Pandora's Music Box

Good "Company" Inc, LLC

Daniel Creamer
dcreamer@mines.edu

Mitchell Waibel
mwaibel@mines.edu

December 10, 2015

1 Project Idea

The goal of this project was to create a prototype music box that read music off of a piece of paper so that the user can make the music box play whatever they want it to. There are six different subsystems in the music box, five controlled through the Arduino and one controlled through digital logic.

1.1 Full Circuit

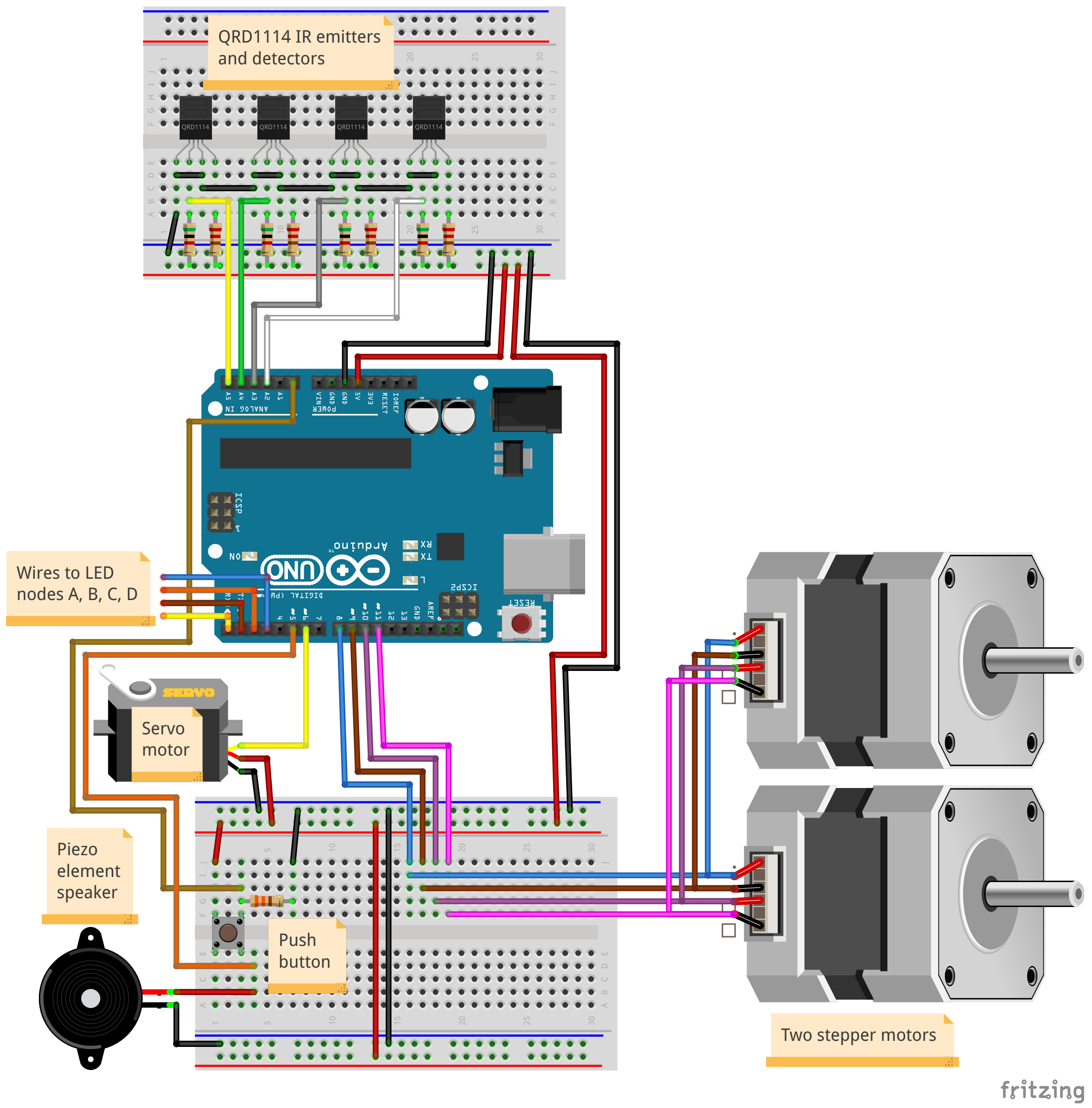


Figure 1: This is the full circuit for the music box. Each of these components are explained in detail in their specific subsystems later on in the report.

