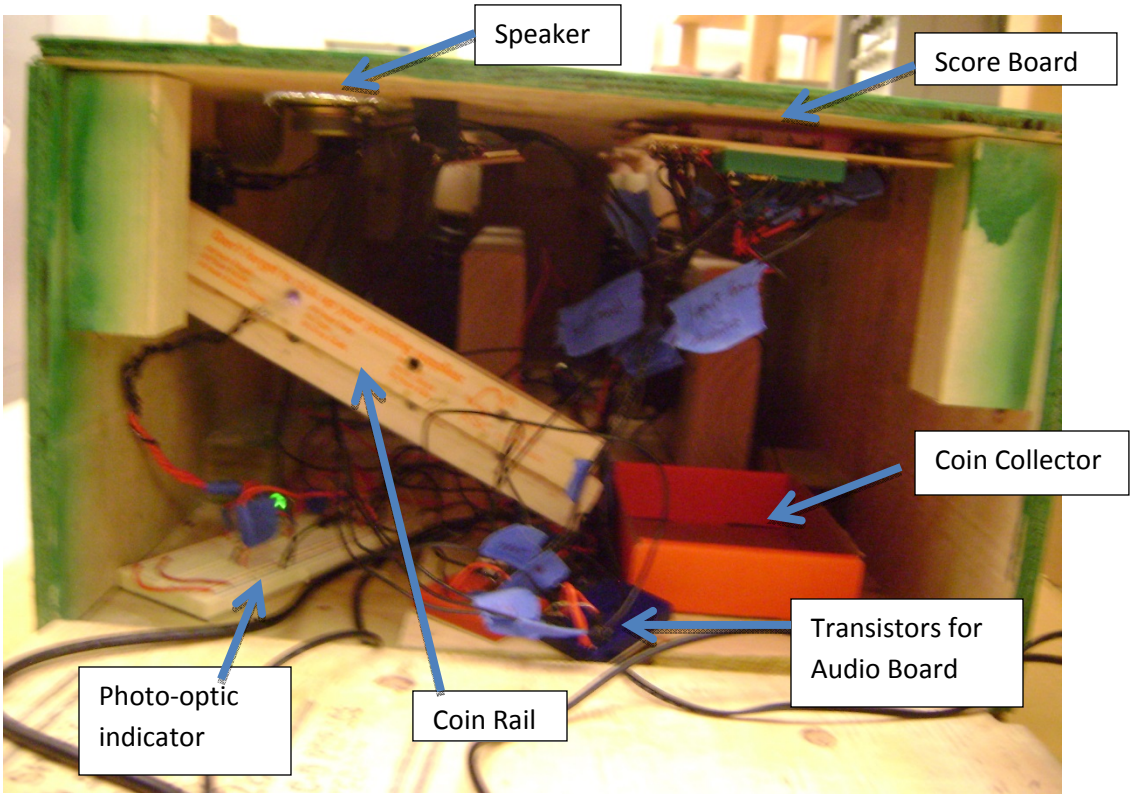


Figure 1: Guts of the device

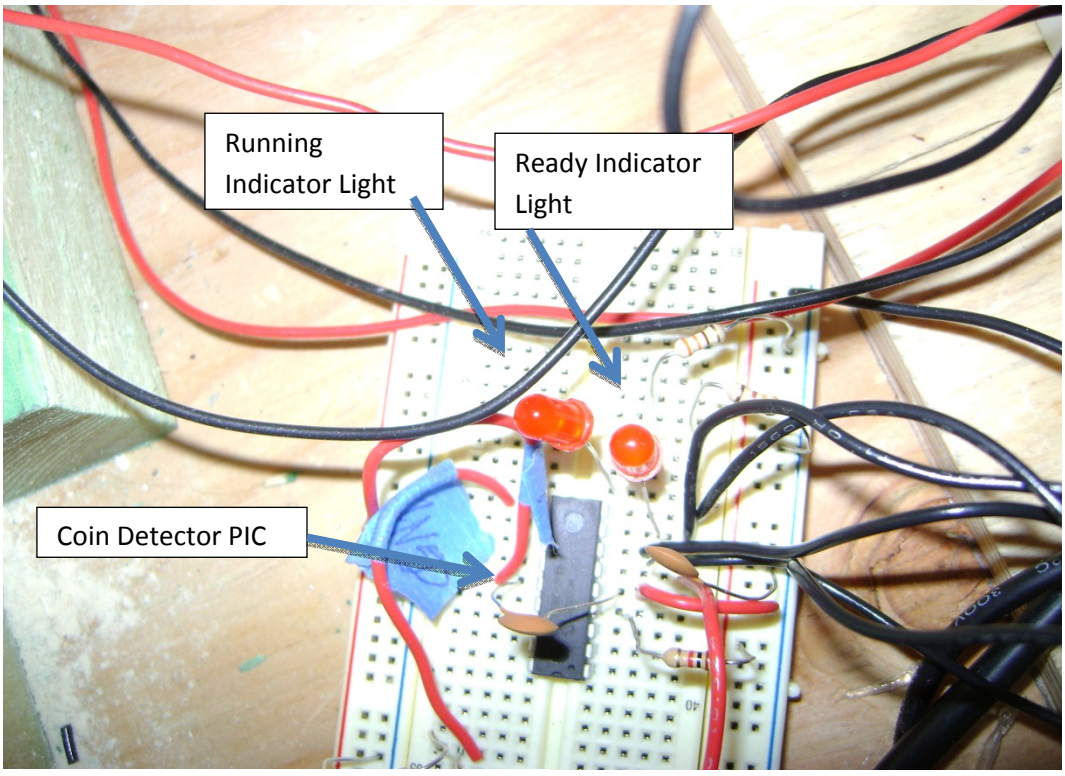


Figure 2: Coin sensor circuit and LED lights

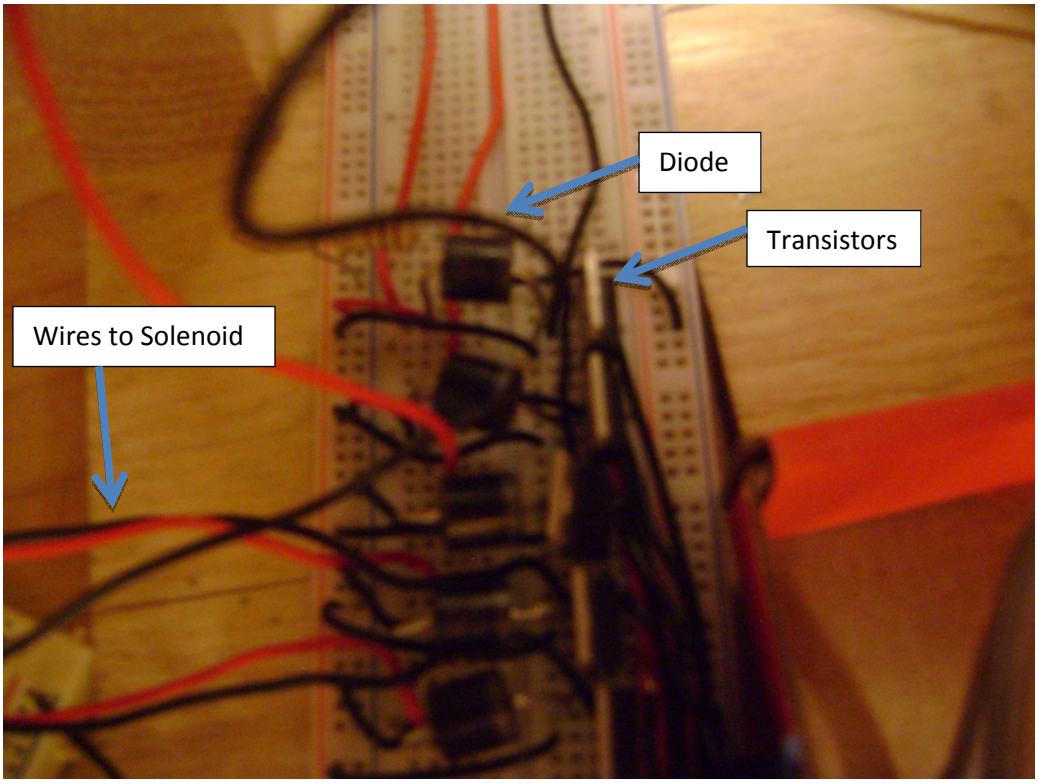


Figure 3: Solenoid Circuitry

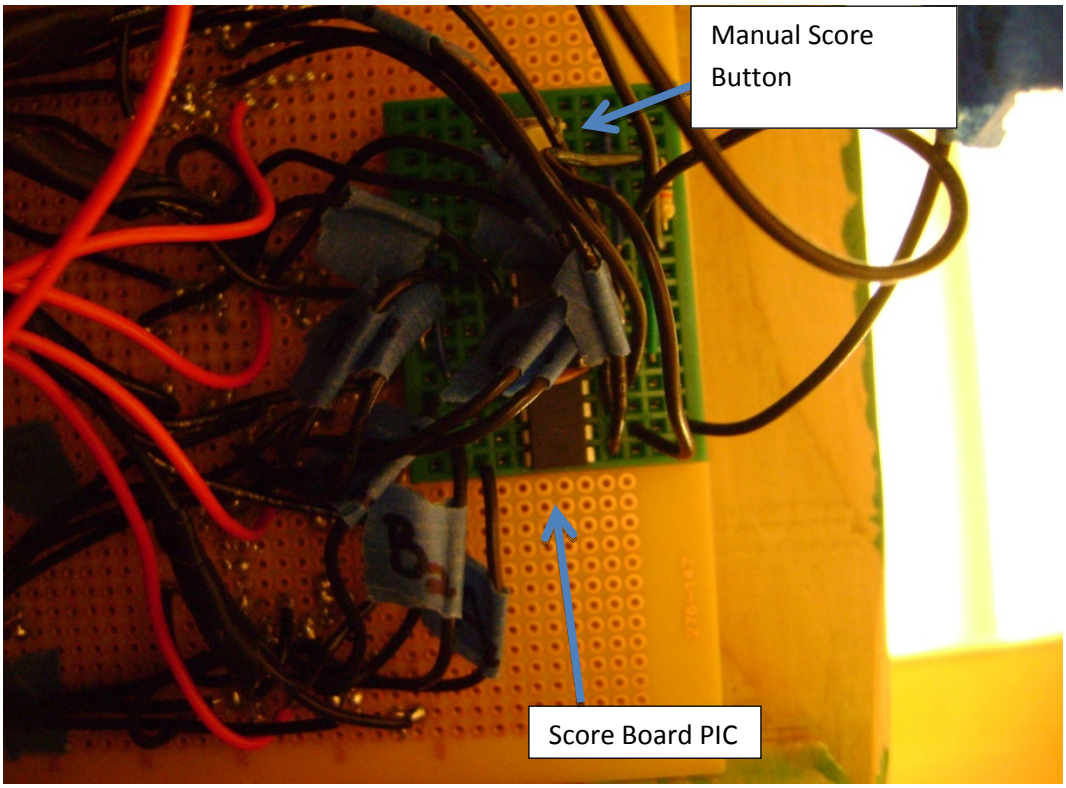


Figure 4: Scoreboard Circuitry

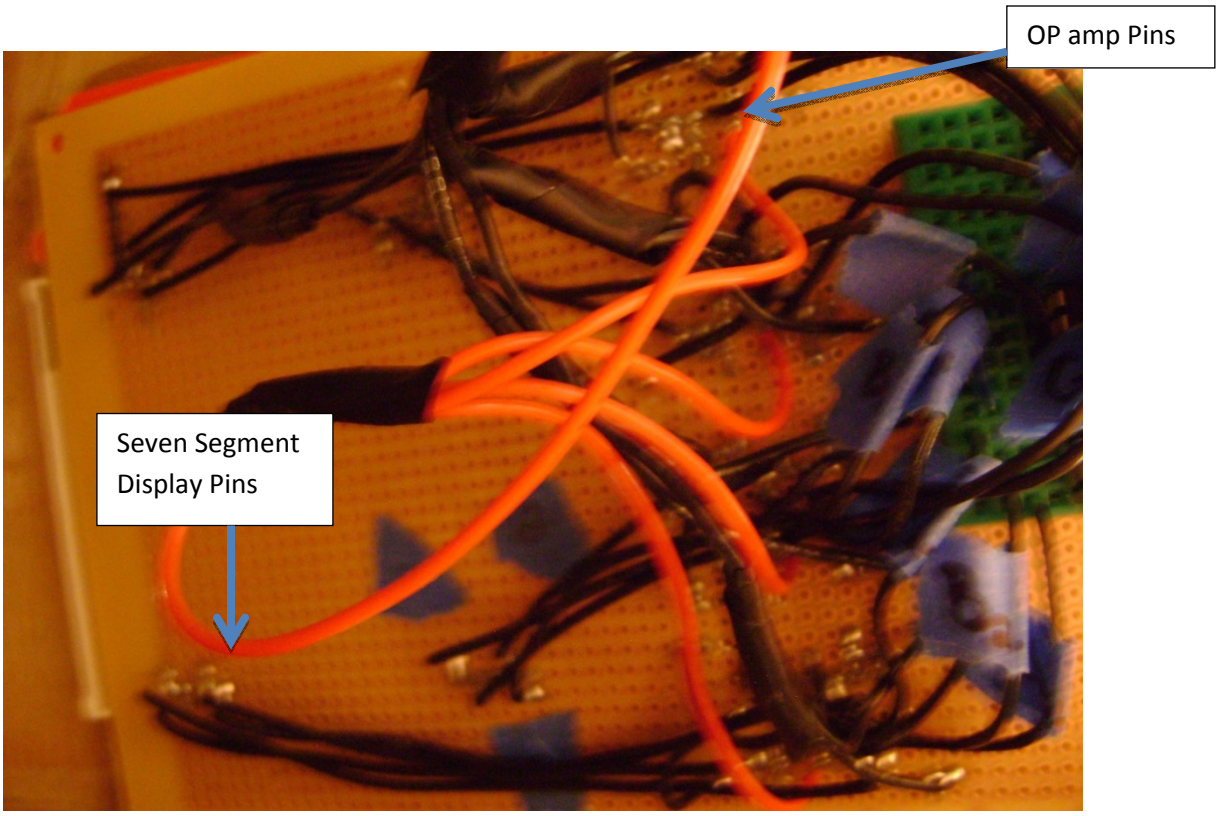


Figure 4: Scoreboard Circuitry

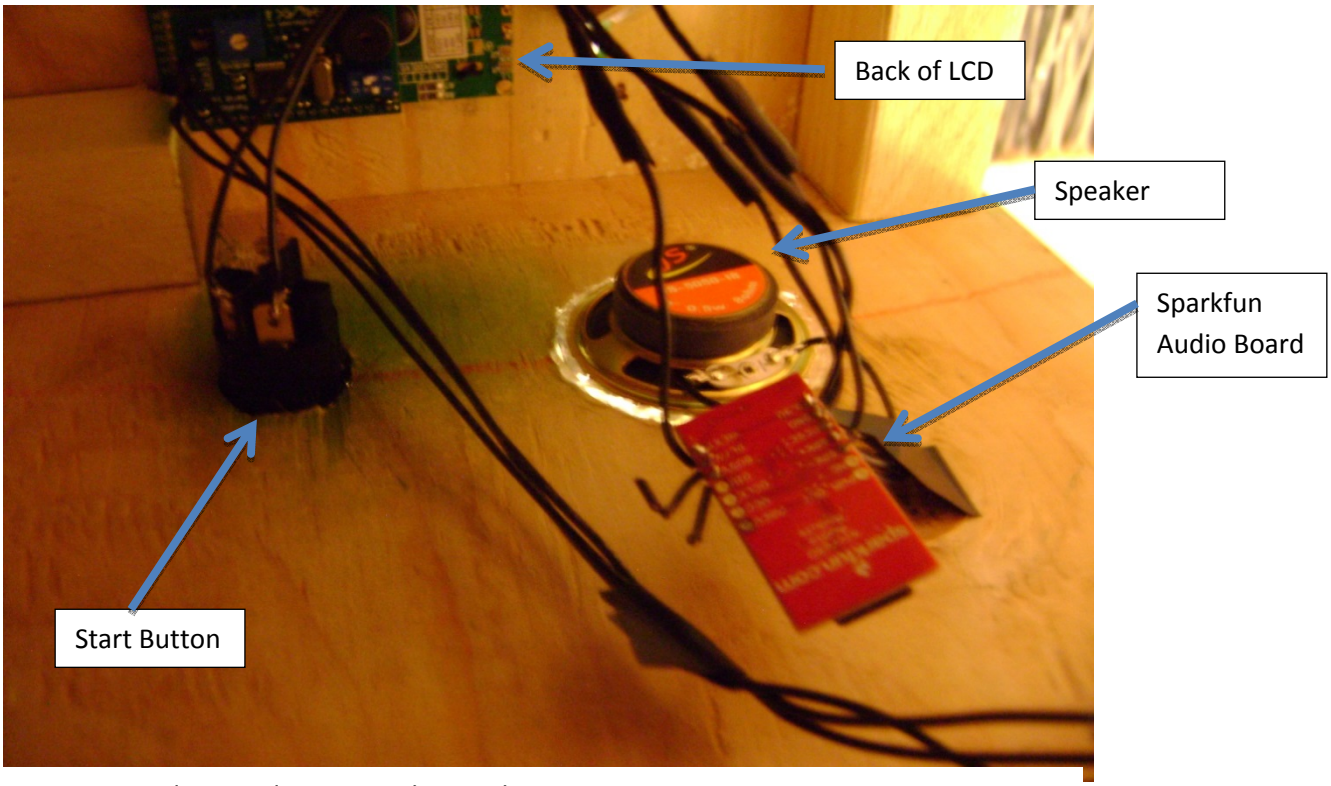


Figure 5: Audio Board, LCD, Speaker, and Start Button

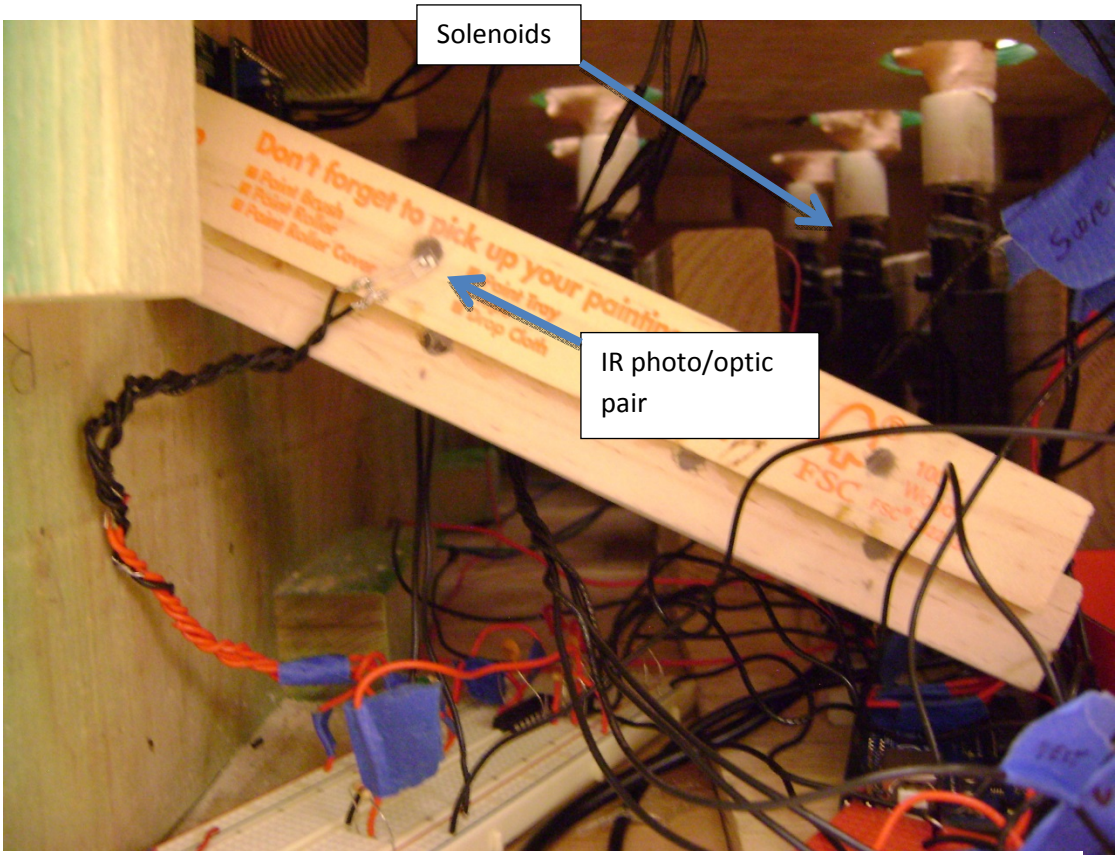


Figure 6: Solenoids and Coin Sensor Track

System Details

Coin Counter, Start Button, and LCD

The coin counter and start button are connected to a PIC16F88, which is then interfaced with an Arduino. The PIC has three lines connecting it to the Arduino: a line indicating whether the game is ready to play, which is determined by the amount of money, another that says whether or not the game is running, and a reset line. The same reset is also connected to the scoreboard.

Initially, the game is waiting for a coin and won't do anything until a coin is added. When a coin is placed in the slot in the front of the game box, it rolls down a ramp where it is detected by an infrared photo optic pair, which signals the PIC to add 25 cents to the game credit. After this, the state of the game is ready to play, and when the start button is pressed the game begins running and 25 cents is subtracted from the player's credit. Multiple coins can be added at once and the PIC will keep track of the total. Thus, multiple games can be played consecutively without adding another coin between rounds as long as there is at least 25 cents of credit before the start button is pressed.

The LCD is run directly by the Arduino, and displays the current state of the game: whether the game is waiting for 25 cents, is ready to play, or is running.

