DESIGN PROJECT 2 – JACK IN THE BOX

Jeremy Blum – INFO4320

DESIGN GOALS

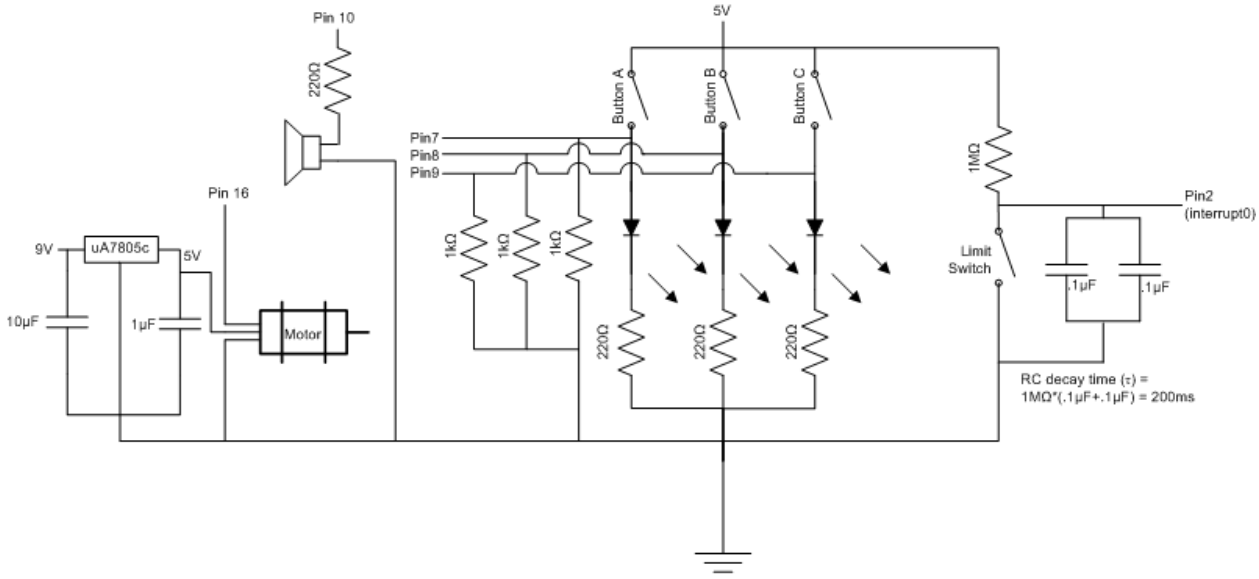
For this project, my goal was to develop a protected Jack-in-the-Box. In other words, I wanted a toy where you have to enter a special code or the box will not open.

1. Break-In Detection
    a. Use a limit switch under the lid to detect if the lid is being opened without a code being entered
    b. If the lid is opened like this, the box will sound an alarm using the built-in speaker, and the Jack will not come up
2. Code Requirement
    a. Use three momentary push-buttons to accept a 6 digit code
    b. When this code is entered correctly, the box will play a happy tune on the speaker, and the jack will popup
    c. To close the box after opening it, press all three buttons simultaneously. The box will then start checking for new inputs, and the alarm will be re-enabled.
    d. When this code is entered incorrectly, the box will make a sad noise, and the jack will not come up
3. Construction
    a. The box is designed in Alibre, then laser cut
    b. An ordinary door hinge is used as the hinge for the box
    c. A servo motor and attached arm are used to both open the box and to pop the Jack up
    d. An internal speaker plays alarm sounds and tones when entering codes
    e. A limit switch under the lid detect when the lid is opened or closed
    f. A handle was created using staples – it is necessary to use as extra weight on the limit switch
    g. The hinge, speaker, and switch are screwed directly into the laser cut acrylic using bolts and heat-molded threading that are designed to be melted right into holes in the acrylic
4. Software
    a. The software is specifically designed to make it easy to change settings
    b. It is even possible to change the passcode and the length of the passcode by only changing two constants
    c. The software uses the pitches and servo libraries to control the servo and speaker respectively

HARDWARE USED

1. Socket Cap Screws: http://www.mcmaster.com/#91292a014/=6i7q7a
2. Threaded Heat Inserts: http://www.mcmaster.com/#94180a321/=6i7q7d
3. No-Mortise Fastinge Door Hinge: Available at most Hardware Stores
4. Limit Switch: http://www.radioshack.com/product/index.jsp?productId=2049718
5. Servo Motor: http://www.jameco.com
6. Speaker: http://www.jameco.com