1.2 Main Code

This code is what initializes and runs the music box. The programs it calls in the loop() function are explained in detail in its corresponding subsystem.

```
boolean ledVal = LOW;     //ledVal initializes the entire system, everything runs while it is HIGH (boolean true)
const int buttonPin = A0;
const int ledPin = 13;
unsigned long previousMillis;
const unsigned long playTime = 30000;

#include <Servo.h>  //setup the lid servo motor
Servo lidServo;
const int lidPin = 6;

#include <Stepper.h>  //setup both the figurine motor and the music motor, they run at the same speed for simplicity
#define steps_per_motor_revolution 32           //our stepper has 32 steps per motor revolution but a 1/64 gear reduction
                                                 //ratio to increase steps per output rotation
#define steps_per_output_revolution 32*64
Stepper steppers(steps_per_motor_revolution, 8, 10, 9, 11);

#include "pitches.h"
int sensorPin1 = A5; //set up the pins used for the IR sensors
int sensorPin2 = A4;
int sensorPin3 = A3;
int sensorPin4 = A2;
int speakerPin = 5;

void setup(){
  pinMode(buttonPin, INPUT); //system starts on button press
  pinMode(ledPin, OUTPUT);    //just there to show if ledVal it "true"
  lidServo.attach(lidPin);    //tell the Arduino what pin the servo is on, not needed for the steppers
  pinMode(sensorPin1, INPUT);
  pinMode(sensorPin2, INPUT);
  pinMode(sensorPin3, INPUT);
  pinMode(sensorPin4, INPUT);
};

/*
the program runs through the button system to continuously check for a button press, then tells the
lid servo to open or close based on the value of ledVal. ledVal also controlls whether the stepper motors
are running.
*/
void loop(){
  boolean start = button(); //declares start as the returned value of button() (either HIGH or LOW)
  lidMotor(start);            //all other functions run while start is HIGH and stops while start if LOW
  motors(start);
  if(start == HIGH){          //plays music when start is HIGH
    music();
  }else{
    noTone(speakerPin);
  };
};
```

2 Subsystems

2.1 Button

2.1.1 Circuit

This subsystem is a simple push button with one pin connected to $5V$ and the other pin connected to ground through a $330k\Omega$ resistor. The Arduino reads the voltage over the resistor, which gives a HIGH reading when the button is pressed and a LOW reading when the button isn't pressed.

2.1.2 Coding

```
 1 //button press changes the value of ledVal, and turns on the LED built into
       pin 13 accordingly.
 2 boolean button(){
 3   unsigned long currentMillis = millis();
 4   if(analogRead(buttonPin) > 900){
 5     previousMillis = currentMillis;
 6     ledVal = !ledVal;
 7     delay(250);
 8   };
 9   digitalWrite(ledPin, ledVal);
10   if(ledVal == HIGH){
11     if(currentMillis - previousMillis >= playTime){
12       previousMillis = currentMillis;
13       ledVal = LOW;
14     };
15   };
16   return ledVal; //gives ledVal to the rest of the program to run while it
         is HIGH
17 };
```

This code reads the voltage from pin A0, and when the voltage is over 900 (out of a range of 0 to 1023 where 5V is 1023), the value of ledVal (an arbitrary variable) changes from the previous value. This allows the program to begin and end at the push of the button. The digitalWrite portion in line 9 simply turns on an LED on the board so we can see if the program should be running. Lines 10-15 use the millis() function, which returns the time since the program initialized, to create a timer in which the music box will stop itself after a certain amount of time.

The millis() function is the only way to create the timer without completely stopping the program. Te delay function canno be used because that function stops the program from continuing for that amount of time.

The program then returns the value of ledVal, which in turn starts the rest of the functions in the music box.