Audio

The speaker is operated by a microSD audio module from Sparkfun which is controlled by the Arduino. For this game, three sound files are stored on the microSD card, and are played at the appropriate times. After the start button is pressed, a countdown track is played. Then the game music begins and the heads start popping up. After 60 heads have gone up, the game finishes and the ending sound plays.

Solenoids/Heads

The solenoids operate at 5 volts and about 2 amps. They are screwed into wooden supports to hold them in place. A BJT transistor was used between the Arduino and each solenoid to allow the solenoids to operate on a higher current than the Arduino can output. Also, a diode was placed in parallel with each solenoid to ensure that there would be no damaging voltage spikes during operation. In order to record hits, a wire was run from the emitter pin on each transistor to the 10 bit A/D converter on the Arduino. If a solenoid is hit while it is on, a negative voltage spike is created by the change in the magnetic field from the solenoid rod being moved against the desired motion. The diode prevents this spike from reaching dangerous levels; however, it does not prevent voltage fluctuations. The spike that is produced from most hits is in the 2 volt range. The Arduino reads the analog value of the voltage spike and if it dips more than two volts while the solenoid is fully active then it will register a hit and turn off the solenoid. If a successful hit is not made within one second, then the head will lower without a hit being registered. Springs were attached between the heads and the bases of the solenoids to ensure that the heads lowered when the solenoids turned off.