SCHEMATIC



There are several key components shown in the schematic. Starting at the left is the 5V regulator and motor. Because USB cannot provide enough current on its own, an auxiliary 9V battery is regulated to 5V to power the servo motor. The motor's position can be adjusted to a precise angle using a pulse-width-modulation (PWM) output from the Arduino. Next is the speaker – Its tone is adjusted based on the width of the PWM signal that it receives from pin 10. Using the pitches library, it is easy to program it to play a short tune. To the right of the speaker are the keypad buttons and their corresponding LEDs. When a button is depressed, it both illuminates the LED (to provide visual feedback to the user), and it sends a high signal to the corresponding arduino pin. Using software debouncing, these signals are analyzed and checked to see if they match the required password. The last portion of the circuit is the limit switch and accompanying debouncing circuitry. Because the switch is used as a hardware interrupt (to ensure intrusion attempts are always caught), software debouncing cannot be used. This is because the millis() and delay() function are inoperable within interrupt functions. To debounce the switch, a simple RC circuit is used, where the time decay constant was made to be 200ms, or approximately the amount of time that it takes for the button output to stop fluctuating. Since capacitors cannot charge instantaneously, their voltage output approaches zero as an exponential decay function while the button bounces. Its voltage only passes the threshold digital voltage once, resulting in a debounced signal.

ADDITIONAL USAGE SCENARIOS

While designing my Jack-in-the-Box, I realized that this design would actually be perfect for protecting something like a wedding ring. Instead of putting the "Jack" on the end of the acrylic rod, you could put jewelry of some kind. You would have to enter a passcode and your jewelery would be presented to you. In addition to the audible alarm, you could potentially have a camera take a picture of the intruder to further protect your jewelery.

SOFTWARE

Following is the Arduino Code for the Jack in the box. I will comment on various sections using this typeface.

```
//Jack-in-the-Box
//INFO4320
//JEREMY E. BLUM

#include <Servo.h>                        //Include the Servo Library
#include "pitches.h"                      //Include the Tone Library

Servo myservo;                            // create servo object to control a servo

const int doorInterrupt = 0;              // Interrupt for Door Trip Sensor  - Pin 2 = 0, Pin 3 = 1
const int doorPin       = 2;              // Pin number for door trip sensor
const int servoPin      = 6;              // Pin to control the Servo Motor
const int buttonPinA    = 7;              // Pin to first unlock button
const int buttonPinB    = 8;              // Pin to second unlock button
const int buttonPinC    = 9;              // Pin to third unlock button
const int speakerPin    = 10;             // Pin to Alarm Speaker

const int lidOpen       = 105;            //Servo Value to Open Lid
const int lidClosed     = 33;             //Servo Value to Close Lid

byte current_buttonA    = LOW;            //State of buttonA
byte old_buttonA        = LOW;            //Previous state of buttonA
byte current_buttonB    = LOW;            //State of buttonB
byte old_buttonB        = LOW;            //Previous state of buttonB
byte current_buttonC    = LOW;            //State of buttonC
byte old_buttonC        = LOW;            //Previous state of buttonC

const int len = 6;                         //DEFINE THE LENGTH OF THE COMBO ARRAY (and consequently the attempt
//array)
const int combo[len] = {1, 2, 3, 3, 2, 1}; //DEFINE THE COMBO ARRAY - The Button Press order to unlock the box
//(button A = 1, B = 2, C = 3)
int attempt[len];                          //Inialize the attempt array
```

Below are the short melodies that are played when the user enters a passcode. They can be defined quite easily thanks the pitches file, which associates actual notes with the PWM outputs

```
// Pass Tones:
const int melody_pass[] = {NOTE_C4,NOTE_G3,NOTE_G3,NOTE_A3,NOTE_G3,0,NOTE_B3,NOTE_C4};
const int noteDurations_pass[] = {4,8,8,4,4,4,4,4 };

//Fail Tones:
const int melody_fail[] = {NOTE_D4,NOTE_D3,NOTE_D2,NOTE_D1};
const int noteDurations_fail[] = {4,4,4,1};

void setup()
{
  myservo.attach(servoPin);             //Attaches the servo to the servo object

  pinMode(doorPin,    INPUT);           //Door trip sensor is an input
  pinMode(servoPin,   OUTPUT);          //The servo is an output
  pinMode(buttonPinA, INPUT);           //The first button is an input
  pinMode(buttonPinB, INPUT);           //The seconds button is an input
  pinMode(buttonPinC, INPUT);           //The third button is an input
  pinMode(speakerPin, OUTPUT);          //The fourth Button is an input

  attachInterrupt(doorInterrupt, alarm, RISING);    //Attach the door interrupt
```

The attempt array is used to hold the input attempts by the user

```
  //Initialize the attempt array with zeros
  for (int i = 0; i < len; i = i++) attempt[i] = 0;

  Serial.begin(9600);                   //Open Serial Connection for debugging

}
```