2.2 Lid

2.2.1 Circuit

Connecting the servo motor is quite easy. The ground and power of the servo is connected to the ground and power on the breadboard, and the signal is then connected to pin 6. This pin must be a pin that output PWM signals.

2.2.2 Coding

```
void lidMotor(boolean play){
  int lidClosed = 90;  //tell the arduino where open and closed is
  int lidOpen = 170;

  if(play == HIGH){   //opens of closes the lid based on the given value of
      play
    lidServo.write(lidOpen);
  }
  else{
    lidServo.write(lidClosed);
  };
};
```

This code tells the Arduino at what positions in the servo that the lid is open or closed. Then, according to the value of play, that is given to it when the program lidMotor is called from the main program, it moves the servo motor to the position of either lidOpen or lidClosed.

2.3 Figurine and Music Sheet Motor

2.3.1 Circuit

The stepper motors require four inputs which alternate the current to four separate parts of the motor such that each time the motor changes current, the shaft moves one step around a revolution. For the motors we got, they required their own power and ground, which is not shown in the full circuit in Figure 1.

Both the motor for the figurine and the motor for the music scroll were connected to the same inputs because they needed about the same speed and it simplified the circuitry and coding substantially. The four inputs into the motor were connected to pins 8, 9, 10, and 11.

2.3.2 Coding

```
void motors(boolean play){
  steppers.setSpeed(800); //sets the speed of the steppers. Should be <=700
      based on https://arduino-info.wikispaces.com/SmallSteppers
  steppers.step(play);     //steps the motor if play is HIGH. only does one
     step per program cycle, so it cycles through the
                           //other programs, expecially to check for another
                                  button press
};
```

The code takes the boolean value play and, if it's HIGH (or true), the stepper moves one step. If it's LOW, the stepper doesn't move. Moving one step at a time allows for the motor to move while also doing the many other things the code controls.

2.4 Music Reader

2.4.1 Circuit

This is one of the more complex part of the circuit, though still quite simple. The components used are the QRD1114, which are a combined IR emitter and detector. The emitter emits IR light, and whether the paper it bounces off is white or black, the detector closes or opens the transistor respectively to let current through. This component is hooked up with two pins connected to ground and the other two connected to power through a 220Ω or $5k\Omega$ as shown in Figure 1, on the breadboard above the Arduino. The voltage of the transistor is then read by the Arduino and sent to other parts of the circuit.

2.4.2 Coding

```
 1 void music(){//plays notes based on the binary input from the IR detectors
 2   int note[] = {
 3     digitalRead(sensorPin1), digitalRead(sensorPin2), digitalRead(
         sensorPin3), digitalRead(sensorPin4)
 4   };
 5   if(note[0] == 0 && note[1] == 0 && note[2] == 0 && note[3] == 1){
 6     tone(speakerPin, NOTE_F3);
 7   }else if(note[0] == 0 && note[1] == 0 && note[2] == 1 && note[3] == 1){
 8     tone(speakerPin, NOTE_G3);
 9   }else if(note[0] == 0 && note[1] == 0 && note[2] == 1 && note[3] == 0){
10     tone(speakerPin, NOTE_A3);
11   }else if(note[0] == 0 && note[1] == 1 && note[2] == 1 && note[3] == 0){
12     tone(speakerPin, NOTE_B3);
13   }else if(note[0] == 0 && note[1] == 1 && note[2] == 1 && note[3] == 1){
14     tone(speakerPin, NOTE_C4);
15   }else if(note[0] == 0 && note[1] == 1 && note[2] == 0 && note[3] == 1){
16     tone(speakerPin, NOTE_D4);
17   }else if(note[0] == 0 && note[1] == 1 && note[2] == 0 && note[3] == 0){
18     tone(speakerPin, NOTE_E4);
19   }else if(note[0] == 1 && note[1] == 1 && note[2] == 0 && note[3] == 0){
20     tone(speakerPin, NOTE_F4);
21   }else if(note[0] == 1 && note[1] == 1 && note[2] == 0 && note[3] == 1){
22     tone(speakerPin, NOTE_G4);
23   }else if(note[0] == 1 && note[1] == 1 && note[2] == 1 && note[3] == 1){
24     tone(speakerPin, NOTE_A4);
25   }else if(note[0] == 1 && note[1] == 1 && note[2] == 1 && note[3] == 0){
26     tone(speakerPin, NOTE_B4);
27   }else if(note[0] == 1 && note[1] == 0 && note[2] == 1 && note[3] == 0){
28     tone(speakerPin, NOTE_C5);
29   }else if(note[0] == 1 && note[1] == 0 && note[2] == 1 && note[3] == 1){
30     tone(speakerPin, NOTE_D5);
31   }else if(note[0] == 1 && note[1] == 0 && note[2] == 0 && note[3] == 1){
32     tone(speakerPin, NOTE_E5);
33   }else if(note[0] == 1 && note[1] == 0 && note[2] == 0 && note[3] == 0){
34     tone(speakerPin, NOTE_F5);
35   }else{
36     noTone(speakerPin);
37   };
38 }
```