The heads of the professors were printed in ABS plastic. Using free software from Autodesk called 123D Catch, photos were uploaded of each professor that were taken from multiple angles while they remained stationary. The software used image recognition to reconstruct a mesh of the scene. Once the mesh was generated it was edited and refined using Autodesk 3ds Max, which is a powerful CAD software that is

free to students and allows a very free manipulation of the mesh. After the aberrations were removed from the mesh, it was exported as a .stl which is the standard format for 3d printing. The printer took just over one hour per head to print at a vertical axis layer height of 300 microns. An acetone bath was given to each head to remove the lines between layers during printing thus making it smooth and glossy. Lastly, the heads were painted to make them more lifelike and then affixed to the solenoids.

Scoreboard

The scoreboard is made up of 2 large (2.24 in) single-digit 10-pin DIP 7-segment LED displays, and is operated by a PIC16F88. When the Arduino registers a hit, it sends a pulse to the PIC, which then increments the score. Each segment of the display corresponds to a pin on the PIC, and the PIC makes the proper pins high or low in order to show the appropriate score.

The 7-segment displays have a common ground, which meant positive logic was required to control each segment individually, so a segment would turn on when its PIC pin went high, and off when it went low. Each segment of the display consists of 4 LEDs in series, and requires at least about 10 V (LEDs turn on at around 7 V but are very dim). Because of this, the 5V output from the PIC needed to be increased. This was accomplished by using op amps to create non-inverting amplifiers between the PIC and each display segment, as shown in Figure 7 below.

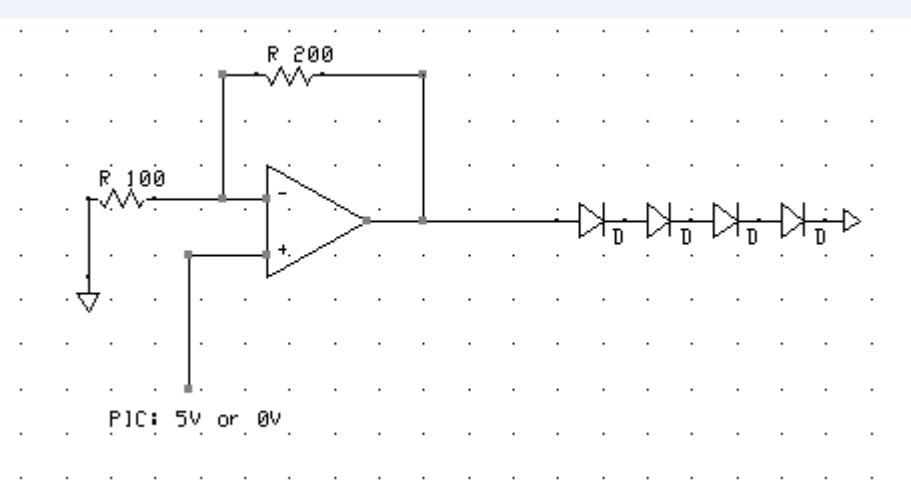
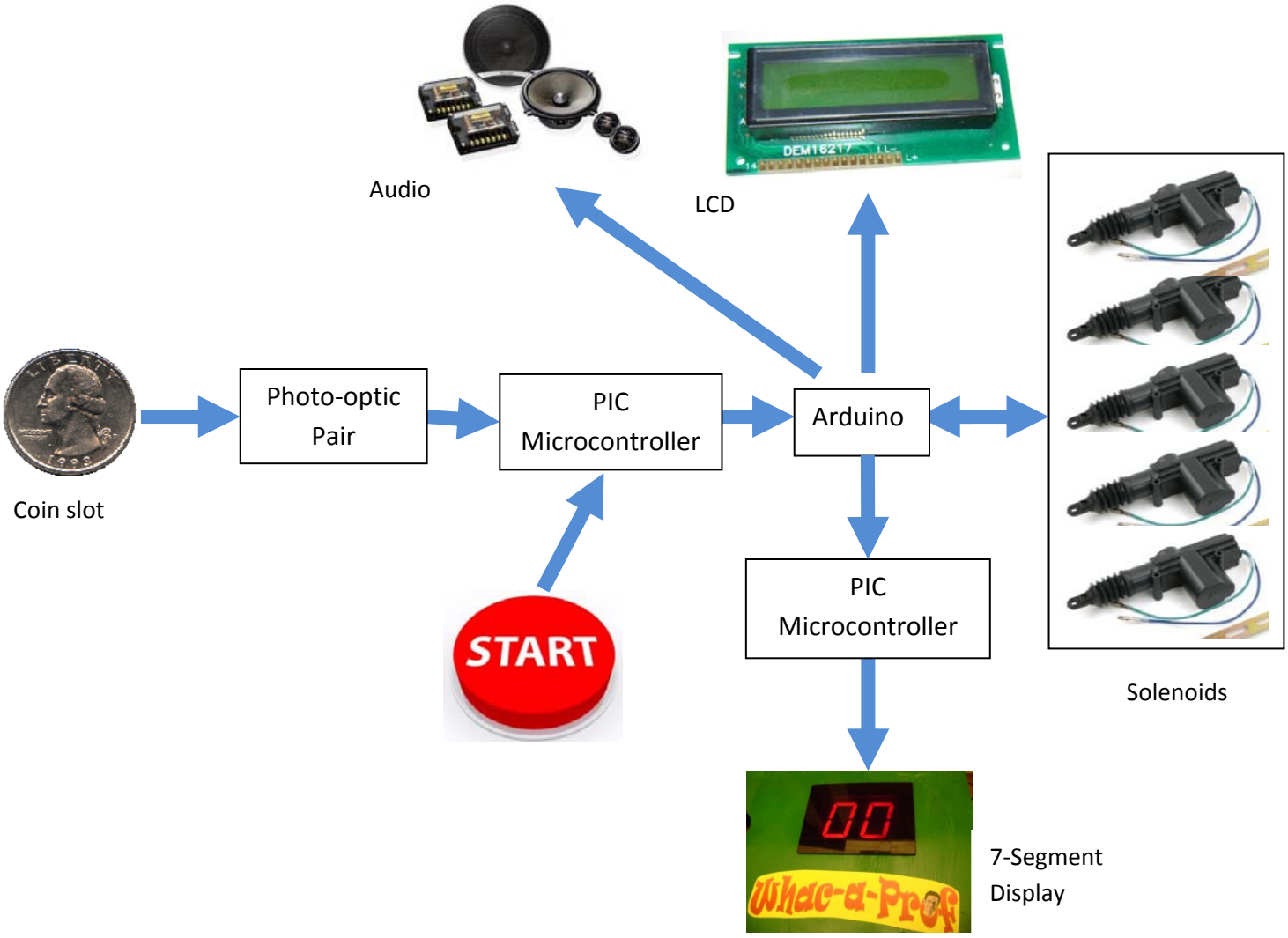