The alarm function is called whenever the limit switch interrupt is activated

```
void alarm()
{
  Serial.println ("ALARM!");
  tone(speakerPin, NOTE_G3, 1000);
```

```
  delayMicroseconds(1000000);
}
```

The bogus function is needed to throw away the interrupt value when the lid is closed after begin opened successfully.

```
void bogus()
{
}
```

This is the software debouncing function for the keypad buttons

```
byte read_button(byte old_value, int pin)              //Debounced Button Reading function
{
  byte current_button = digitalRead(pin);              //Read the Current Button State
  if ((old_value == LOW) && (current_button == HIGH))  //If it is transitioning to high...
  {
    delay(5);                                          //Wait 5ms
    current_button = digitalRead(pin);                 //Then read it's "high" value
  }
  if ((old_value == HIGH) && (current_button ==LOW))   //If it is transitioning to low...
  {
    delay(5);                                          //Wait 5ms
    current_button=digitalRead(pin);                   //Then read the value
  }
  return(current_button);                              //return the de-bounced button value
}
```

The which_button() function is used when checking button inputs to ensure they are in the right order

```
int  which_button()
{
  current_buttonA = read_button(old_buttonA, buttonPinA);
  current_buttonB = read_button(old_buttonB, buttonPinB);
  current_buttonC = read_button(old_buttonC, buttonPinC);
  if      ((old_buttonA == LOW) && (current_buttonA == HIGH)) return (1);
  else if ((old_buttonB == LOW) && (current_buttonB == HIGH)) return (2);
  else if ((old_buttonC == LOW) && (current_buttonC == HIGH)) return (3);
  else    return (0);
  old_buttonA = current_buttonA;
  old_buttonB = current_buttonB;
  old_buttonC = current_buttonC;
}
```

This plays the appropriate song when the users fails or succeeds in entering the code

```
void play_song(boolean pass)
{
  if (pass)
  {
    for (int thisNote = 0; thisNote < 8; thisNote++)
     {
       int noteDuration_pass = 1000/noteDurations_pass[thisNote];
       tone(speakerPin, melody_pass[thisNote],noteDuration_pass);
       int pauseBetweenNotes = noteDuration_pass * 1.30;
       delay(pauseBetweenNotes);
     }
  }
  else
  {
     for (int thisNote = 0; thisNote < 8; thisNote++)
     {
       int noteDuration_fail = 1000/noteDurations_fail[thisNote];
       tone(speakerPin, melody_fail[thisNote],noteDuration_fail);
       int pauseBetweenNotes = noteDuration_fail * 1.30;
       delay(pauseBetweenNotes);
     }
  }
}

void loop()
{

 myservo.write(lidClosed);                    //The lid should always start closed
 delay(15);                                   //Give the motor time to actuate

 //This will read the button press attempts!
 for (int i=0; i<len; i++)
 {
```

The following wait delay is why the limit switch must be an interrupt input

```
  while (which_button() == 0);               //Stalls while waiting for a button to be pressed
  attempt[i] = which_button();              //Adds button press to tracking array

  for (int i = 0; i < len; i = i++) Serial.print(attempt[i]);

  while (which_button() == attempt[i]);         //Wait for the button to be released
  Serial.println("");
}
```

After reading the buttons inputs, the values are checked against the passcode

```
//Was the correct Code Entered?
int correct = 0;
for (int i=0; i<len; i++)
{
  if (attempt[i] == combo[i]) correct++;
}
boolean good = false;
if (correct == len) good = true;

//If the Correct Code was entered...
if (good)
{
```

The interrupt must be detached before opening the lid, or the alarm will sound!

```
  detachInterrupt(doorInterrupt);    //Disable Alarm
  myservo.write(lidOpen);            //Open the lid
  Serial.println("PASS!");
  play_song(true);                   //Play a tune!

  //Keep lid open until all three buttons are pressed at once
  while (digitalRead(buttonPinA) == LOW || digitalRead(buttonPinB) == LOW || digitalRead(buttonPinC) == LOW);
  myservo.write(lidClosed);
  delay (2000);     //Give time for buttons to be released
```

Reattach the interrupt so the alarm will go off

```
  attachInterrupt(doorInterrupt, bogus, RISING);    //Attach the door interrupt
  attachInterrupt(doorInterrupt, alarm, RISING);
}
//If the wrong code was entered...
else
{
  Serial.println("FAIL!");
  play_song(false);                  //Play FAIL Tune

}
```

Do it all over again!

```
for (int i = 0; i < len; i = i++) attempt[i] = 0;  //Reset Attempt array to zeros
}
```