The portion of this code that is used for the music reading is just lines 2-4. It makes a list of the digitalRead from Arduino pins A2 through A5 to tell whether the paper above it is black (which gives a HIGH) or white (which gives a LOW). This list is then used for the speaker() poriton of the code.

2.5 Speaker

2.5.1 Circuit

The speaker we used was a simple piezo element, with the negative pin connected to ground and the positive pin connected to pin 5 on the Arduino. No other components were necessary for this part of the circuit.

2.5.2 Coding

This code is combined into the code under section "Music Reader." The code for outputting to the speaker begins at line 5.

This code may look complicated, but really it only looks bad because the Arduino has to check through all the cases of the possible inputs in order output the correct frequency to the speaker. Using the if else statements, it determines what the correct output to the speaker. Depending on this, it gives a frequency to the speaker using the tone() command. The frequencies that are used (NOTE_XX) are defined in a separate header file to make the code more compact.

A useful feature of this code is that it is made using gray code, so that if it detects a note that is slightly different from the actual note, the frequency will only be slightly off.

2.6 Lights

The intention for the light subsystem was to arrange a light system which would generate a corresponding light for each note. This was arranged to generate different colors using a three color LED, using the sixteen unique combinations used to generate notes. A digital circuit was designed to use two three color LED's to generate several different color combinations, however it became apparent that a lack of digital circuitry and the necessary elements would limit the subsystem to using a single three color LED.

2.6.1 Circuit

Different digital circuits were designed to control each input to the LED. The circuit was generated by choosing the outputs for the different colors. Then these outputs were randomly assigned to the different combinations of inputs that would eventually be coordinated with the signals associated with the different notes. By using Karnaugh mapping the coordinated light show could be produced by a digital circuit rather than using internal code of the RedBoard and preserving more of the data for the other routines which would need to be run independently. Its important to note that the circuit was originally intended to function for a pair of three color LED's, but that upon discovering the sufficient components were not available (and in all likelihood not requisite for a proof of concept design) a single LED is being used and hence the design work being shown below is only for a single three color LED.

A	B	C	D	B1	G1	R1
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	0	1
1	0	1	1	0	0	1
1	1	0	0	0	0	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	0	1	1

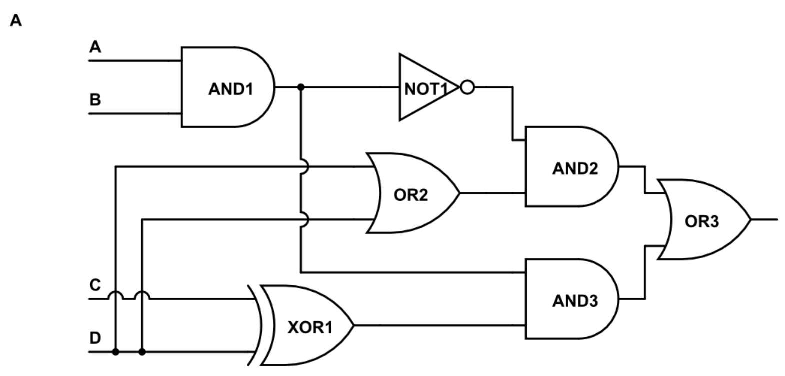
Then Karnaugh mapping was performed for each input node of the LED. The first of which was performed for the blue input node.