Figure 7: Non-inverting amplifier circuit used for each segment of LED display.

The op amps used were in the form of 4 quad op amp ICs. Power to the op amps is provided by a power supply that produces 16 V (but was nominally 12 V).

***NOTE**: The scoreboard would have been much simpler to make with 7-segment displays that used a common voltage source. Thus, segments could be controlled individually by sinking current, and the op-amps would not have been necessary. (Display would use a common voltage source instead of common ground).

Functional Diagram



Design Evaluation

A. **Output Display**
   - 7-segment LED display– 2 digits; used to display the score of the game
   - LCD -16x2 characters; used to display idle, ready, and running states

B. **Audio Output Device**
   - Audio-Sound Breakout Board loaded with three tracks; countdown, in-play music, and end-of-game sound
   - Speaker connected to sound board

C. **Manual Data Input**
   - Start button
   - Manual scoreboard increment button

D. **Automatic Sensor Input**
   - Photo-optic pair – used as a coin sensor which puts the device into the 'ready' position
   - Voltage measurement from solenoids – the Arduino reads a negative voltage spike when the professors/solenoids are hit and reports that to the scoreboard so it can increment

E. **Actuators, Mechanisms, and Hardware**
   - Solenoids -12VDC, 2.6A, 5/8" throw
   - 3D printing of professor heads that were created
   - Coin chute
   - Wood box created to house device

F. **Logic, Processing, and Control; AND Miscellaneous (functional elements not covered in the categories above)**
   - Programmed logic – Arduino Uno
   - Calculations and data storage/retrieval – two PIC16F88
   - Multiple interfaced PICs controlling the scoreboard, coin sensor, and Start button
   - Arduino Uno – all PICs feed into this as it runs the solenoids and sends data to the two PICs
   - Closed-loop feedback control
   - op amps – used for scoreboard

Partial Parts List

Solenoid

Power Door Lock
Model Number: 93964P1
American Science and Surplus
$4.50 each (5 used in project)

These solenoids are 12 V DC actuators that are push/pull and have a 5/8” throw.

Photo Optic Pair

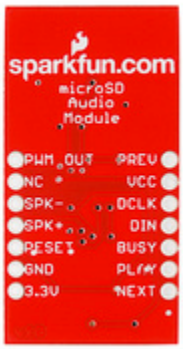
Infrared Emitters and Detectors
Model Number: SEN-00241
Sparkfun
$1.95 per pair

The infrared emitters and detectors are touch-less sensors that can be interfaced with a microcontroller or in the case of our project, an Arduino. The detector is a NPN transistor that is biased by IR light.

Speaker

Speaker – 0.5W 8 Ohm
Model Number: COM-09151
Sparkfun
$1.95
This is a small speaker that is ideal for small-scale robotic projects.

Micro-SD Audio Module



Audio-Sound Breakout – WTV020SD
Model Number: WIG-11125
Sparkfun
$19.95
The Audio-Sound Breakout has a battery connector and a micro-SD card slot. The selected audio can be loaded on the micro-SD card, plugged in to power, and then the playback can be triggered.

Arduino

Arduino Uno-R3
Model Number: DEV-11021
Sparkfun
$29.95 (Acquired for $15)
This arduino uncludes an ATmega328 microcontroller and requires an input voltage of 7-12V. It has 14 digital I/O pins (6 PWM outputs), 6 analog inputs, and 32k flash memory.

Op Amp

LM324 Quad Op Amp (14-Pin DIP)
Model Number: 2761711
RadioShack
$2.39 (Used 4 in project)
These op amps can function when the difference between the two supplies is 3 V to 30 V and VCC is at least 1.5V more positive than the input common mode voltage. In our project, we needed to limit the amount of voltage reaching the 7-segment display so we used these op amps.

Lessons Learned

Problem	Solution	Lesson Learned
Tried to build our own LED 7-segment display to use as our scoreboard. This was dim, hard to orient and did not look as nice as a premade display.	Bought 7-segment displays online which were very cheap ($5) anyways and worked very well. However, op amps became necessary with this solution.	Look online for components and don't assume that it will be easier/cheaper to build your own.
Scoreboard was continuously incrementing even when heads were not hit.	We implemented a pull down resistor on the input to the PIC from the Arduino.	When having issues check to see if adding a resistor or capacitor will solve your problem.
The solenoids would not go down after we hit them or register a point on the scoreboard because the Arduino was not registering a voltage spike.	We had to calibrate the Arduino through trial and error to find an accurate voltage value to be the cutoff used to determine when a head is it.	Sometimes trial and error, although time consuming, can be the best way to solve programming issues.
One of the solenoids would count a score even when the head had not been hit.	We isolated this solenoid from the rest in the programming and changed the voltage spike that the Arduino would read.	If something does not work as a whole try to separate it from the grouping and individually troubleshoot the issue.
The solenoids would stay up even after the voltage to them had been stopped because there was not enough weight to push them back down.	We solved this by gluing a spring from the base of our solenoids to the bottom of our professors in order to pull back down the professors after the voltage had been removed.	Do not be afraid to think outside the box and use different hardware in order to accomplish your goal.
Our audio board had an SD card input but would only read a very specific file type which was difficult to work with.	We had to download a conversion program to attempt to convert the songs to this format and had to cut down the length of our audio clips.	Before buying any part make sure to read all the specific details to guarantee that the programming and manipulating of it will be easy to do.
Applied too much voltage to one of our PICs and an LCD causing it to break	We bought a new LCD screen.	Always check the voltage ratings and make sure to have extras of components that may be hard to get last minute.
We couldn’t control the scoreboard by sinking current with a PIC or other chip because it had a common cathode, and the display required at least 7 V to operate.	We had to use op amps to amplify the voltage output by the PIC.	Make sure to check whether a component can source or sink current and plan accordingly.

Coming into this course all of us knew what a huge task the mechatronics project would be and though it took countless hours to complete, we all learned valuable lessons that we will be able to use in our futures as engineers. Our group worked incredibly well together, to which I think we can attribute our functional and very unique project. We started out working weekly to brainstorm project ideas and functionality and then started picking up the pace working on the actual fabrication, programming, and wiring of our device. Although we all had incredibly different and busy schedules we were all able to make it work and contribute our fair share. We were able to split up responsibilities based on our strengths and weaknesses so that we could be efficient and create the highest quality product possible. This lesson been stated a million times but it is so true, start early! We wish we would have started more of the fabrication early on so that we were not working on that aspect at the last minute. Especially since the end of the semester is already so busy it is important to get a good start on the project.

It took us a while to settle on our "Whac-a-Prof" idea. We originally had our hearts set on creating a Segway until we realized what an undertaking that really would be. We had come up with doing a standard whac-a-mole arcade game but it wasn't until we decided to use professor's heads as the moles that we all became very excited about the prospects of the project. One of the main lessons we learned is to make sure to be passionate or excited about your project. We all loved the idea of 3D printing professor heads and we think we really intrigued all of our classmates with this addition. This excitement made us want to work on the project instead of forced us to work on it. It is definitely important to have fun with it!

Appendix

Table 1: Bill of Materials (Components not mentioned in class)

Part Number	Quantity Number	Name	Price ($)	Total ($)
1	3	Infrared Emitters and Detectors	1.95	1.95
2	1	Audio-Sound Breakout - WTV020SD	19.95	19.95
3	1	Speaker - 0.5W 8Ohm	1.95	1.95
4	1	Arduino Uno - R3	29.95	15
5	6 (used 5)	Power Door Lock	4.5	22.5
6	4	LM324 Quad Op Amp (14-Pin Dip)	2.49	9.96
				71.31

Table 2: Bill of Materials (Components used in lab or mentioned in textbook)

Part Number	Quantity Number	Name	Price ($)	Total ($)
1	5	Common BJT Tramsistors - NPN 2N3904	0.75	3.75
2	1	Resistor 330 Ohm - SMD (strip of 50)	1.5	1.5
3	1	Basic 16x2 Character LCD - RGB Backlight 5V	14.95	14.95
4	6 (Used 5)	NTE5814 - Diode	2.01	12.06
5	6 (Used 5)	NTE152 Transistor	2.39	14.34
6	2	PK15 Amplifying Transistor	3.49	6.98
7	1	330 ohm 1/4W 5% Carbon Film Resistor pk/5	1.49	1.49
8	4 (Used 2)	Display, 7-Seg, CC, Red, I Digit (LED Display)	4.49	17.96
9	1	Coolerguys 100-240v AC to 12 / 5v DC 4pin Molex 2A Power Adapter	14.5	14.5
				87.53

Table 3: Bill of Materials (Hardware)

Part Number	Quantity Number	Name	Price ($)	Total ($)
1	4	Breadboard - Mini Self-Adhesive	2.95	8.85
2	1	Micro Perfboard	3.49	1.49
3	1	Acrylic Sheets	3	3
4	1	1-1/8" Dowel Rod	4.1	4.1
5	1	5/8" Whitewood	4.89	4.89
6	1	2.5 Oz. Solder	5.99	5.99
9	1	Electric Tape	0.99	0.99
11	3	Spring	0.92	3
12	1	Wood Knob	1.49	1.49
13	1	SPST Push Button	5.99	5.99
14	1	Wood	30	30
				69.79

Table 4: Bill of Materials (Tools, fasteners, paint, etc.)

Part Number	Quantity Number	Name	Price ($)	Total ($)
1	1	Soldering Iron	27.95	27.95
2	1	Black insulated wire	3.96	3.96
3	1	Soldering Iron Stand	7.95	7.95
4	1	Super Glue	2.47	2.47
5	4	Flat washers	0.45	1.8
6	1	#14 x 1-1/2 in. Box Nails	3.72	3.72
7	1	#8 x 1-1/4 in. Zinc-Plated Flat-Head Wood Screw (100-Pieces)	5.58	5.58
8	1	#6 x 1 in. Zinc-Plated Round-Head Slotted Drive Wood Screw (12-Pieces)	1.18	1.18
9	1	Hinges	3.99	3.99
10	10	Sand Paper	1.19	11.9
7	1	Pointing Trowel 5.5"	3.29	3.29
8	2	Wood Filler	6.99	13.98
10	2	Green Spray Paint Gloss	4.49	8.98
				96.75

Total Cost including tax, shipping, and handling (without materials in Table 4):

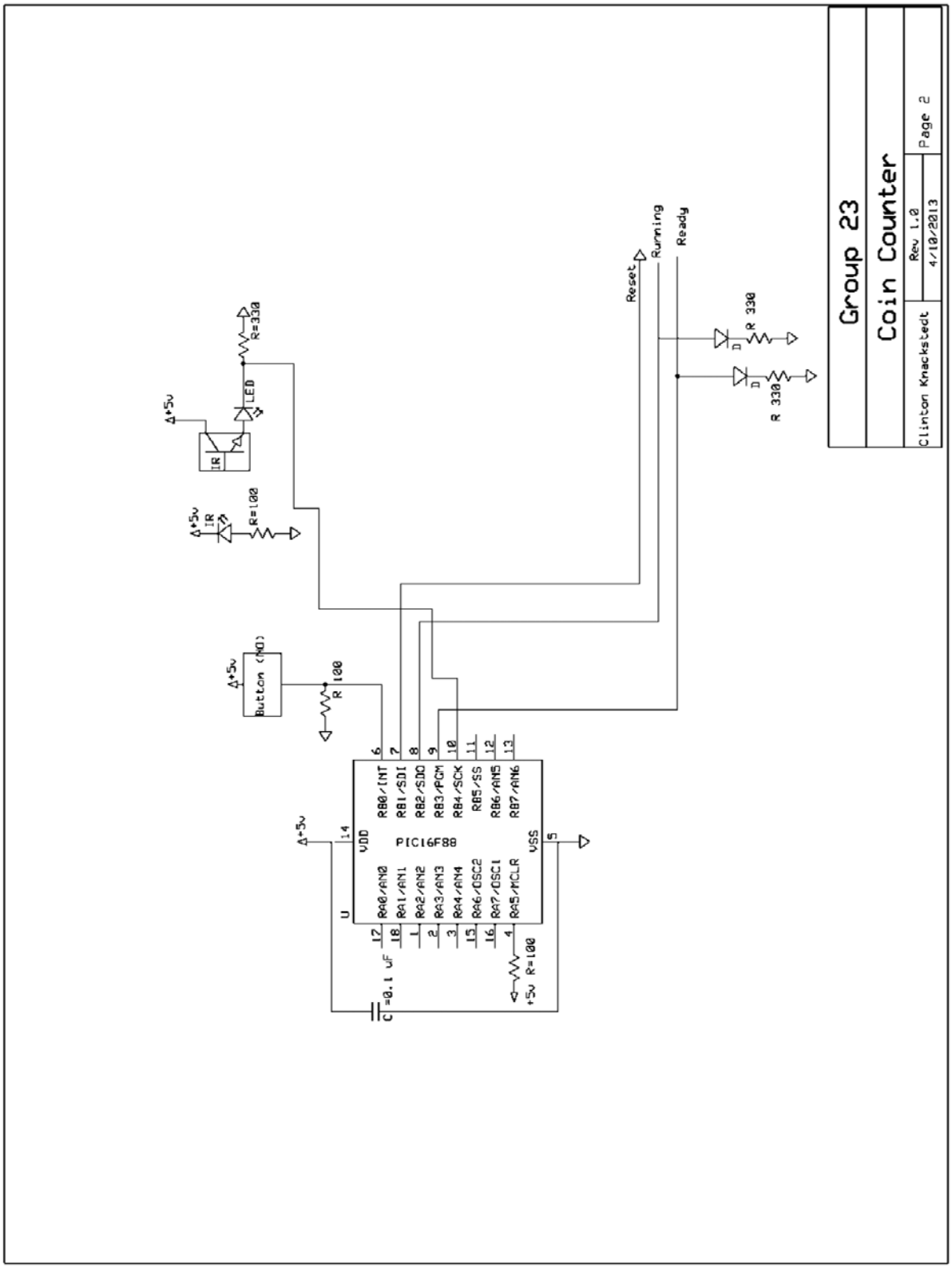
Total Tax and Shipping and Handling:	**40.36**
Total Project Cost	**268.98**

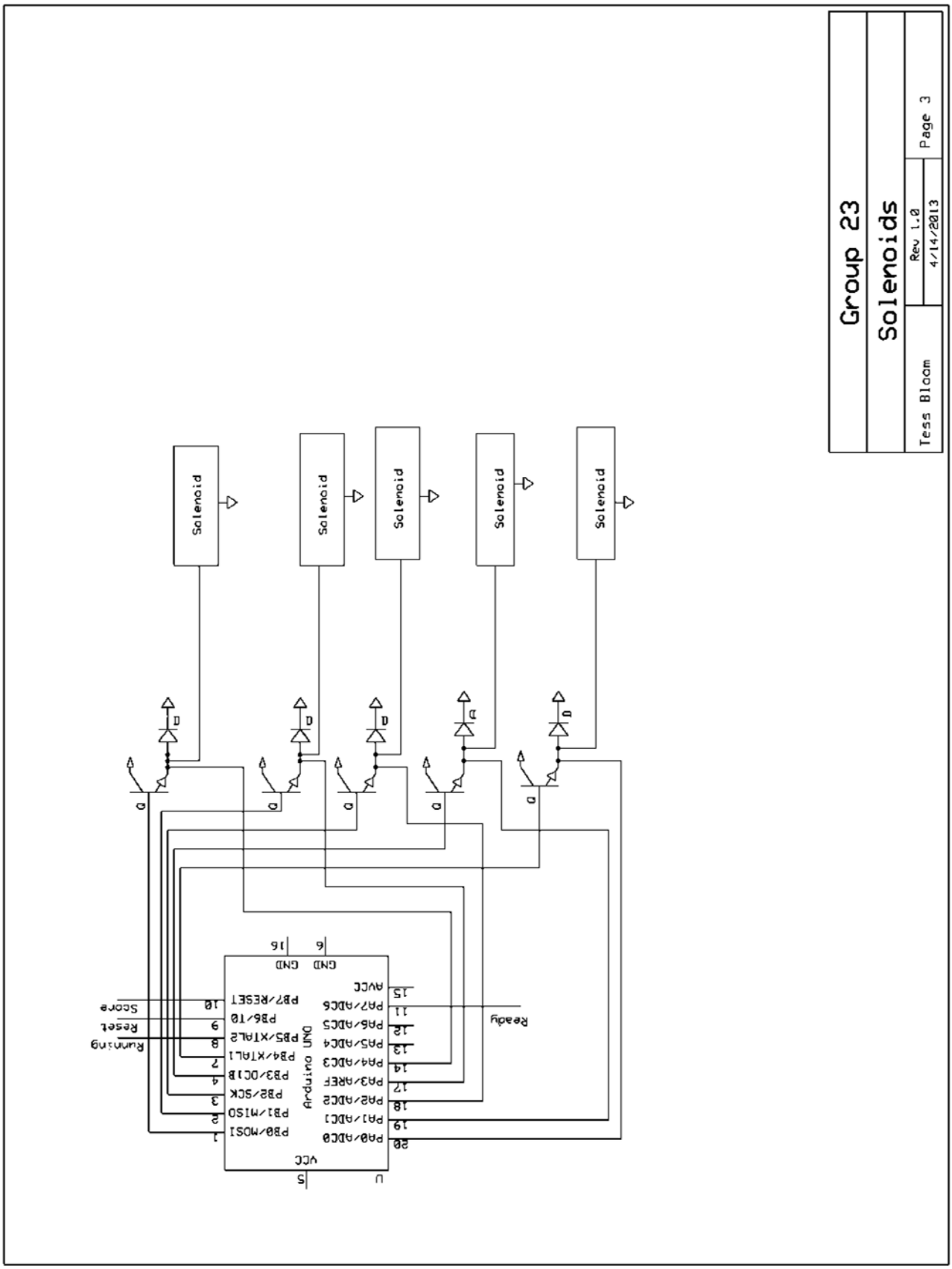
In the total project cost, the price of the materials included in Table 4 were left out because these are either tools that were bought or things such as paint that are not necessary to the functionality of the device.