B1	00	01	11	10
00	0	0	0	0
01	1	0	1	0
11	1	0	0	0
10	1	0	1	0

This mapping then generates the logic equation (inverted inputs are labeled with an apostrophe).

$$B1 = \overline{A} * \overline{B}(C + D) + AB(C \oplus D) \tag{1}$$

The Circuit created is shown as an image here.



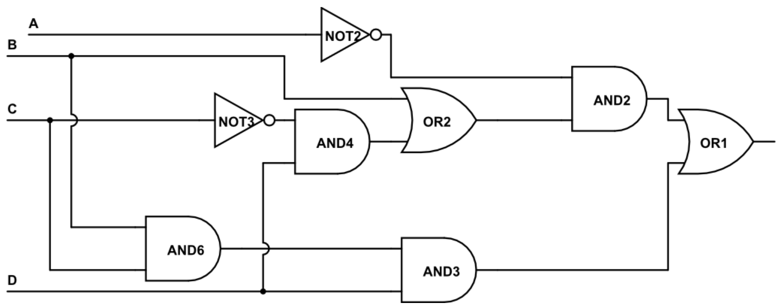
This process was then repeated for the Green input node.

G1	00	01	11	10
00	0	1	0	0
01	1	1	0	0
11	0	1	1	0
10	0	1	0	0

From this table the logic equation is represented below.

$$G1 = \overline{A}(B + \overline{C}D) + (BC)D \tag{2}$$

The Circuit created is shown as an image here.



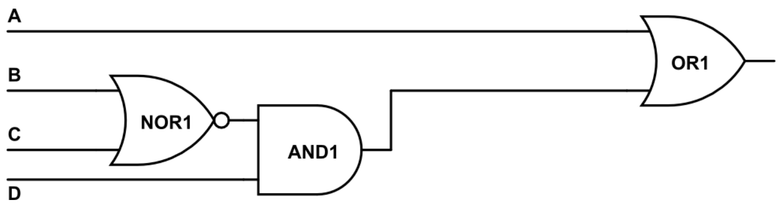
Finally the process is repeated for the red pin of the LED.

R1	00	01	11	10
00	0	0	1	1
01	1	0	1	1
11	0	0	1	1
10	0	0	1	1

This produces a logic equation

$$R1 = A + (\overline{B + C})D \tag{3}$$

This then represents the circuit pictured below.



3 Issues

To begin the first major issue confronted was finding motors which would be capable of generating the several different motions that were needed for a functioning music box. Originally equipped with the servo-motors provided in the kits proved a challenge as several of the desired motions required a full 360 degrees of motion, which the servos are not capable of without some forcible modification. This problem was resolved through the use of stepper motors which combine the precise control used to program servo-motors and then full 360 degrees of motion needed for both the music reader component and the figurine platform spinner.

Another major obstacle was devising a system which would be capable of producing music. It was determined early on to use small strips of paper to store the songs which would be played, and that a system would be devised to read the notes off the sheet and play the notes. Originally, the intention was to use infrared LED's and infrared photo-diodes, which when obstructed by the black ink of the markings on the sheet music would produce a signal that would be used to produce the different notes that the box would play. Instead, more complex components were found which would emit their own light, gauge the reflection of this light off a surface and then generate output current. The advantage of these elements being that when they shut off definitively when the reflection changed.

Unfortunately several issues persisted through to the creation of the prototype. The most apparent of these issues is the tendency of the box to play continuously, as it will detect individual notes but they are obscured by continuous noise generated by the system. The other pressing problem with the system is the speaker's capable mimicry of a DSL internet connection possessed by a vengeful demon. However, the system does produce (with a broad definition) music.

Another final limitation is simply the Arduino's power capability. The system designed uses the Arduino to power three separate motors. The power demands of these motors and the other system appears to tax the Arduino somewhat as each of the motors seems to spin lethargically, notably the motor spinning the figurine jerks slowly largely preventing the delicate spinning that one might associate with a more traditional music box. This was resolved by using an external power source such that the Arduino wasn't overwhelmed by the load of the three engines. In a more mobile solution a system of batteries would enable the system to behave entirely wirelessly.