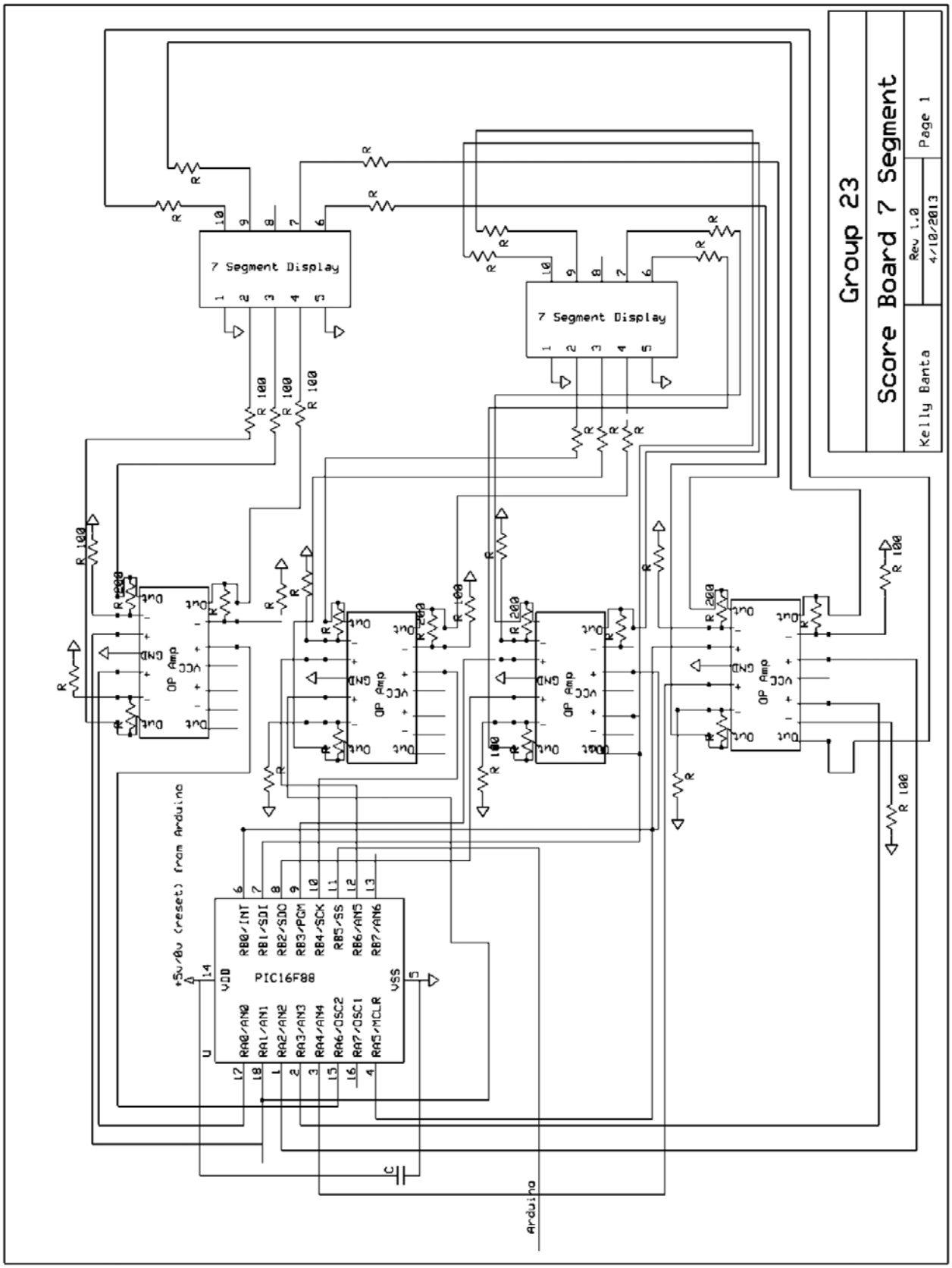
Figure 1: Price that an actual Whac-A-Mole game is selling for (from http://www.hammacher.com)

The full-scale Whac-a-Mole arcade game that is shown above is selling for $4,500. Including the cost of the hardware, glue, paint, electronics, etc., the cost to manufacture our Whac-a-Mole model was about 1/19 the cost of what this model is selling for.

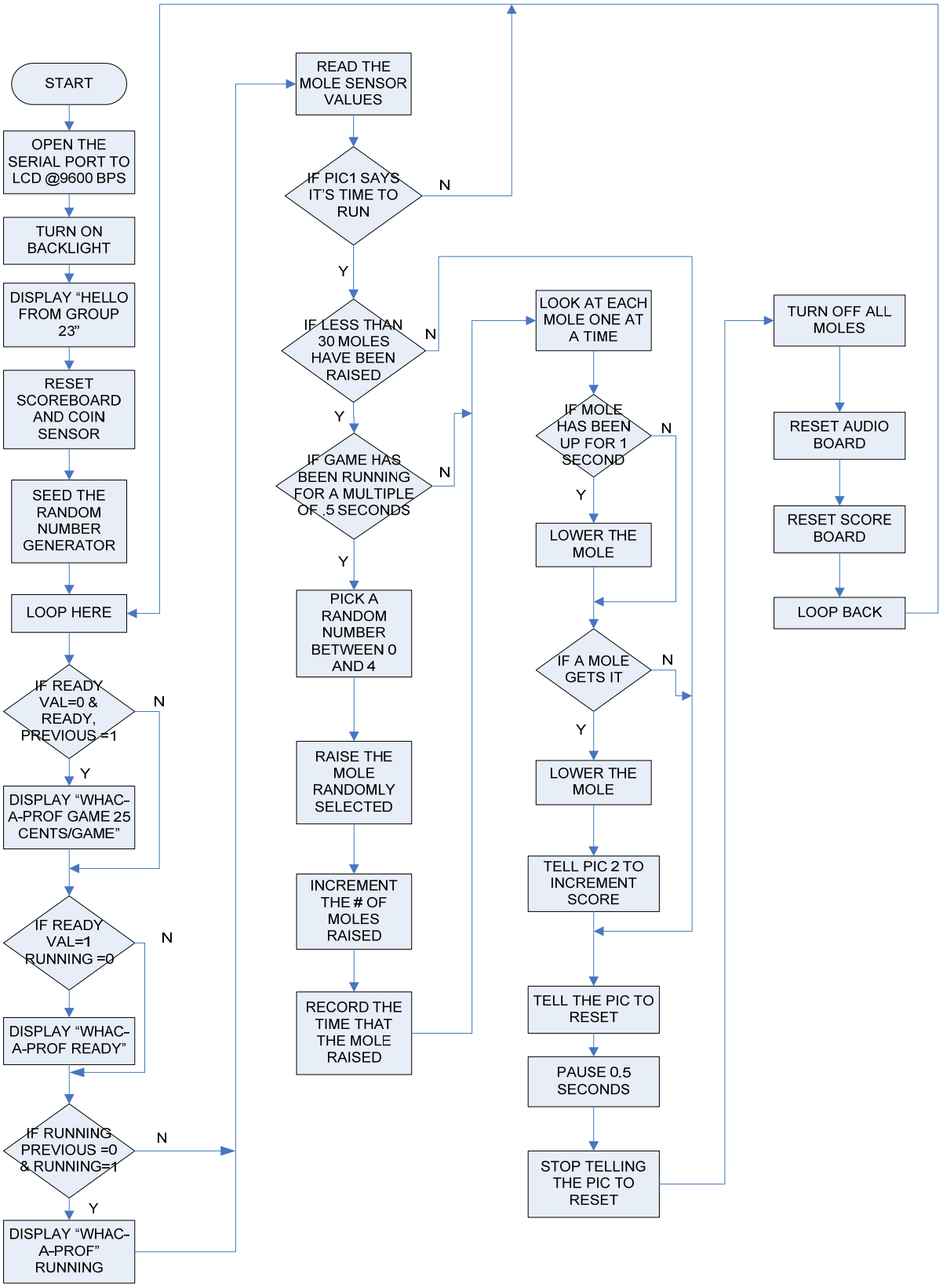
Wiring Diagrams



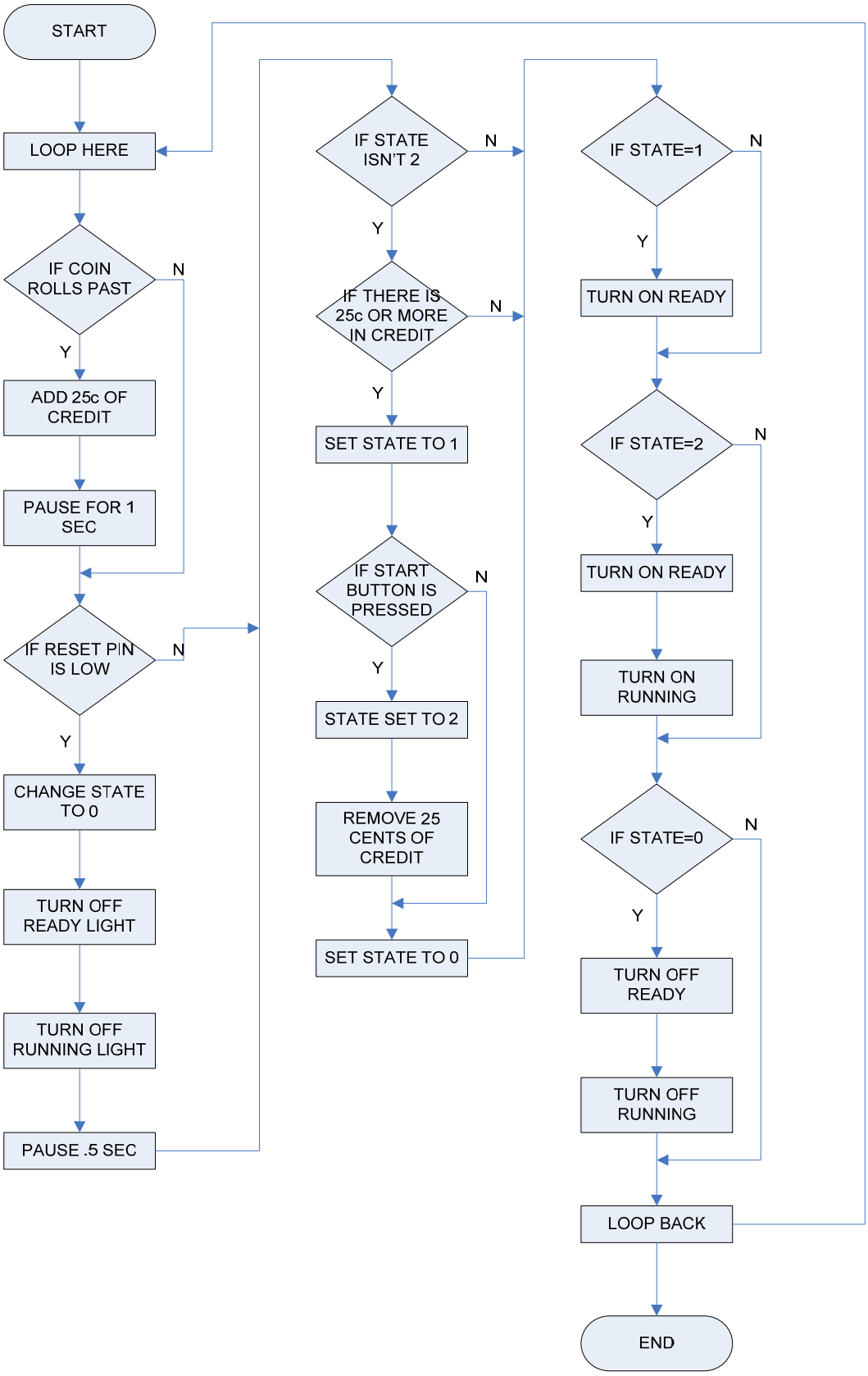

Group 23
Solenoids
Tess Bloom
Rev 1.0
4/14/2013
Page 3
Solenoid
Solenoid
Solenoid
Solenoid
Solenoid
Arduino UNO
Running
Reset
Score
Ready


Group 23
Score Board 7 Segment
Rev 1.0
4/10/2013
Page 1
Kelly Banta
7 Segment Display
7 Segment Display
OP Amp
PIC16F88
+5v/0v (reset) from Arduino
Arduino

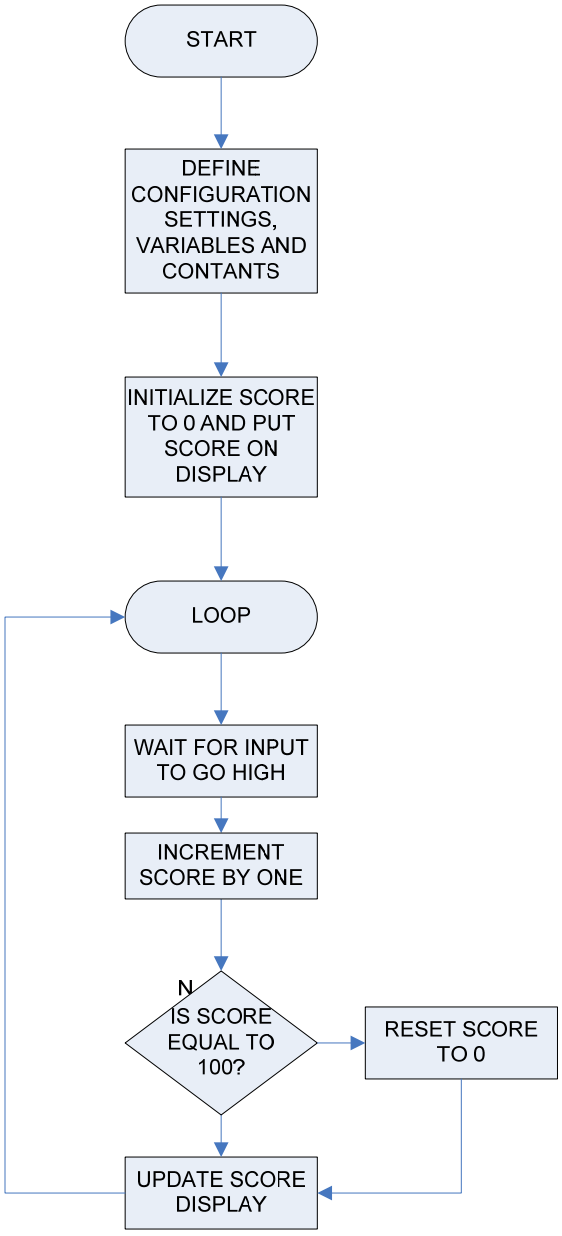
Arduino Flowchart



Coin Sensor PIC Flowchart



Scoreboard Flowchart



Moles Program

```
//variable delcerations:
//stores random number that is generated for which mole goes up
long randNumber1;
//stores the number of moles that have been "raised"
int moleup = 0;
//sets up pins for the 5 different moles on the Arduino in an array variable
int mole[6] = {7, 6, 5, 4, 2};
//stores the time that the 5 different moles were raised last
float moletimer[6] = {0, 0, 0, 0, 0};
//sets the pins for the A/D conversion of the hits signal
int molesensor[6] = {A0, A1, A2, A3, A4};
//stores the values of the signal
int molesensorvalue[6] = {0, 0, 0, 0, 0};
//sets the pin for the reset connecton between the Arduino and the two other PICs
int reset = 9;
//sets the pin for the score connection between the Arduino and the scoreboard PIC
int score = 10;
//sets the pin for the ready signal between the Arduino and the coin counting PIC
int ready = 11;
//sets the pin for the next signal to the audio board
int next = 12;
//stores tohe old value of the running pin
int runningprev = 0;
//stores the old values of the ready pin
int readyprev = 1;
//sets the pin to reset the audio board
int audioreset = 13;


//sets digital pin 1 as a serial out pin for the LCD screen communication
const int TxPin = 1;

//required to setup the LCD screen
//As a note to future students this serial stuff is kind of hard to work with and we had screen tearing from a bad sinc
//we only used this because it was all Radioshack had and we were close to the deadline if at all possible use a regular 16x2 display
```

```
//as seen on the recommended parts page on the Mech 307 website here is the link:
//http://mechatronics.colostate.edu/pic.html#Devices
#include <SoftwareSerial.h>
SoftwareSerial mySerial = SoftwareSerial(255, TxPin);



//set up area only runs once at startup
void setup() {
  //sets pin mode for some of the connection
 //next, audioreset, and the serial connection are output only
  pinMode(next, OUTPUT);
  pinMode(audioreset, OUTPUT);
  pinMode(TxPin, OUTPUT);
  //sets the serial pin high by default
  digitalWrite(TxPin, HIGH);

  //starts Serial connunication with LCD and set the baud rate at 9600b/s
  mySerial.begin(9600);
  //pause .1 seconds
  delay(100);
  mySerial.write(12);               // Clear
  mySerial.write(17);               // Turn backlight on
  delay(5);                       // Required delay
  mySerial.print("Hello, world...");  // First line
  mySerial.write(13);               // Form feed
  mySerial.print("from Group 23");   // Second line
  mySerial.write(212);               // Quarter note
  mySerial.write(220);               // A tone
  delay(3000);                    // Wait 3 seconds
  mySerial.write(18);               // Turn backlight off

  //set mole pins as outputs
  for (int i=0; i<5; i++) {
    pinMode(mole[i], OUTPUT);
  }
  //set reset pin as output
  pinMode(reset, OUTPUT);
  //set reset high to make sure that the scoreboard and coins counter are zeroed
  digitalWrite(reset,HIGH);
```

```
  //set score pin to output
  pinMode(score, OUTPUT);
  //set audioreset pin low to reset it (audio board has negative logic)
  digitalWrite(audioreset, LOW);

  //seed the random number generator with the milli seconds running times the static in the
analog port 5 for redundency
  randomSeed(analogRead(5)+(millis()));
}



//this section runs over and over again forever
void loop() {
  //update mole sensor values
  for(int k=0; k<5; k++) {
   //stores the mole sensor value var to the current level
   molesensorvalue[k] = analogRead(molesensor[k]);
  }
  //store the current value of running wire in the var
  int running = digitalRead(8);
  //store the current value of the ready wire in the var
  int readyval = digitalRead(ready);
  //if not ready but was ready last round
  if (readyval == 0 & readyprev == 1) {
    //add menu for LCD saying "Whac-a-prof 25 cents per game"
    mySerial.write(12);                // Clear
    mySerial.write(17);                // Turn backlight on
    mySerial.print("Whac-a-Prof");      // First line
    mySerial.write(13);                // Form feed
    mySerial.print("25 cents/game");    // Second line
    delay(100);                        // pause .1 seconds
  }
  //if not running but ready
  if (readyval == 1 & running == 0) {
    //LCD message saying "Whac-a-prof ready"
    mySerial.write(12);                // Clear
    mySerial.write(17);                // Turn backlight on
    mySerial.print("Whac-a-Prof");      // First line
    mySerial.write(13);                // Form feed
```

```
    mySerial.print("Ready");           // Second line
    delay(100);                       // pause .1 seconds
  }
  //if running but not running during the last run
  if (runningprev == 0 & running == 1) {
    //add menu saying "Whac-a-Prof running"
    mySerial.write(12);               // Clear
    mySerial.write(17);               // Turn backlight on
    delay(5);                        // Required delay
    mySerial.print("Whac-a-Prof");     // First line
    mySerial.write(13);               // Form feed
    mySerial.print("Running");         // Second line
    digitalWrite(next,HIGH);          // tell audio board to play countdown song
    delay(50);                       // delay .05 seconds to make sure the signal gets to the audio board
    digitalWrite(next,LOW);           // stop sending the signal
    delay(9950);                      // pause 9.95 seconds to let the song play out
    digitalWrite(next,HIGH);          // tell audio board to play game sound (Benny Hill theme
song edited)
    delay(50);                       // delay .05 seconds to make sure the signal gets to the audio board
    digitalWrite(next,LOW);           // stop sending the signal
  }
  //store current ready and running values for next round
  runningprev = running;
  readyprev = ready;
  //if running wire is active
  if (running == 1) {
    //if less than 60 moles have been raised
    if (moleup < 60) {
      //if it has been a multiple of .2 seconds since startup
      if (millis()%200==0) {
        //generate a random number between 0 and 9 inclusive
        randNumber1=random(0,10);
        //if a mole was selected (0-4)
        if(randNumber1 <5) {
          //activates the selected mole so that it will go up if it isn't already up
          digitalWrite(mole[randNumber1], HIGH);
          //increment the moleup counter
          moleup++;
          //set the timer to the current number of second since start
          moletimer[randNumber1] = millis()/100;
```

```
      }
    }
    //cycle through each mole one at a time
    for (int j=0; j<5; j++) {
     //check if the mole is up
     int up = digitalRead(mole[j]);
     //if the mole is up
     if(up == 1) {
      //if the mole has been on for one second or more
      if((millis()/100) - moletimer[j] == 10) {
       //turn off or lower the mole
       digitalWrite(mole[j], LOW);
      }
      //see if mole has been hit based on A/D conversion of the state of the mole
      else if(molesensorvalue[j] <770) {
       //if the mole had been up for more than .2 seconds (this is because while
       //the mole goes up it looks like it is being hit to the Arduino
       if(millis()/100-(moletimer[j]) > 2){
        //lower/turn off mole
        digitalWrite(mole[j], LOW);
        //send a pulse to the scoreboard PIC so that the scoreboard will increment
        digitalWrite(score, HIGH);
        digitalWrite(score, LOW);
       }
      }
     }
    }
   } else {  //if the mole up counter is at or above 60 it is time to end the round (mole up should
never go over 60 just reach it)
    //make sure the audio board is ready to proceed to the next song
    digitalWrite(next,HIGH);
    //cycle through all of the moles and turns them off
    for (int j=0; j<5; j++) {
     digitalWrite(mole[j], LOW);
    }
    //pause .5 seconds
    delay(500);
    //have the audio board play the ending song
    digitalWrite(next,LOW);
    //delay 3 seconds for sound
```

```
    delay(3000);
    //reset the scoreboard PIC and tell the coins counter that the round is over
    digitalWrite(reset, LOW);
    //reset the Audio board for the next round
    digitalWrite(audioreset,HIGH);
    //pause .5 seconds for the other PICs to reset
    delay(500);
    //return to normal
    digitalWrite(reset, HIGH);
    digitalWrite(audioreset,LOW);
    //reset mole up counter for the next round
    moleup = 0;
   }
  }
}
```

Coin Program

```
'*****************************************************************
'*  Name    : COINS.BAS                                          *
'*  Author  : Clinton Knackstedt                                 *
'*  Notice  : Copyright (c) 2013 [select VIEW...EDITOR OPTIONS] *
'*          : All Rights Reserved                                *
'*  Date    : 4/6/2013                                           *
'*  Version : 1.0                                                *
'*  Notes   :                                                    *
'*          :                                                    *
'*****************************************************************

'These three lines change the configuration setting in the menu before uploading the code to the
PIC
'They allow you to upload it without changing any of the settings.
#CONFIG
   __CONFIG _CONFIG1, _INTRC_IO & _PWRTE_ON & _MCLR_OFF & _LVP_OFF
#ENDCONFIG

' Set the internal oscillator frequency to 8 MHz
DEFINE OSC 8
OSCCON.4 = 1
OSCCON.5 = 1
OSCCON.6 = 1


'Turn off the A/D convertors (required for the PIC16F88, to use associated pins for digital I/O)
ANSEL = 0
```

```
'declare vars
'start button is a N0 button
startbutton var PORTB.0

'This is a reset line that allows the Arduino
'to tell the PIC that it is time to reset for the next round
reset var PORTB.1

'handshake connection that tell the Arduino that there is
'money available when high
ready var PORTB.3
'handshake connection that tell the Arduino to start the round when high
running var PORTB.2

'input from photo transistor
sensor1 var PORTB.4

'var that stores the current amount of money that is in the system
money var byte
'variable that stores the state that the menu screen should be in numerically
state var byte

'startup commands resets the menu state and the money value
 state = 0
 money = 0



mainloop:
  'count the money
   if(sensor1 == 0) then          'when coin goes through
     money = money + 25           'add 25 cents
     pause 1000                 'wait to let the coin roll past
   endif

   if (reset == 0) then          'if Arduino is telling the PIC to reset
     state =  0                 'reset the menu to default
     low ready                        'set ready low to reset signals to Arduino
     low running                      'set running low so the Arduino doesn't start another round
     pause 500                        'pause to let Arduino finish its reset
   endif
```

```
if(state != 2) then          'if not running
  if(money >= 25) then       'if money has at least 25 cents
    state = 1                'ready
    if(startbutton == 1) then  'if the startbutton is pressed
      state = 2                'running
      money = money - 25       'subtract money for round cost
    endif
  else
    state = 0                'waiting for money
  endif
endif

if(state == 1) then          'if ready
  high ready                 'set ready signal to Arduino high
endif

if(state == 2) then          'if running
  High running               'set ready an d running high for the Arduino telling it to start
  high ready
endif

if(state ==0) then           'if waiting for money set outputs to Arduino low
  low ready
  low running
endif

Goto mainloop                ' Do it forever
End
```

Scoreboard Program

```
'****************************************************************
*
'*  Name    : SCOREBOARD.BAS
*
'*  Author  : [Kelly Banta]
*
'*  Notice  : Copyright (c) 2013 [select VIEW...EDITOR OPTIONS]
*
'*          : All Rights Reserved
*
'*  Date    : 3/10/2013
*
'*  Version : 1.0
*
'*  Notes   :
*
```

```
'*          :
*
'*****************************************************************
*
'Define configuration settings (different from defaults)
#CONFIG
    __CONFIG _CONFIG1, _INTRC_IO & _PWRTE_ON & _MCLR_OFF &
_LVP_OFF
#endconfig

'Set the internal oscillator frequency to 8 MHz
define OSC 8
OSCCON.4 = 1
OSCCON.5 = 1
OSCCON.6 = 1


'Turn off the A/D converters (required for the PIC16F88, to use
associated pins for digital I/O)
ANSEL = 0

'Declaring variables:

'counts the score to be printed on the display
score var byte              'counter variable
scoretens var byte          'tens digit
scoreones var byte          'ones digit

'array to contain codes for each number 0-9
pins var byte[10]

'assign each element of the array Ð for each number, tell the
appropriate Ôpins corresponding to segments of the display to go
high or low
pins[0] = %11011101
pins[1] = %00010100
pins[2] = %11001110
pins[3] = %01011110
pins[4] = %00010111
pins[5] = %01011011
pins[6] = %11010011
pins[7] = %00011100
pins[8] = %11011111
pins[9] = %00011111

'Delcare inputs/outputs
TRISB = %00100000        'PortB.5 is input from Arduino
```

```
TRISA = %00000000


'initialize score to zero
score = 0

'update the output to the seven-segment display in case it is
sending out something other than 00
gosub Updatepins


'main loop: increment score when Arduino sends a pulse to PIC
myloop:
  if(PORTB.5 ==1) then        'when input goes high
    if(score<100) then        'if the score is below 100
     score = score + 1        'increment score
    endif
    if(score=100) then        'when score reaches 100, change it
back to 0 (which should never happen)
     score = 0
    endif
    gosub Updatepins          'update the seven-segment display
to show the new score
    pause 250            '0.25 sec delay to account for pulse
from Arduino
   endif
goto myloop              'loop forever


'subroutine to update seven-segment display
Updatepins:
 'value of tens digit
  scoretens=score/10
 'value of ones digit
 scoreones=score-(scoretens*10)

 'output the score
 PORTA= pins[scoreones]    'PORTA: ones digit
 PORTB= pins[scoretens]    'PORTB: tens digit
return 'go back to the place we left in the main loop

end
```